UAB MedDream
K. Petrausko st. 26,
LT-44156, Kaunas,
Lithuania
info@meddream.com
www.meddream.com

MedDream (version 8.9.1)

Install manual of Viewing functionalities

(version 8.9.1, 2026-10-01)

© 2026, MedDream UAB, Kaunas

All rights reserved in the event of granting of patents or registration as a utility patent.

All names of companies and products mentioned in this manual may be trademarks or registered trademarks. References to products of other manufacturers are for information purposes only. Such references are intended neither as an approval nor a recommendation of these products. MedDream UAB accepts no liability for the performance or use of such products.

Other brand names, software and hardware names used in this manual are subject to trademark or patent protection. The quoting of products is for informational purposes only and does not represent a trademark misuse.

This Install Manual is protected by copyright. Unless expressly authorized in writing, dissemination, duplication or other commercial exploitation of this documentation set or communication of its contents or parts of it is not permitted. In case of infringement, the violator may be liable to pay compensation for damages.

Specifications due to technical developments are subject to change. This Install Manual is not subject to the revision service. Please contact the manufacturer or authorized dealer to request the latest edition of Install Manual.

Table of Contents

1 Introduction **1**
1.1 Installation prerequisites . . . 1

2 Requirements **2**
2.1 Server-side requirements . . . 2
2.1.1 Minimal hardware requirements . . . 2
2.1.2 Software requirements . . . 3
2.1.3 Time synchronization . . . 3
2.1.4 Licensing (Linux) . . . 4
2.2 Client-side requirements . . . 4
2.3 Telemetry and statistical data collection . . . 5
2.4 Decommissioning of MedDream . . . 5

3 Integration with PACS **6**
3.1 Integration modes . . . 6
3.1.1 Direct access mode on a different host . . . 6
3.1.2 Saving measurements, Key Objects, segmentations and screenshots . . . 7
3.1.3 Multiple PACSes in parallel . . . 8
3.2 Direct access to PACS database mode . . . 9
3.2.1 PacsOne . . . 9
3.2.2 DCM4CHEE . . . 17
3.2.3 ClearCanvas . . . 25
3.2.4 Conquest . . . 27
3.3 Integration over Web or other APIs . . . 29
3.3.1 Orthanc . . . 29
3.3.2 DICOMweb . . . 35
3.3.3 Amazon S3 . . . 53
3.3.4 AWS HealthImaging . . . 62
3.4 DICOM mode . . . 66
3.4.1 DICOM mode notes . . . 66
3.4.2 MedDream configuration for DICOM mode . . . 67
3.5 File system access mode . . . 73
3.5.1 File system mode notes . . . 73
3.5.2 MedDream configuration for File system mode . . . 74
3.6 Non-DICOM file system mode . . . 76
3.6.1 Non-DICOM mode notes . . . 76
3.6.2 MedDream configuration for Non-DICOM file system mode . . . 77

4 Image Access from hospital information system (HIS) **78**

5 Deployment **80**
5.1 Configuration . . . 80
5.1.1 Essential configuration and the first run . . . 80
5.1.2 System.json . . . 83
5.1.3 EcgAnnotations.json . . . 97
5.1.4 The ECG print template . . . 99
5.1.5 Dicomprint.json . . . 102
5.1.6 Built-in authentication and authorization . . . 106
5.1.7 Predefined login credentials . . . 115
5.1.8 The Reporting function . . . 115
5.1.9 Adjusting Java memory limits . . . 119
5.1.10 Enabling SSL . . . 121
5.1.11 Setting up ATNA event logging . . . 123
5.1.12 The external links plugin . . . 124
5.1.13 Reference for remaining configuration parameters . . . 129

5.1.14 Opening the H2 Console . . . 167
5.1.15 Using MySQL/MariaDB or PostgreSQL for Spring Batch and Quartz Scheduler . . . 168
5.1.16 Changing the HTTP context path . . . 170
5.1.17 Adjusting a reverse proxy . . . 170
5.1.18 Caching studies via DICOM Listener . . . 171
5.1.19 Monitoring via Prometheus/Grafana . . . 173
5.1.20 Implementing per-user settings . . . 174
5.1.21 Custom columns in Patient History / Patient Studies . . . 176
5.1.22 Passing configuration via environment variables . . . 179
5.1.23 Setting up Azure SSO with SAML 2.0 . . . 179
5.1.24 Setting up Keycloak SSO with SAML 2.0 . . . 182
5.1.25 Avoiding service restart after some configuration changes . . . 183
5.1.26 Changing the log level without a restart . . . 184
5.2 Running as a service . . . 185
5.2.1 Windows . . . 185
5.2.2 Linux (System-V init) . . . 185
5.2.3 Linux (systemd) . . . 186
5.2.4 Linux (upstart) . . . 187
5.3 JRE or JDK? . . . 188
5.4 Verification checklist . . . 188
5.5 Option: dockerized environment . . . 190
5.6 Updating the license . . . 190
5.7 Cleaning server-side caches . . . 190
5.8 Creating a dump of an H2 database . . . 191
5.9 Restoring an H2 database from a dump . . . 191
5.10 File-based H2 storage and non-default configRoot under Windows . . . 192
5.11 Using strong passwords in MySQL/MariaDB accounts . . . 192
5.12 Validation of MedDream.jar signatures . . . 192
5.13 Adding/updating a third-party library in MedDream.jar . . . 193
5.14 Backing up MedDream . . . 193
5.14.1 Locations and files . . . 193
5.14.2 Backup frequency . . . 195
5.15 Cleaning preparation cache via persistent index (experimental) . . . 196
5.16 DICOM Print . . . 198
5.17 High-Availability configuration . . . 198
5.17.1 Load balancer configuration . . . 198
5.17.2 MedDream Viewer nodes configuration . . . 199
5.17.3 Redis / Valkey configuration . . . 200
5.17.4 Shared fast storage configuration . . . 200
5.17.5 MedDream Token Service nodes configuration . . . 200
5.17.6 High Availability behavior . . . 201
5.17.7 Validation Checklist . . . 201
5.17.8 Notes & best practices . . . 202

6 Additional software **202**
6.1 Browser plugin . . . 202
6.2 Writing an ISO file to a USB stick . . . 203
6.3 Alternative to MedDreamBurn . . . 203

7 Security considerations **203**
7.1 Default passwords . . . 204
7.2 Search engines . . . 204
7.3 DCM4CHEE 2.x . . . 204
7.4 Embedding into IFRAME etc. . . . 204
7.5 php.ini . . . 205
7.6 Browser's cache . . . 208
7.7 Firewalls . . . 208
7.8 Summary: Minimum relevant IT security requirements . . . 209

8 Localization **211**

9 Rebranding **212**
9.1 Changing the progress bar color . . . 216

10 Troubleshooting **217**
10.1 Preventive measures . . . 217
10.2 Antivirus software . . . 218
10.3 Log files of the Java-based core . . . 218
10.4 Video conversion . . . 219
10.5 Client side: browser download path selection . . . 220
10.5.1 Google Chrome . . . 220
10.5.2 Mozilla Firefox . . . 220
10.5.3 Microsoft Edge . . . 220
10.5.4 Safari . . . 221
10.6 Client side: enabling clipboard operations . . . 221
10.7 Client side: monitor calibration . . . 221
10.8 Symptom: on Linux the loading pauses just after logging in, possibly with wrong webpage colors . 221
10.9 Symptom: queries to MS SQL Server take seconds from Java but milliseconds from Management Studio . . . 222
10.10 Symptom: video not playable on Firefox under Windows . . . 222
10.11 Symptom: strange “class not found” exceptions when restrictions are present in the token . . . 223
10.12 Symptom: “Unable to determine Dialect without JDBC metadata” when connecting to a database 223
10.13 Symptom: MPR views show as black screen in Chrome . . . 224
10.14 Known limitations and their solutions . . . 224
10.14.1 Share functionality skips some objects . . . 224

A Appendix: Configuration changes since 8.8.1 **225**
A.1 Core service command line . . . 225
A.2 application.properties, generic . . . 225
A.2.1 New options or values . . . 225
A.2.2 Options that are handled differently . . . 226
A.2.3 Removed options . . . 227
A.3 application.properties, AmazonS3 plugin 3.8.1 . . . 227
A.3.1 New options or values . . . 227
A.4 application.properties, AWSHealthImaging plugin 1.3.11 . . . 227
A.4.1 New options or values . . . 227
A.5 application.properties, Dicomweb plugin 6.16.1 . . . 227
A.5.1 New options or values . . . 227
A.6 application.properties, FileSystem plugin 7.3.0 . . . 228
A.7 system.json . . . 228
A.7.1 New options or values . . . 228
A.8 rebranding_configuration.json . . . 229
A.8.1 Options that are handled differently . . . 229
A.9 medcadPlugins/links.json . . . 229
A.9.1 Removed options or values . . . 229

B Appendix: Upgrading 8.8.1 to 8.9.1 (Linux) **230**

C Appendix: Upgrading 8.8.1 to 8.9.1 (Windows) **232**

1. Introduction

MedDream Viewer functionality (called “MedDream DICOM Viewer” later in this Manual) is a HTML-based package for a PACS server that is designed to aid professionals in every day decision-making process, connecting all the medical data into a unified and fast-performing network. MedDream ensures a fast and reliable way to search, present, and analyze the medical data (images and video files) on various devices: computers, smartphones, tablets, and so forth.

MedDream covers: radiology, cardiology, oncology, gastroenterology and many other fields of medical application. It seamlessly integrates with various medical imaging devices, such as: ultrasound (US), magnetic resonance (MRI), positron emission tomography (PET), computed tomography (CT), endoscopy (ES), mammography (MG), digital radiography (DR), computed radiography (CR), ophthalmology, and so forth.

Core MedDream DICOM Viewer uses are:

- Replacement of hard copies, e.g., film archives, paper documents, etc.
- Remote access. MedDream provides the possibility to be mobile and work from any place in the world where the Internet is accessible. More than one person can access and view medical records at one time. Such functionality speeds up the collaboration among the professionals. So that a doctor in the hospital and a doctor who is in a different location may view the medical data and discuss it simultaneously. The patient’s medical history, various studies, and images are found much faster compared to the conventional paper-based methods.
- MedDream can be used as a standalone WEB Viewer or integrated into MedDream PACS, PacsOne PACS, dcm4chee Archive, Conquest, ClearCanvas, and Orthanc PACS systems. Moreover, MedDream can be adapted to the client’s PACS system and easily integrated into RIS/HIS workflow.
- MedDream has multiple functions such as searching of studies, viewing, analyzing, saving, exporting, forwarding images and videos, etc.

MedDream is dedicated only to show ePHI (Electronic protected health information) contained in the study file. Administrators, doctors, and patients decide how to use ePHI.

Note: The manufacturer, company “MedDream UAB”, does not keep Electronic Protected Health Information (ePHI).

1.1. Installation prerequisites

Make sure the server conforms to the Server-side requirements in the next chapter.

Most installation tasks will require administrative rights on the server (for example, setup of a system service or adjustment of file system permissions).

Before integrating with a particular system, read the corresponding part of the 3 Integration with PACS chapter first. That way, you’ll know in advance what integration-specific data or cooperation is needed from administrators of that system. For example, direct integrations will require suitable database credentials, while Query/Retrieve even requires configuring the PACS accordingly.

Warning: If you are updating to a newer version of MedDream or are applying a “patch” / “hotfix”, please make a backup of the existing installation. Under Linux, this also includes the related system services – usually not in the MedDream directory but elsewhere (depends on distribution).

Warning: When copying large multi-line examples of configuration or shell commands from this document, pay attention to the typesetting symbols ␣ and ↪, like here:

```
java -XX:MaxRAMPercentage=50.0 -DANTLR_USE_DIRECT_CLASS_LOADING=true -cp␣
↪MedDream-8.9.1.jar org.springframework.boot.loader.launch.PropertiesLauncher
```

After copying you need to 1) manually join the lines and remove characters that represent ␣ and ↪, 2) add a space where ␣ was present.

Unfortunately, broken lines with line continuation characters \ are not supported by all target environments; for example, they are OK in Linux shells but not in Windows Command Prompt or PowerShell.

2. Requirements

2.1. Server-side requirements

2.1.1. Minimal hardware requirements

Concurrent connections	CPU cores*	RAM**
1 connection	4 cores	8 GB
2 connections	4 cores	8 GB
5 connections	4 cores	8 GB
10 connections	4 cores	12 GB
20 connections	6 cores	16 GB
30 connections	8 cores	32 GB
60 connections	16 cores	32 GB
60+ connections	+1 core per 10 connections	+1 GB per 5 connections

* Equivalent of 5th generation Intel i5 2GHz CPU core (Q1'15 or later).

** Additional RAM must be reserved for OS, Database, PACS, and/or other services if installed in the same machine.

The storage is used for software (fixed 1 GB size) and temporary data cache. Performance of the caching storage directly affects the image open speed; therefore, SSD, RAM or high-speed disk storage is recommended for it.

Note: It is recommended to allocate the fastest disk storage, with **free space for 10 days' worth of new studies**, for the MedDream cache in order to have a better viewing performance.

The network bandwidth will directly affect image open speed. At least 100 Mbit/s is recommended.

Note: Minimal hardware requirements depend on the number of concurrent users, workload, and image types. It is recommended to allocate 20-80% more resources for unusual workload or specific data types.

2.1.2. Software requirements

For security-related configuration, see 7.8 Summary: Minimum relevant IT security requirements.

2.1.2.1. Recommended operating systems

We recommend the following operating systems on the server side:

- Windows Server 2019, Windows Server 2022, Windows Server 2025
- Windows 10, Windows 11 (64-bit only) and above
- Linux: Debian 13, Ubuntu 26.04 LTS, Fedora 44. The minimum glibc version is 2.17 for `x86_64` and 2.27 for `aarch64`.

2.1.2.2. Java

MedDream is a Java-based application and requires a JRE. (*It will run under a JDK, too.*) Currently supported Java runtime is Java 17 LTS. We recommend builds from adoptium.net.

The build must be 64-bit, as under 32-bit builds, most kinds of DICOM images will not be displayed.

2.1.2.3. VTK

Under Linux, the MPR Export functionality requires VTK as a system package. (Under Windows, bundled binaries are used instead).

Distribution	Package name
Debian/Ubuntu	`libvtk9-java`
Fedora	`vtk-java`
openSUSE	`vtk-java`
RHEL/CentOS	`vtk-java` (via EPEL)
Arch Linux	`vtk` (or `vtk-git` via AUR)
Alpine Linux	*package not available*

You can omit the package if MPR Export won't ever be used.

2.1.2.4. FFmpeg

A custom 64-bit limited-functionality build of FFmpeg for Windows and Linux is included in the installation archive and used automatically on these platforms.

A different path can be specified during installation (or afterwards) in order to use some other build.

2.1.3. Time synchronization

The system time needs to be reasonably accurate – for example, within one minute.

MedDream is designed as a single unprivileged application and therefore does not include means of updating the system time. The time should be synchronized by the operating system instead. **It is a responsibility of the system administrator** to set up a network time client and monitor its behavior.

A correct system date is important for adequate behavior of MedDream (for example, the license might contain date-based limits) and other parts of the infrastructure. Some network time clients even refuse to work when the date is off by too large an amount.

2.1.4. Licensing (Linux)

Under Linux, the most popular form of the MedDream license is bound to hostname and the MAC addresses of network adapters. The hardware (or virtualization) solution must **ensure that they are constant**. Using this license in a Docker container might be impossible unless the MAC is fixed via the `--mac-address` option of `docker run`.

Normally, Dockerized installations, especially cloud-based, imply that an Internet connection is available. For those, we offer a different kind of license that periodically queries our server (lic.meddream.com).

2.2. Client-side requirements

.	Desktop Web	Mobile iOS Web	Mobile Android Web
CPU	Intel i3 4 core CPU or better	—	—
RAM*	8+ GB of RAM, 256+ MB of video memory	2+ GB	2+ GB
Storage space	10+ GB	2+ GB	2+ GB
Network bandwidth**	100+ Mbit/s	100+ Mbit/s	100+ Mbit/s
Web Browsers	Chrome 148+, Firefox 151+, Safari 26+, Microsoft Edge 148+	Safari 26+, Chrome 148+	Chrome 148+, Firefox 151+

* for CT, MRI, PET-CT client-side MPR/MIP rendering:

- 64-bit CPU and 64-bit operating system;
- graphic board with >=1 GB video memory (hardware acceleration enabled in the browser);
- 12 GB of RAM to open more than 800 images;
- 16 GB of RAM to open more than 1500 images;
- 24 GB of RAM to open more than 3000 images (cardiac or functional imaging, MG Tomosynthesis).

* for MG Mammography:

- 64-bit CPU and 64-bit operating system;
- graphic board with >=1 GB video memory (hardware acceleration enabled in the browser);
- 16 GB of RAM.

* for MG Tomosynthesis:

- 64-bit CPU and 64-bit operating system;
- graphic board with >=1 GB video memory (hardware acceleration enabled in the browser);
- 16 GB of RAM.

* Depending on the total amount of RAM on the user's workstation, the browser is allowed to allocate the restricted amount of memory, and therefore, the MedDream application may load not more than 16 GB of data. Due to these limitations, the browser may run out of memory if the user loads several large computed tomography, mammography, or tomosynthesis studies.

** Network bandwidth will directly affect image open speed.

Note: JavaScript and cookies must be enabled for the corresponding MedDream backend address.

Note: Hardware acceleration should be enabled in the web browser for better performance.

For optimal performance and compatibility, browsers should be kept up to date.

Note: The Mozilla Firefox browser has certain memory limitations: MIST MPR functionality will fail on series where uncompressed DICOM data is larger than 2 GB.

Note: MedDream enables users to view all data stored with a bit depth of up to 16 bits per color channel. However, it displays 8 bits per color channel at a time due to web browser limitations (maximum display of 256 values per color channel). You can utilize the Window leveling tool to explore the entire data range within the images.

Note: **Server-side** requirements are detailed in another chapter earlier in this document.

2.3. Telemetry and statistical data collection

1. The Software may automatically collect and transmit certain non-personal, statistical, and technical information ("Telemetry Data") to MedDream servers at regular intervals or upon specific events.
2. Telemetry Data may include, but is not limited to, information such as Software version, license identification, module activation status, configuration parameters, usage frequency, performance metrics, error logs, and anonymized interaction data (e.g., clicks, navigation, or general usage patterns).
3. The purpose of such data collection is to verify licensing compliance, ensure Software security and integrity, improve performance and stability, and enhance overall usability and user experience (UI/UX).
4. No patient, medical image, personal, or identifiable user data is collected, transmitted, or processed. All Telemetry Data is aggregated and anonymized before any analysis or storage by MedDream.
5. Opt-out: Collection of UI/UX telemetry data (related to anonymized user interface interactions) may be disabled at any time by contacting support@meddream.com for instructions. However, collection of essential licensing, security, and performance-related telemetry data cannot be disabled, as it is required to ensure Software integrity and compliance.
6. By installing or using the Software, the user acknowledges and agrees to the collection and processing of Telemetry Data as described above, in accordance with MedDream's Privacy Policy.

2.4. Decommissioning of MedDream

Note: A particular version of MedDream is supported for a limited time (see the Info window or User Manual). When support is no longer provided, we strongly recommend updating to a supported version. Be aware that the customer is responsible for the outcome (including, but not limited to, personal data protection) from operating an unsupported version. Unsupported software versions are used at the customer's own risk.

3. Integration with PACS

Support for various PACSes is implemented in the form of plugins. Their .jar files are expected in the same directory, sys/plugins.

They have no version numbers in the name in order to prevent hard-to-troubleshoot situations where multiple versions of the same plugin are present. Currently, if there is a need to know the version number of the plugin, then one can open the JAR file as a ZIP file and look for the file META-INF/MANIFEST.MF; it will contain, among other things, the `Bundle-Version` property.

```
Bundle-Version: 2.0.10.SNAPSHOT
```

Warning: Do not mix plugins from different MedDream versions. There is no backward compatibility.

Ideally, the entire backend will crash immediately (during start) if there is even one incompatible plugin left in the sys/plugins directory. Note that plugins are loaded even if not referenced in application.properties: that configuration only allows MedDream to use them afterwards. In the worst case, some unpredictable behavior will happen later during use.

3.1. Integration modes

MedDream can access studies from PACS using:

- Direct access to PACS database (PacsOne, DCM4CHEE, ClearCanvas, Conquest, etc.).
- Web-based and other APIs (Orthanc, DICOMweb).
- DICOM 3.0 native interface (Query/Retrieve).
- Direct file system access where a PACS is not necessary. (However, for users' convenience, some third-party user interface will still need to pass file/directory names to MedDream.)

Commercial integration solutions are also possible on request. In that case, documentation with confidential information will be available as separate versions of this Manual.

For best performance, it is recommended to use direct access to the PACS database whenever available.

3.1.1. Direct access mode on a different host

Warning: In the Direct access to PACS database mode, MedDream must either:

- be installed on the same host as PACS, or
- have access to studies' files, and likely path remapping needs to be configured.

Some databases contain paths to DICOM files stored on the PACS host (not on some common network storage, etc.) and it is not always possible to create identical mount points or network drive letters on the MedDream host.

For this case, most PACS plugins support the setting `mappedStorageLocation` (or with a similar name) that specifies substrings in the path to be replaced with different substrings.

An example with the PacsOne plugin: `...].mappedStorageLocation=/volume1/DICOMNL\=O:\|/volume2/DICOMNL\=P:\`. Here `/volume1/DICOMNL\` is replaced with `O:\` under which the network resource shared by a Linux system is mounted on a Windows system; in the same fashion, `/volume2/DICOMNL\` is replaced with `P:\`. For best results, please take a look at Java logs for original paths.

Due to limitations of Windows, a service might fail to access a network drive mapped in an interactive session (even under the same user, even with the "Enable to interact with desktop" option turned on). In this case, avoiding a drive letter should give better results – just remap PACS-local directories to CIFS paths that would otherwise be attached to a drive letter (`...].mappedStorageLocation=/volume1/DICOMNL\=\\server\share\DICOMNL/`).

Note: In case of some transient database connection problems, PacsOne saves the received file not under the configured short-term directory but under `/var` (Linux) or `C:\` (Windows). This results in paths like `C:\2022-04-06-WED\...` that will not be accessible from another machine by default. As sharing the entire directory, even with restrictive file system permissions, is not desired due to security considerations, you should be able to work around this by sharing and remapping particular per-day directories: `C:\2022-04-06-WED` –> `\\server\anotherShare\2022-04-06-WED` etc. This is a recommended temporary solution when doctors need a particular study up quickly.

A script for easier moving of these files to the intended location is not available yet. To avoid this situation until a new PacsOne version fixes its behavior, make sure PacsOne is also restarted after restarting MySQL, especially when it's on a different host; in particular, always stop the PacsOne service during any MySQL maintenance.

3.1.2. Saving measurements, Key Objects, segmentations and screenshots

To save Key Objects, Presentation States (measurements), segmentations and secondary capture screenshots, the following parameters must be added to the configuration of every PACS integration plugin:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storeScpIp=127.0.0.1`

  IP address (or hostname) of the Storage Server.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storeScpPort=104`

  Storage Server Port.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storeScpAet=PACSONE`

  Storage Server AE Title.

Despite being declared at a plugin level (this reduces the probability of some human errors), they are used by a common Storage SCU.

The local AE Title is configured globally by `com.softneta.meddream.dcmsnd.bind` as before (the default value is `MEDDREAM`). The PACS must be configured to accept C-STORE sessions from this AE Title.

Since 7.9, every plugin can have a distinct local AE title that will be used when uploading annotations on behalf of this plugin:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storeScuAet=UPLOADER`

If this parameter is commented out or empty, then the global value is used.

Note: All four parameters mentioned above are needed for the Query/Retrieve plugin, too, unless it also has storageApiEnabled=true; in that case, the plugin's own parameters remoteAET, remoteHost, remotePort and localAET are used instead – it then uploads annotations over its own C-STORE client.

Similarly, the Dicomweb plugin with storageApiEnabled=true doesn't need the storeScp* parameters because then it uploads annotations over its own STOW-RS client.

3.1.3. Multiple PACSes in parallel

With most plugins, multiple PACSes can be defined simultaneously and then used in the Search window to open studies (also called "Multi-PACS setup").

Warning: Opening the same study from different PACSes *at the same time* is not fully supported. In particular, the Viewer window doesn't indicate from which PACS a particular study is opened. This will lead to confusion, especially in the case of different versions of the study (for example, an additional series from a workstation is present on one PACS and not in another).

It is also possible to combine two or more plugins ("split configuration") for a particular storage: one plugin for Search results and study structure, a different one for retrieving files, etc. Look for imageApiEnabled=false or searchApiEnabled=false in the documentation of every plugin (some plugins can't be combined with others, though).

In a similar fashion (storageApiEnabled=true), one can control which plugin in the combination will upload annotations. However, as of 8.9 there are only three plugins (QR, Dicomweb, and partly AmazonS3), that support storageApiEnabled=true.

These APIs are almost always mutually exclusive. The only exception since 8.8 is eventApiEnabled=true, which is allowed in every plugin of a storage. This is because some plugins are caching data in their Search API: when a plugin like that is combined with another's Image API, then both caches will need cleaning (for example, by the "Clear caches" functionality).

Note: Previously we were suggesting to use a common cache directory, and eventApiEnabled=false for a plugin that has imageApiEnabled=false, so that the "smarter" plugin does the cleaning. This is no more needed. However, if you decide to remove "eventApiEnabled=false" that was causing a conflict earlier, then you should also **configure separate cache directories (instead of a common one)** for a more straightforward and efficient behavior. Particularly, 8.8+ cleans structure-related plugin caches after uploading an annotation, however some plugins implement this by blindly removing an entire study-level subdirectory, which would also contain the DICOM files in case of a common cache directory.

Since 8.4 it became possible to use a plugin *of the same type* when combining the Search API and Image API set up for different PACSes. For example, two partial instances of the QR plugin can implement different Called AE Titles for C-FIND and C-MOVE SCPs on the same server (not possible otherwise as the "remoteAET" setting is common to all SCUs in the plugin).

Below is an example that accesses the same DCM4CHEE 5.x server via three plugins: direct integration for the Search window and study metadata, QR for fetching files, and Dicomweb for uploading annotations. Note how [0] doesn't need eventApiEnabled=false because that's the default in this case ("true" as default can only be due to settings not present here).

```
com.softneta.meddream.pacs.configurations[0].type=Dcm4chee5
com.softneta.meddream.pacs.configurations[0].id=threePlugins
com.softneta.meddream.pacs.configurations[0].imageApiEnabled=false
com.softneta.meddream.pacs.configurations[0].url=jdbc:mariadb://172.16.32.12:3306\
/dcm4chee5latest
com.softneta.meddream.pacs.configurations[0].pacsVersion=5.25
com.softneta.meddream.pacs.configurations[0].username=pacs5latest
com.softneta.meddream.pacs.configurations[0].password=pacs5latest
com.softneta.meddream.pacs.configurations[0].storage=fs1=\\\\172.16.32.12\\dicom1;\
fs2=\\\\172.16.32.12\\dicom2\\PACS-5-latest

com.softneta.meddream.pacs.configurations[1].type=QR
com.softneta.meddream.pacs.configurations[1].searchApiEnabled=false
com.softneta.meddream.pacs.configurations[1].id=threePlugins
com.softneta.meddream.pacs.configurations[1].remoteAET=DCM4CHEE
```

(continues on next page)

(continued from previous page)

```
com.softneta.meddream.pacs.configurations[1].remoteHost=172.16.32.12
com.softneta.meddream.pacs.configurations[1].remotePort=11112
com.softneta.meddream.pacs.configurations[1].localAET=MEDDREAM
com.softneta.meddream.pacs.configurations[1].cacheDir=f:/recv/DCM4CHEE
com.softneta.meddream.pacs.configurations[1].cmove.returnErrorMessages=true

com.softneta.meddream.pacs.configurations[2].type=Dicomweb
com.softneta.meddream.pacs.configurations[2].id=threePlugins
com.softneta.meddream.pacs.configurations[2].searchApiEnabled=false
com.softneta.meddream.pacs.configurations[2].imageApiEnabled=false
com.softneta.meddream.pacs.configurations[2].storageApiEnabled=true
com.softneta.meddream.pacs.configurations[2].stowRsUrl=http://172.16.32.12:8080/\
dcm4chee-arc/aets/DCM4CHEE/rs
```

3.2. Direct access to PACS database mode

As of May 2025, the plugins that are treated as direct integration, are “FileSystem”, “PacsOne”, “Dcm4chee2”, “Dcm4chee5”, “Conquest”, “ClearCanvas”.

This mode contains certain optimizations based on assumption that DICOM files are immediately available on the file system:

- one can use trackDicomModifications=true to automatically refresh the preparation cache if the file timestamps change on the PACS file system;
- when converting video files, there is no need to keep a copy of the PixelData attribute in the preparation cache (disk space economy);
- when using the local C-STORE SCP to trigger preparation due to prepareReceivedFile=true, the DICOM file is removed as soon as preparation finishes because it has no other uses. (In contrast, the QR plugin can reuse the file further and avoid downloading it once more; in a similar fashion some other plugins can alternatively download an updated file over Q/R, and will therefore expect it to stay.) You can disable that with . . . clearPrepareReceivedFile=false among plugin’s settings:

  ```
  com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].
  ↪clearPrepareReceivedFile=false
  ```

On the other hand, there are specific optimizations for non-direct integration plugins:

- the Preload Series and Preload Study functions will force download of all DICOM objects in the study, parallelizing that over a separate thread pool configured by settings downloadDicomCoreThreadCount, downloadDicomMaxThreadCount, downloadDicomQueueCapacity.

3.2.1. PacsOne

In the case of the PacsOne Server (https://pacsone.net/), MedDream implements direct database access mode.

A similar configuration called “Standard DB integration” uses the same plugin and a minimalistic database structure unrelated to the real PacsOne Server; the database is to be filled by a custom solution implemented by the customer. Please contact support@meddream.com for example scripts, etc.

Note: “Standard DB integration” doesn’t support saving of annotations and screenshots by itself; the whole solution is read-only with respect to the database and file system. The customer needs to implement their own solution to receive the objects over DICOM C-STORE and update the database/files accordingly. The DICOM C-STORE is the main upload mechanism available at the moment, although the Dicomweb plugin can also be utilized just for uploading via its STOW-RS client.

Since 8.3.1, MedDream supports downloading of DICOM files that have been stored on AWS S3, including PacsOne's long-term storage. Gzip encoding applied by the server is transparently removed.

3.2.1.1. PacsOne notes

Note: applet.php for MedDream is not included in the installation archive anymore. **To obtain it, please contact support@meddream.com.**

applet.php is regularly tested with PHP versions from 5.4 to 7.1, though newer ones might also work. It requires the php_curl PHP module.

Warning: PacsOne and mD Java Core processes must run as the same user because newer versions of PacsOne create subdirectories with permissions too strict for different users. Please ensure that on Linux operating systems, PacsOne and mD Core services use the same user or belong to the same group.

Warning: PacsOne must store the received DICOM files in the "DicomPart10" format. The other format, "Native", is not fully supported by MedDream (and you might have problems with other DICOM-related software). This setting is chosen during installation and later can be verified as follows:

- Windows: in the registry, REG_SZ `StorageFormat` under `HKEY_LOCAL_MACHINE\SOFTWARE\RainbowFish Software\PacsOne\${AeTitle}`;
- Linux: the setting `StorageFormat` in file(s) *.ini near PacsOne.exe.

PacsOne doesn't have an index on series.modality. If searches with a particular modality are unacceptably slow compared to the "All" choice, then you'll need to use the following MySQL command:

```
ALTER TABLE series ADD INDEX (modality);
```

Until 6.0, the HIS integration by Patient ID was using a certain kind of fuzzy matching that included coerced values of Patient ID. For example, /?patient=12345 will also list studies with Patient ID "12345[some_original_value]". Since 6.0, this is turned off by default. If you still need the legacy behavior, then configure the "PacsOne" plugin with `strictSearchIsEnabled=false`. This also changes the Patient History window.

Since MySQL 5.7, the ONLY_FULL_GROUP_BY mode is on by default. Consequently, MedDream's Search function uses a compatible and much slower query by default. If performance is paramount, we advise disabling ONLY_FULL_GROUP_BY and configuring the "PacsOne" plugin with `useModalityAggregation=false`.

The new Reporting function is incompatible with PacsOne's "studynotes" and "attachment" tables. It is no more possible to use PacsOne for accessing reports created in MedDream and vice versa.

3.2.1.2. MedDream configuration for PacsOne

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. If you intend to reuse PacsOne user management, then configure this kind of authentication in the `application.properties` file:

```
spring.profiles.include=auth-pacsone
authorization.defaultLoginPermissions=SEARCH,PATIENT_HISTORY,\
    UPLOAD_DICOM_LIBRARY
#authorization.users[0].userName=user1
#authorization.users[0].role=DOCUMENT_VIEW,BOUNDING_BOX_EDIT
```

(continues on next page)

(continued from previous page)

```
#authorization.users[0].accessibleStorages=st1,st2
authorization.remapRoles.Doctors=SMART_DRAW_EDIT
authentication.database.databaseName=DB_NAME
authentication.database.jdbcUrl=jdbc:mariadb://DB_HOST:DB_PORT/DB_NAME
```

(for example, `jdbc:mariadb://127.0.0.1:3306/${authentication.database.databaseName}` – we encourage placeholders to minimize human errors).

The above example of defaultLoginPermissions setting globally covers permissions not available in PacsOne user management. Also, please remember that PacsOne's "Export" privilege is mapped to both EXPORT_ISO and EXPORT_ARCH. The users[] setting can apply permissions to individual users.

In 8.4+, you can also use `authorization.remapRoles.*` to assign additional permissions to a particular group created in the PacsOne GUI ("Doctors" in this example). User's membership in a group like this is supported only for remapRoles; **MedDream will ignore PacsOne privileges assigned to the group; it still reads only privileges of that user**.

Every user will automatically have either REPORT_VIEW or REPORT_UPLOAD permission, depending on the PacsOne "upload" privilege. It is not possible to have users without access to Reporting.

Users with the PacsOne "System Administration" privilege will automatically have the ADMIN permission in MedDream. This interferes with per-user settings because USER_SETTINGS permission has a lower priority when deciding which settings file is to be used.

A different authentication method can still be configured according to 5.1.6 Built-in authentication and authorization.

Note: By default, application.SAMPLE.properties enables `auth-inmemory`, and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counterintuitive. Make sure to comment the unneeded line out.

---

3. Edit the `application.properties` file and update the options related to the PacsOne plugin. The documentation below is for plugin version 6.8.0.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=PacsOne`

     Use this specific value of `type` when connecting to PacsOne.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=PacsOneRouter`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple PacsOne instances by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in the report database and preparation cache.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].forwardApiEnabled=false`

(optional) Specifies if PacsOne-specific Forward functionality in the plugin is to be used.

Default is `false`: MedDream itself reads DICOM files via the plugin (the same way as when displaying images) and sends them to the recipient, using the same C-STORE client as for sending new PR/KO objects.

With `true`, MedDream only creates a forwarding job that is handled by the PacsOne server; as a result, studies are sent by the PACS service (often on a separate machine). **A single plugin must be configured.** A related setting, `com.softneta.meddream.dcmsnd.forwardingMethod=plugin`, is required. Database tables "dbjob" and "journal" must be writable by the MySQL user identified by `username` and `password` below. 8.9.0+ is able to create series- and instance-level forwarding jobs, too.

In both cases the list of forward destinations is taken from system.json; fetching the destinations from the PacsOne database is not supported.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=true`

  (optional) Specifies if this plugin/configuration pair implements "Clear cache" functionality; this might be desired when images are hosted on Amazon S3 and therefore need to be cached locally. Default is `true` if `dicomCacheDirectory` is not empty, and vice versa.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:mariadb://127.0.0.1:3306/PACS_DB_NAME`

  JDBC connection string. Please note that Oracle is not supported (PacsOne Server for MySQL is the only supported version).

  Note: SSL requirements can be disabled by adding the following arguments:

  - **verifyServerCertificate=false** Disables server certificate verification.
  - **useSSL=false** Disables SSL usage.
  - **requireSSL=false** Disables the SSL requirement.

  The plugin also supports PostgreSQL (`jdbc:postgresql://127.0.0.1:5432/DB_NAME`), Microsoft SQL Server (`jdbc:sqlserver://127.0.0.1:1433;database=DB_NAME`) and IBM DB2 (`jdbc:db2://<HOST>:<PORT>/<DATABASE_NAME>:defaultSchema=<SCHEMA>;`) for the "Standard DB integration" cases.

  `useSSL=false` in the MariaDB connector might also need `allowPublicKeyRetrieval=true`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserName`

  Username for connecting to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

  Password for connecting to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].mappedStorageLocation=D:/DICOM=Y:\\DICOM1|E:/DICOM=Y:\\DICOM2`

  (optional) Defines replacement paths for MedDream installed on a different host than the PACS and accessing images through mapped network drives.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useModalityAggregation=true`

  (optional) Defines if modalities will be aggregated in database query results. Aggregation ensures that all modalities are visible in the Modality column but this could cause performance degradation compared to older PHP-based MedDream versions. If set to false, then SQL

mode must *not* contain `ONLY_FULL_GROUP_BY`. More information about SQL mode can be found at https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html.

`true` is incompatible with Microsoft SQL Server ("Standard DB integration") and will result in errors.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=`

  (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` forces wildcard matching everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window.

  `null` or unset is the default. Before 8.0.0 this meant exact matching for Patient History and HIS integration, but wildcard matching for the Search window. Starting from 8.0.0, HIS integration scenarios *by Patient ID* use a query `origid = <value> OR origid LIKE <value>[%`. It will match both exact Patient ID and its typical coerced version, for example, "PAT001" and "PAT001[SENDERAE-123456]". **An exact match now requires true.**

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictUserAetIsEnabled=true`

  (optional) If true, the user will see only studies from assigned AEs. In the PacsOne GUI, see "Assign Dicom Studies Received From This AE To Web Users" under the configuration of a particular AE. Note that this setting covers only users, not their groups. Value `true` is ignored under PostgreSQL.

  Additionally, `true` also requires that the PacsOne user doesn't have the "View private data" privilege or MedDream application.properties configures PACSONE_VIEW_ONLY_PUBLIC for that user.

  The user identity is taken from the Login page. The HIS integration scenario with a non-empty "user.id" in the token was not tested, while the unsafe URL integration won't benefit from this filter due to the same user identity (authentication.his.username).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictGroupAetIsEnabled=true`

  (optional) Similar to strictUserAetIsEnabled but on group level, that is, it includes **groups** where the current user belongs, selected under "Assign Dicom Studies Received From This AE To Web Users" in the PacsOne GUI. If enabled together with strictUserAetIsEnabled, then the resulting filters are combined with an OR operation. Unsupported under PostgreSQL so far, too. The same logic with "View private data" and PACSONE_VIEW_ONLY_PUBLIC applies.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictFilterByAETs=ORTHO423,ENT105`

  (optional) Comma-separated list of AETs whose studies the user is able to access (when the token specifies a study but not a user name, or a different authentication than auth-pacsone is used). Empty by default.

  If the token specifies a user name and strictUserAetIsEnabled/strictGroupAetIsEnabled are configured, then this parameter has no effect.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables support for monitoring study structure changes (see structureListenerConfig in system.json). Disabled by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].defaultStartDate=20181107`

  (optional) Default "from" date in `YYYYMMDD` format when empty in the user interface.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].defaultEndDate=20181108`

  (optional) Default "to" date in `YYYYMMDD` format when empty in the user interface.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomCacheDirectory=${com.softneta.meddream.tempDir}/dcm/amz1`

  (optional) A cache directory for files downloaded from AWS S3. Not configured by default, therefore subsequent access will be slower and will waste AWS traffic.

  A cached file is reused forever; there is no support for If-Modified-Since semantics. You'll need to set up a separate entry under `com.softneta.temp-cleaner.tempItems[]` (see *Temporary files cleaner*).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsS3Marker=aws_s3`

  (optional) A magic value that indicates an S3 path instead of a local filesystem path. The default value is `aws_s3`, likely hardcoded in PacsOne.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsS3Separator=:`

  (optional) The character that separates `awsS3Marker` from the remaining path. The default value is `:`, likely hardcoded in PacsOne.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsS3ConnectionInfoCacheTimeInSec=600`

  (optional) AWS S3 bucket and region identifiers extracted from the database will be reused for this many seconds (600 by default).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encoding=ISO-8859-1`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingTo=ISO-8859-8`

  (optional) Specify encoding of database strings when it's not UTF-8.

  - `encoding=ISO-8859-1`, `encodingTo=`:
    - search query is converted from UTF-8 to ISO-8859-1
    - search results are converted from ISO-8859-1 to UTF-8
  - `encoding=ISO-8859-1`, `encodingTo=ISO-8859-8`:
    - search results are converted from ISO-8859-1 to ISO-8859-8
    - search query is converted from ISO-8859-8 to ISO-8859-1
  - `encoding=`, `encodingTo=ISO-8859-8`:
    - search results are converted from UTF-8 to ISO-8859-8
    - search query is converted from ISO-8859-8 to UTF-8

  `encoding` and `encodingTo` are empty: UTF-8 is assumed everywhere (the outcome might still be influenced by JVM system encoding).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].connectionInitSql=SET NAMES latin1`

  (optional) A custom SQL statement that will be executed after a database connection is established. The default value is the same as above when "jdbc:mariadb:..." is found in the URL, and empty otherwise.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomPersonName=false`

(experimental) Since 8.0.0, plugins are expected to return the full DICOM-formatted patient name, like "last^first^middle^prefix^suffix=ideographic=phonetic", for proper reformatting in Core (also see `personNameConfiguration` in system.json). `false` turns this off and is normally not required.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].optimized=true`

  (experimental) Use different queries for better performance when the database has been optimized accordingly (see 3.2.1.4 Optimizing the database (MySQL/MariaDB)).

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for the chosen method (for example, PacsOne users in the case of `spring.profiles.include=auth-pacsone`).

3.2.1.3. Integration into PacsOne web interface

After you successfully log into MedDream and are able to access studies on PacsOne, the system can additionally be configured to open MedDream from the PacsOne web interface.

Warning: All users of PacsOne will have the same set of MedDream permissions, configurable in `USER_PRIVILEGES` (applet.php) or by `authorization.defaultHisPermissions` (application.properties). If some users must not have access to, say, Forward or Reporting, then this restriction is necessary for everyone.

1. Extract the directory "token-service" from the installation archive into a separate directory on the server (example result: /opt/token-service/token-service.xml). From now on it will be called the installation directory of the Token Service.
2. Make a backup of PacsOne's php/applet.php.
3. Copy and Replace MedDream's applet.php over the PacsOne version of the file.
4. Verify whether `MEDDREAM_URL`, and especially `MEDDREAM_STORAGE_ID`, at the beginning are correct. `SSO_VALIDATOR_URL` will need correction, too, if the Token Service is configured with a different port (server.port in its own application.properties).
   - If you added MedDream to a legacy PacsOne installation and Apache is proxying it as `/meddream`, then MEDDREAM_URL is good already.
   - If MedDream is running on a different port without proxying, then MEDDREAM_URL can include the full address, like `//localhost:8080/`.
   - If you are using MedDreamPACS, which offers MedDream at `/` and the PACS web interface at `/Pacs`, then just use `/` for MEDDREAM_URL.
5. Start the Token Service from a separate console window: change to its installation directory, then run `java -jar token-service.jar`. Wait until the message "Application ... started" appears.
6. In the PacsOne web interface, find some studies, tick at least one checkbox and click the "Show" button.

Note: Default values of `security.frameOptionsPolicy` and `security.contentSecurityPolicy` disallow embedding MedDream in an IFRAME, which is used by applet.php. The browser is refusing to display the embedded viewer, and its console shows an error.

The quickest initial solution is to disable both in MedDream's application.properties:

```
security.frameOptionsPolicy=NONE
```

Afterwards you can experiment with `security.contentSecurityPolicy=frame-ancestors 'self' <server IP address or hostname>`. The final value of this parameter should include other directives, for example, `frame-ancestors 'self' <server IP address or hostname>; default-src 'self'; style-src 'self'`

`'unsafe-inline'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; child-src 'self' blob:; connect-src 'self' *; img-src 'self' data:;  worker-src 'self' blob:;.`

The Token Service has its own application.properties, not relevant to this issue.

Note: The "Show" button will appear on patient, study, series and image levels; however, MedDream 7.5+ supports the study level only, while PacsOne doesn't allow hiding the button selectively.

For later convenience the token validator should run as a service. The installation directory already provides helper files token-service.debian, token-service.redhat, token-service.xml, token-service-jar-wrapper.*, token-service.exe, and a separate version of application.properties. Please follow the 5.2 Running as a service chapter and do not forget to use the corresponding file names.

Warning: It is recommended to remove or rename PacsOne's risdirect.php. Our applet.php is not designed for it, will not work in such a scenario, and this combination might cause a slight security risk.

3.2.1.4. Optimizing the database (MySQL/MariaDB)

The plugin's setting `optimized=true` depends on certain database optimizations and improves search performance.

- The list of modalities in the study is taken from a custom dedicated column, `study.modality_list`, instead of collecting distinct values from related series. This column is not present in the PacsOne database schema (and not in the Standard DB Integration schema, too); it must be added manually. We offer a couple of triggers that update this column automatically.
- If the user enters a search key that starts with "*", then the search will match anywhere in the text (slower), instead of only at the beginning of the text. Not all involved columns are indexed by default.

Update the schema:

```
ALTER TABLE `series`
ADD INDEX `modality` (`modality`)
USING BTREE;

ALTER TABLE `study` ADD `modality_list` VARCHAR(64);
ALTER TABLE `study`
ADD INDEX `private` (`private`),
ADD INDEX `readingphysician` (`readingphysician`),
ADD INDEX `requestingphysician` (`requestingphysician`),
ADD INDEX `description` (`description`),
ADD INDEX `studydate_time` (`studydate`, `studytime`),
ADD INDEX `sourceae` (`sourceae`),
ADD INDEX `modality_list` (`modality_list`)
USING BTREE;

ALTER TABLE `patient`
ADD INDEX `firstName` (`firstname`),
ADD INDEX `lastName` (`lastname`),
ADD INDEX `middleName` (`middlename`)
USING BTREE;
```

Fill the study.modality_list column for existing studies:

```
UPDATE study SET modality_list = (SELECT GROUP_CONCAT(DISTINCT modality SEPARATOR ' ') FROM␣
↪series WHERE series.studyuid = study.uuid);
```

Add the triggers that will do the same for future studies:

```
CREATE TRIGGER after_series_insert AFTER INSERT ON series FOR EACH ROW
UPDATE study SET modality_list = (SELECT GROUP_CONCAT(DISTINCT series.modality SEPARATOR '
↪') FROM series WHERE series.studyuid = NEW.studyuid) WHERE study.uuid = NEW.studyuid;

CREATE TRIGGER after_study_insert BEFORE INSERT ON study FOR EACH ROW
SET NEW.modality_list = (SELECT GROUP_CONCAT(DISTINCT series.modality SEPARATOR ' ') FROM␣
↪series WHERE series.studyuid = NEW.uuid);
```

3.2.2. DCM4CHEE

For DCM4CHEE (https://www.dcm4che.org/), MedDream implements the direct database access mode.

3.2.2.1. DCM4CHEE notes

Warning: Support for DCM4CHEE 4 was not ported from PHP due to very limited demand.

For DCM4CHEE 2, compatibility is regularly tested only with DCM4CHEE 2.4.17.

In this PACS, when patient name and other demographics are changed in its GUI, the corresponding DICOM files are not updated on disk (only their copies exported via various interfaces are). However, MedDream does not merge both sources of data; for example, Info Labels always fetch the attributes from the file and will therefore show outdated information. The setting com.softneta.meddream.newObjectsUseMetadataFromPacs can only compensate for a single aspect of the issue; however there are more. **If patient demographics are often corrected at the PACS instead of the modality, we recommend using Q/R or DICOMweb integration instead.** MedDream 8.1+ alleviates that to some extent by adding Q/R support to the DCM4CHEE plugin: if patient demographics were updated on the PACS, then corresponding DICOM files will be downloaded via a C-MOVE request.

Warning: A DCM4CHEE 5.31+ database allows different patient demographics assigned to the same person: N records of `patient_id` –> same record of `patient` table <– M records of `study`. Essentially, studies from other institutions, with possibly different demographics, still belong to the same person. **MedDream doesn't fully support this,** and expects a single Patient ID in any related APIs. The Dcm4chee5 PACS plugin orders such aliased IDs by patient_id.pk, then uses the oldest result. This at least allows Core to memorize the oldest Patient ID when a HIS integration session is created, and successfully validate all subsequent requests against it. In general, it may not be possible to see the alternative IDs and names in some scenarios.

3.2.2.2. MedDream configuration for DCM4CHEE 5.x

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to the DCM4CHEE 5.x plugin. The documentation below is for plugin version 6.4.0.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Dcm4chee5`

     Use this specific value of `type` when connecting to DCM4CHEE 5.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=MainArchive`

Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple DCM4CHEE instances by using different values of `id` and remaining options.

An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

  (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

  (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:mariadb://127.0.0.1:3306/PACS_DB_NAME`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:postgresql://127.0.0.1:5432/PACS_DB_NAME`

  JDBC connection string. Supported DBMSes are MariaDB, MySQL (via the MariaDB connector only) and PostgreSQL.

  In case of connection problems, it might be necessary to add some more parameters to this string: `useJDBCCompliantTimezoneShift=true` for a time zone shift, `serverTimezone=UTC` for a specific timezone, `useSSL=false` to disable SSL, etc. (The first parameter-value pair is preceded by "?", the remaining pairs – by "&".) See https://mariadb.com/docs/connectors/mariadb-connector-j/about-mariadb-connector-j for MariaDB, or https://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html for MySQL, or https://jdbc.postgresql.org/documentation/use/ for PostgreSQL.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserName`

  Username used to connect to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

  Password used to connect to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encoding=ISO-8859-1`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingTo=ISO-8859-8`

  (optional) Specify encoding of database strings when it's not UTF-8.

  - `encoding=ISO-8859-1`, `encodingTo=`:
    - search query is converted from UTF-8 to ISO-8859-1
    - search results are converted from ISO-8859-1 to UTF-8
  - `encoding=ISO-8859-1`, `encodingTo=ISO-8859-8`:
    - search results are converted from ISO-8859-1 to ISO-8859-8
    - search query is converted from ISO-8859-8 to ISO-8859-1
  - `encoding=`, `encodingTo=ISO-8859-8`:
    - search results are converted from UTF-8 to ISO-8859-8
    - search query is converted from ISO-8859-8 to UTF-8

`encoding` and `encodingTo` are empty: UTF-8 is assumed everywhere (the outcome might still be influenced by JVM system encoding).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].connectionInitSql=SET NAMES latin1`

  (optional) A custom SQL statement that will be executed after a database connection is established. Empty by default.

  The value in the example might be required for MariaDB as an alternative to the ... charset option.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].aggregateDataByDatabase=true`

  (optional) Different approaches to collect distinct series-level values of *Modality* and *Source AET* for every study in the search results. `false` might improve performance in some cases (was not tested extensively).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storage=fs1=c:\\mnt\\NAS1;fs2=c:\\mnt\\NAS2`

  Filesystem root directories, as configured in the LDAP tree via entries named "dcmStorageID". The plugin doesn't read this configuration from LDAP at the moment.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].pacsVersion=5.7`

  (optional) Two-component DCM4CHEE version number that defines what database fields are available. Namely, 5.7 adds the *Number of Frames* attribute; 5.23 stores a person name in 3 fields instead of 15 and renames series.src_aet to series.sending_aet; 5.31 relates the tables "patient" and "patient_id" in a different fashion. If the version of your PACS is not 5.7-5.22 (expected by default), then you'll need to adjust this parameter to avoid SQL errors.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomPersonName=false`

  (experimental) Since 8.0.0 plugins are expected to return the full DICOM-formatted patient name, like "last^first^middle^prefix^suffix=ideographic=phonetic", for proper reformatting in Core (also see `personNameConfiguration` in system.json.) `false` turns this off and is normally not required.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables support for monitoring study structure changes (see structureListenerConfig in system.json). Disabled by default.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for the chosen method.

Q/R support for updated DICOM files

The plugin can detect whether patient demographics are changed: it parses AttributeModificationDateTime in the "dicomattrs.attrs" database record referenced by the "patient" table and compares with a value cached on disk. If a newer value is found, then the cached one is updated, and a C-MOVE request downloads updated DICOM files to the local C-STORE SCP. Therefore, MedDream can read up-to-date files from its own cache. If patient demographics were not changed, then MedDream reads the original DICOM files by their paths in the database.

This mechanism is a compromise between direct integration (the fastest) and Q/R (which provides up-to-date DICOM files). The slower mode will be used only for studies with updated patient demographics.

Note: Make sure that MedDream, PACS and DBMS are on the same time zone, or else the changes might not be detected correctly.

The timestamps are compared only when the following options are configured:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].localAET=MEDDREAM`

  The AE Title of the C-MOVE client, and the destination of C-MOVE operations.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteAET=DCM4CHEE`

  The AE Title of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteHost=127.0.0.1`

  IP address or hostname of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remotePort=11112`

  Port number of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].patientUpdateCacheDir=${com.softneta.meddream.tempDir}/timestamps`

  Directory for caching timestamps of patient demographics records. As a combination of the plugin's `.id` and Study UID is hashed and distributed among subdirectories, you can use the same patientUpdateCacheDir for multiple instances of the plugin.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDir= ${com.softneta.dicomStoreService.saveDirectory}/DCM4CHEE/`

  A directory for DICOM files from `remoteAET` received by the local C-STORE SCP.

The remaining options are for slight adjustments of the behavior:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].retryDelay=1000`

  Period in milliseconds for checking whether the DICOM file has been received.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=10000`

  Maximum total duration in milliseconds for waiting for a received file.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].fetchStudyAfterStructure=false`

  Start the C-MOVE command (if needed according to timestamps) as early as possible: just after fetching the study structure, not on actual request to fetch a DICOM file.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionCache=false`

  When true, every session has its own subdirectory under `cacheDir` that is cleaned automatically when the session ends (cleaning also requires options `eventApiEnabled=true` and `com.softneta.preparation.useSessionCache=true`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionPatientUpdateCache=false`

  When true, every session has its own subdirectory under `patientUpdateCacheDir` that is cleaned automatically when the session ends (cleaning also requires options `eventApiEnabled=true` and `com.softneta.preparation.useSessionCache=true`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

Set this to true together with `useSessionCache` or `useSessionPatientUpdateCache` (and also `com.softneta.preparation.useSessionCache`) for automated cleaning of per-session caches.

Since 8.3.0, `true` is the default value when Q/R mode is configured (options localAET, remoteAET, remoteHost, remotePort, cacheDir, patientUpdateCacheDir are present). Furthermore, EventAPI also implements the Clear Cache functionality at the plugin side.

When set to false, the plugin will support neither Clear Cache nor automated cleaning of per-session caches. If explicitly set to true, however Q/R mode is not configured, then the plugin won't attempt any cleaning, too.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].clearPrepareReceivedFile=false`

  When this plugin is configured for the Q/R fallback, and the DICOM Listener is also used to create preparation jobs on reception of a DICOM file, then `false` is definitely needed here. Otherwise those received files are removed automatically by the Listener before they can be used by the plugin.

  It's a Core setting, the plugin doesn't know anything about it.

3.2.2.3. MedDream configuration for DCM4CHEE 2.x

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. **(Option 1)** Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. **(Option 2)** If you intend to (partially) reuse DCM4CHEE 2.x user management, then configure this kind of authentication in the `application.properties` file:

```
spring.profiles.include=auth-dcm4chee2
authentication.database.username=APP_USER
authentication.database.password=APP_PASS
authentication.database.jdbcUrl=jdbc:mariadb://DB_HOST:DB_PORT/DB_NAME
authorization.defaultLoginPermissions=SEARCH,DOCUMENT_VIEW
authorization.remapRoles.Doctor=REPORT_UPLOAD
authorization.remapRoles.WebAdmin=ADMIN
authorization.users[0].userName=user
authorization.users[0].role=UPLOAD_DICOM_LIBRARY,EXPORT_ARCH
```

   `.jdbcUrl` supports not only MariaDB but also PostgreSQL (`jdbc:postgresql://DB_HOST:DB_PORT/DB_NAME`) and MS SQL Server (`jdbc:sqlserver://DB_HOST:DB_PORT;database=DB_NAME`). `.username` and `.password` are credentials of an application user for reading the database tables; you can use the same credentials for the plugin below. *Oracle is not supported for auth-dcm4chee2; Oracle-based PACS integration will need to use a different authentication method.*

   `.defaultLoginPermissions` lists common permissions. In this example, the DCM4CHEE's predefined "Doctor" role additionally gets REPORT_UPLOAD, and the predefined user "user" gets UPLOAD_DICOM_LIBRARY and EXPORT_ARCH. Because "user" belongs to "Doctor" by default, it now has REPORT_UPLOAD, UPLOAD_DICOM_LIBRARY, EXPORT_ARCH, REPORT_UPLOAD and everything from defaultLoginPermissions. Similarly, the predefined user "admin" who has "WebAdmin" and "Doctor" roles in DCM4CHEE, now has ADMIN, SEARCH, etc. (except UPLOAD_DICOM_LIBRARY and EXPORT_ARCH).

   The full list of permissions can be found in 5.1.6 Built-in authentication and authorization. That chapter also describes a few universal authentication methods.

   Note: By default, application.SAMPLE.properties enables `auth-inmemory`, and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times,

and the outcome might be counterintuitive. Make sure to comment the unneeded line out.

Warning: auth-dcm4chee2 expects insecurely stored passwords. We don't recommend this authentication method, especially for new installations, and plan to remove it in the future. Please inform support@meddream.com if your business case still depends on DCM4CHEE 2.x user management.

4. Edit `application.properties` file and update the options related to DCM4CHEE 2.x plugin. The documentation below is for plugin version 5.3.1.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Dcm4chee2`

     Use this specific value of `type` when connecting to DCM4CHEE 2.x.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=RouterPacs`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple DCM4CHEE instances by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in the report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:mariadb://127.0.0.1:3306/PACS_DB_NAME`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:postgresql://127.0.0.1:5432/PACS_DB_NAME`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:sqlserver://127.0.0.1:1433;database=PACS_DB_NAME`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:oracle:thin://127.0.0.1:1521/xe`

     JDBC connection string. Supported DBMSes are MariaDB, MySQL (via the MariaDB connector only), PostgreSQL, Microsoft SQL Server and Oracle.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserName`

     Username used to connect to the database.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

     Password used to connect to the database.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].defaultStoragePath=C:/DCM4CHEE_INSTALL_DIR/server/default`

Base directory for relative paths. By default, DCM4CHEE stores the files relative to a certain location below the installation directory (see the example), unless one sets up in advance an absolute filesystem prefix via addRWFileSystem() in JMX Console.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].charset=latin1`

  (optional) Defines the character set used in the database. If not empty, conversion from this character set to UTF-8 is attempted; otherwise, the data is passed as is (default). `utf8` might work in case of double encoding.

  Note: The PACS GUI (/dcm4chee-web, /dcm4chee-web3, etc.) is able to correctly display data even if the database server was configured incorrectly before sending the images, and therefore parts of the database accessed by the plugin now contain garbled text or question marks (information is permanently lost already). *DCM4CHEE has a certain workaround against this situation; there is no equivalent of it in the plugin.*

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].connectionInitSql=SET NAMES latin1`

  (optional) A custom SQL statement that will be executed after a database connection is established. Empty by default.

  The value in the example might be required for MariaDB as an alternative to the ... charset option.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].enableStatistics=false`

  (optional) `true` enables query statistics such as query time and so on. You might also want to set `logging.level.org.hibernate=DEBUG` in application.properties to get more details.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].mappedStorageLocations=D:/DICOM=Y:\\DICOM1;E:/DICOM=Y:\\DICOM2`

  (optional) Defines replacement paths for MedDream that is installed on a different host than the PACS, and is accessing images through mapped network drives or network paths.

  We recommend configuring defaultStoragePath with a ready-to-use value, instead of using a PACS-local value and adding a separate replacement rule here. Of course, the corresponding folder must be shared on the PACS, too.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables support for monitoring study structure changes (see structureListenerConfig in system.json). Disabled by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].addPatientBirthDateToDescription=false`

  (optional) If true, Study Description is prefixed by Patient's Birth Date. This is for display only – searching by this date is not possible.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomPersonName=false`

  (experimental) Since 8.0.0 plugins are expected to return the full DICOM-formatted patient name, like "last^first^middle^prefix^suffix=ideographic=phonetic", for proper reformatting in Core (also see `personNameConfiguration` in system.json.) `false` turns this off and is normally not required.

5. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for the chosen method.

Q/R support for updated DICOM files

The plugin can detect whether patient demographics are changed: it compares the database field "patient.updated_time" with "patient.created_time" and a value cached on disk. If a newer value is found, then the cached one is updated, and a C-MOVE request downloads updated DICOM files to the local C-STORE SCP. Therefore, MedDream can read up-to-date files from its own cache. If patient demographics were not changed, then MedDream reads the original DICOM files by their paths in the database.

Note: Make sure that MedDream, PACS and DBMS are in the same time zone, or else the changes might not be detected correctly.

This mechanism is a compromise between direct integration (the fastest) and Q/R (which provides up-to-date DICOM files). The slower mode will be used only for studies with updated patient demographics.

The timestamps are compared only when the following options are configured:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].localAET=MEDDREAM`

  The AE Title of the C-MOVE client, and the destination of C-MOVE operations.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteAET=DCM4CHEE`

  The AE Title of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteHost=127.0.0.1`

  IP address or hostname of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remotePort=11112`

  Port number of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].patientUpdateCacheDir=${com.softneta.meddream.tempDir}/timestamps`

  Directory for caching timestamps of patient demographics records. As a combination of the plugin's `.id` and Study UID is hashed and distributed among subdirectories, you can use the same patientUpdateCacheDir for multiple instances of the plugin.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDir=${com.softneta.dicomStoreService.saveDirectory}/DCM4CHEE/`

  A directory for DICOM files from `remoteAET` received by the local C-STORE SCP.

The remaining options are for slight adjustments of the behavior:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].retryDelay=1000`

  Period in milliseconds for checking whether the DICOM file has been received.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=10000`

  Maximum total duration in milliseconds for waiting for a received file.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].fetchStudyAfterStructure=false`

  Start the C-MOVE command (if needed according to timestamps) as early as possible: just after fetching the study structure, not on the actual request to fetch a DICOM file.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionCache=false`

  When true, every session has its own subdirectory under `cacheDir` that is cleaned automatically when the session ends (cleaning also requires options `eventApiEnabled=true` and `com.softneta.preparation.useSessionCache=true`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionPatientUpdateCache=false`

  When true, every session has its own subdirectory under `patientUpdateCacheDir` that is cleaned automatically when the session ends (cleaning also requires options `eventApiEnabled=true` and `com.softneta.preparation.useSessionCache=true`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

  Set this to true together with `useSessionCache` or `useSessionPatientUpdateCache` (and also `com.softneta.preparation.useSessionCache`) for automated cleaning of per-session caches.

  Since 8.3.0, `true` is the default value when Q/R mode is configured (options localAET, remoteAET, remoteHost, remotePort, cacheDir and patientUpdateCacheDir are present). Furthermore, EventAPI also implements the Clear Cache functionality at the plugin side.

  When set to false, the plugin will support neither Clear Cache nor automated cleaning of per-session caches. If explicitly set to true, however, Q/R mode is not configured, then the plugin won't attempt any cleaning, too.

  In general, an explicit `false` is needed when imageApiEnabled=false.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].clearPrepareReceivedFile=false`

  When this plugin is configured for the Q/R fallback, and the DICOM Listener is also used to create preparation jobs on reception of a DICOM file, then `false` is definitely needed here. Otherwise those received files are removed automatically by the Listener before they can be used by the plugin.

  It's a Core setting, the plugin doesn't know anything about it.

3.2.3. ClearCanvas

For ClearCanvas (https://clearcanvas.github.io/), MedDream implements the direct database access mode.

3.2.3.1. ClearCanvas notes

Support was tested with ClearCanvas 2.0 and 13.2.

Warning: Because the commercial version of ClearCanvas became end-of-life in 2020, support is tested only occasionally. MedDream 8.0.0 was not tested.

As IIS occupies the port 80, MedDream will need to run on a different port (the `server.port` setting). If MedDream URL looks "not nice" because of that, then you can use IIS as a reverse proxy; please contact support@meddream.com for instructions.

This plugin doesn't support studyStructureCountersEnabled=true.

3.2.3.2. MedDream configuration for ClearCanvas

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to ClearCanvas plugin. The documentation below is for plugin version 5.2.0.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=ClearCanvas`

     Use this specific value of `type` when connecting to ClearCanvas.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=OldArchive`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple ClearCanvas instances by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:sqlserver://127.0.0.1:1433;database=ImageServer`

     URL should be specified in format: `jdbc:sqlserver://<HOST>\\<INSTANCE>:<PORT>;database=<DATABASE_NAME>`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserName`

     Username used to connect to the database.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

     Password used to connect to the database.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].mappedStorageLocation=C:\\FS=Z:\\Studies`

     (optional) Defines pair of path strings that should be replaced in order to use mapped storages. Syntax for defining strings: “oldPathPart=newPathPart”.

     In case of multiple rules, separate them by pipe character and enter more specific ones first: `E:\\DICOM2=\\\\CLRCANVAS-SRV\\DICOM2\|E:\\DICOM=Z:\\DICOM`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=`

     (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` forces wildcard matching everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window. `null` or unset is the default: exact matching for Patient History and HIS integration, wildcard matching for Search window.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1:MEDDREAM_PORT/`. Use login credentials that are valid for chosen method.

3.2.4. Conquest

For Conquest (https://www.natura-ingenium.nl/dicom.html) MedDream implements the direct database access mode.

3.2.4.1. Conquest notes

MedDream does not support the proprietary "V2 (allows NKI compression)" image format. All images that were received by Conquest with this setting on, including those of the example patient "HEAD EXP2", will be unusable.

This plugin doesn't support studyStructureCountersEnabled=true.

3.2.4.2. MedDream configuration for Conquest

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to Conquest plugin. The documentation below is for plugin version 5.1.23.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Conquest`

     Use this specific value of `type` when connecting to Conquest.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=ResearchPacs`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple Conquest instances by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:mariadb://127.0.0.1:3306/PACS_DB_NAME`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:postgresql://127.0.0.1:5432/PACS_DB_NAME`
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].url=jdbc:sqlserver://127.0.0.1:1433;database=PACS_DB_NAME`

JDBC connection string. Supported DBMSes are MariaDB, MySQL (via MariaDB connector only), PostgreSQL, Microsoft SQL Server, dBASE 3.

MS SQL server isn't directly supported by Conquest. This is a special case when Conquest uses an ODBC source backed by the SQL Server, and MedDream connects to the server directly.

Please contact support@meddream.com if you would like to use dBASE 3 integration.

In case of connection problems, it might be necessary to add some more parameters to this string, like `useSSL=false` to disable SSL. (The first parameter-value pair is preceded by "?", the remaining pairs – by "&".) See https://mariadb.com/docs/connectors/mariadb-connector-j/about-mariadb-connector-j for MariaDB, or https://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html for MySQL, or https://jdbc.postgresql.org/documentation/use/ for PostgreSQL, or https://learn.microsoft.com/en-us/sql/connect/jdbc/setting-the-connection-properties for Microsoft SQL Server.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserName`

  Username used to connect to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

  Password used to connect to the database.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].pathToDicomIni=E:\\Conquest\\dicom.ini`

  Full path to the Conquest configuration file dicom.ini. MedDream will read `MAGDeviceXXX=...` lines from it.

  There is no "mappedStorageLocation" configuration option like in other plugins. In case when MedDream is on a different host, one can copy dicom.ini to a file on MedDream host, leave only MAGDeviceX lines and update them with shared paths that are accessible on MedDream host. For example, `MAGDevice1=E:\DICOM` might change to `MAGDevice1=\\conquest-srv\dicomstorage1`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useModalityAggregation=true`

  (optional) Aggregate results in the DBMS (not in the plugin). Performance might become worse.

  `false` by default, and can't be set to true for MS SQL Server or dBASE 3.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encoding=ISO-8859-1`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingTo=ISO-8859-8`

  (optional) Specify encoding of database strings when it's not UTF-8.

  - `encoding=ISO-8859-1`, `encodingTo=`:
    - search query is converted from UTF-8 to ISO-8859-1
    - search results are converted from ISO-8859-1 to UTF-8
  - `encoding=ISO-8859-1`, `encodingTo=ISO-8859-8`:
    - search results are converted from ISO-8859-1 to ISO-8859-8
    - search query is converted from ISO-8859-8 to ISO-8859-1

  - `encoding=`, `encodingTo=ISO-8859-8`:
    * search results are converted from UTF-8 to ISO-8859-8
    * search query is converted from ISO-8859-8 to UTF-8

  If `encoding` and `encodingTo` are empty (default), then UTF-8 is assumed everywhere (the outcome might still be influenced by JVM system encoding).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingF=cp850`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingFTo=ISO-8859-13`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].encodingFTest=true`

  (optional) Like `encoding` and `encodingTo` but for file system paths. Conquest uses the Patient ID attribute in subdirectory names, and national characters in the attribute might make the files inaccessible.

  If empty (default), then no character set conversion is performed.

  `encodingFTest=true` attempts to guess the encoding until the file becomes accessible. You will need to look for hints in "Test Encoding" log messages (INFO level), then configure `encodingF` and `encodingFTo` accordingly.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].connectionInitSql=SET NAMES latin1`

  (optional) A custom SQL statement that will be executed after a database connection is established. Empty by default.

  The value in the example might be required for MariaDB as an alternative to the ...encoding and ...encodingTo options.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for chosen method.

3.3. Integration over Web or other APIs

3.3.1. Orthanc

For Orthanc (https://www.orthanc-server.com/) MedDream uses its custom Web API. DICOM files are also downloaded through it, and can be cached for better performance.

Direct database access wasn't implemented as it would likely offer little performance gain due to database schema complexity. You might try the DICOM mode or DICOMweb as alternatives.

3.3.1.1. Orthanc notes

The API cannot be used to authenticate MedDream users because tested versions of Orthanc offer only global credentials (for a single user). These credentials must still be entered in plugin configuration, though.

When using preparation triggered by forwarding of images to the DICOM Listener, you should use the `onStableStudy` event in Orthanc for forwarding. Immediate forwarding (before the import to Orthanc database has finished), like based on the `onStoreInstance` event, will result in problems as MedDream won't be able to get recent study metadata this early. The example Lua script can be obtained from support@meddream.com.

3.3.1.2. MedDream configuration for Orthanc

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to Orthanc plugin. The documentation below is for plugin version 5.2.6.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Orthanc`

     Use this specific value of `type` when connecting to Orthanc.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=Telemedicine`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple Orthanc instances by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

     (optional) Set this to `false` in order to disable support for Clear Cache. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].baseUrl=http://127.0.0.1:8042`

     Base URL that points to the root of Orthanc Web API. (Endpoints `/tools/find`, `/instances` etc. will be appended automatically.)
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserLogin`

     Username for Basic Authentication.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

     Password for Basic Authentication.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomCacheDirectory=${com.softneta.meddream.tempDir}/dcm/Telemedicine`

     (optional) A directory for caching images and study structure. Without this setting a file is downloaded anew every time and it's not possible to examine the contents for troubleshooting. Study structure is cached together with a "last study update" timestamp from Orthanc; this ensures that an updated structure is always fetched from the server (excess caching is prevented).

     This plugin supports URIs (starting with `file://`). Making them relative to tempDir or another setting is encouraged to minimize human errors.

     Multiple instances of this plugin should have different directories in order to not mix up potentially different versions of the file with the same Study/Series/SOP Instance UIDs.

As of MedDream 7.5.1+, this directory is not cleaned automatically and you need to configure a separate entry under `com.softneta.temp-cleaner.tempItems[]` (see *Temporary files cleaner*).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDicomFiles=true`

  (optional) If false, dicomCacheDirectory is used only for caching the study structure.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].pythonPlugin=false`

  (optional) `true` enables a faster method of fetching the study structure. However, the Python plugin and a custom script must be configured in Orthanc; see the next chapter.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=true`

  (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` forces wildcard matching everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window. `null` or unset is the default: exact matching for Patient History and HIS integration, wildcard matching for Search window.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchPoolSize=10`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imagePoolSize=10`

  (optional) Maximum number of threads for fetching search results or metadata, and downloading DICOM files, respectively. Increasing the value might improve performance, however it also increases probability of timeouts and then raising `searchRequestRepeat` / `imageRequestRepeat` is advised.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchRequestRepeat=15`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageRequestRepeat=5`

  (optional) In case of download failure the request will automatically be retried this number of times. Each retry is delayed by 300 ms, 600 ms, 900 ms and so on.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].nonStrictSearchPattern=00100020:*{value}*;00080050:*{value}*`

  (optional) Specifies where to put wildcard characters when implementing a non-strict search. The value in the example is used by default, and yields the same behavior as before.

  Entries are separated with `;`. Format of the entry is `<tag code>:<value pattern>`. The `<value pattern>` part must contain a placeholder `{value}` that will be replaced with the search key; furthermore, space characters in the key are replaced with `*`. Currently recognized values for `<tag code>` are `00100020` and `00080050`.

  This would match only values beginning with the search key: `00100020:{value}*;00080050:{value}*`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchSymbols=*,?`

  (optional) Comma-separated list of wildcard characters. They aren't allowed in strict search scenarios, and will yield an error message.

  The default value with two characters from the Standard is a security improvement: it makes sure that the PACS won't return studies to which the end user has no access. An empty string means pre-8.0 behavior.

In non-strict scenarios, including strictSearchIsEnabled=false or keys like Study Description, these characters are used to decide that the search key is good enough — namely, it doesn't need formatting by `nonStrictSearchPattern` and spaces aren't to be replaced with asterisks. When the end user is using the Search window, (s)he can explicitly add wildcards where deemed appropriate: for example, `PID001` is sent to the PACS as `*PID001*`, `PID 001` – `*PID*001*`, `PID001*` – `PID001*`, `PID?001` – `PID?001`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables support for monitoring study structure changes (see structureListenerConfig in system.json). Disabled by default.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for chosen method.

3.3.1.3. Orthanc Python plugin

The Python plugin in Orthanc allows scripting custom behavior. A script (given below) returns study metadata significantly faster than the dedicated method of Orthanc API.

Note: Orthanc 1.6.0 or newer is required.

Prepare the script file

Save the following into a file meddream.py:

```
import json
import orthanc
def GetStudyInfo(output, uri, **request):
   studyId = request['groups'][0]
   # Fetch all series and instances once
   series = json.loads(orthanc.RestApiGet(f'/studies/{studyId}/series'))
   instances = json.loads(orthanc.RestApiGet(f'/studies/{studyId}/instances?expand'))
   # Pre-index instances by series ID for fast lookup
   series_to_instances = {}
   for instance in instances:
       parent = instance['ParentSeries']
       if parent not in series_to_instances:
           series_to_instances[parent] = []
       series_to_instances[parent].append(instance)
   result = []
   for serie in series:
       tags = serie['MainDicomTags'].copy()
       tags['OrthancSeriesID'] = serie['ID']
       tags['ParentStudy'] = serie['ParentStudy']
       series_instances_info = []
       # Get instances that belong to this series
       for inst in series_to_instances.get(serie['ID'], []):
           inst_tags = inst['MainDicomTags'].copy()
           inst_tags['OrthancInstanceID'] = inst['ID']
           # Fetch metadata once per instance
           metadata = json.loads(
               orthanc.RestApiGet(f'/instances/{inst["ID"]}/metadata?expand')
           )
           inst_tags.update(metadata)
```

(continues on next page)

(continued from previous page)

```
            series_instances_info.append(inst_tags)
        tags['Instances'] = series_instances_info
        result.append(tags)
    output.AnswerBuffer(json.dumps(result), 'application/json')

orthanc.RegisterRestCallback('/studies/(.*)/info', GetStudyInfo)
```

Install the plugin

Orthanc in Docker

The easiest and fastest way to have Orthanc with Python plugin is to use a Docker image that already includes the plugin: https://hub.docker.com/r/jodogne/orthanc-python.

For example, if you are installing MedDream with Orthanc and want to use the Python plugin, then use the following commands:

```
docker network create orthanc
docker run --restart=always --network=orthanc --name orthanc -v /mnt/orthanc-db/:\
    /var/lib/orthanc/db/ -itd -p 4242:4242 -p 8042:8042 jodogne/orthanc-python
docker run --restart=always --network=orthanc --name meddream -itd -p 80:8080 -e \
    integration=study meddream/orthanc-dicom-viewer:8.1.0
```

Orthanc under Windows

Install Python 3.7-3.8 on the Orthanc machine (for example, Python 3.8.8 64bit from https://www.python.org/downloads/windows/). Make sure to check *Add Python 3.X to PATH* in the installer.

Download the plugin depending on installed Python and Orthanc versions from https://www.orthanc-server.com/browse.php?path=/plugin-python. Pay special attention to 32 or 64bit, and what minimum Orthanc version is required by the chosen plugin version. At the time of this writing, plugin 3.1 didn't work properly with Orthanc 1.9.3.

Copy the DLL to the Orthanc "Plugins" directory, for example, C:\Program Files\Orthanc Server\Plugins\OrthancPython-Win64-Python3.8-1.0.dll. Then restart the Orthanc service.

Orthanc under Linux

You should install the `orthanc-python` package from your native Debian/Ubuntu distribution if available, or compile the plugin from sources.

Enable the plugin and install the script

Orthanc in Docker

Verify that Python plugin is activated by opening your plugins page, for example: http://localhost:8042/app/explorer.html#plugins.

Copy meddream.py to any suitable location in Docker container, for example:

```
docker cp meddream.py orthanc:/usr/local/share/orthanc/plugins/meddream.py
```

In the configuration file orthanc.json, verify path to the plugins directory and add the path to your script, for example:

```
"Plugins" : [
    "/usr/share/orthanc/plugins", "/usr/local/share/orthanc/plugins"
],
"PythonScript" : "/usr/local/share/orthanc/plugins/meddream.py",
```

It might be easier to copy orthanc.json from the container to your environment (`docker cp orthanc:/etc/orthanc/orthanc.json orthanc.json`), edit and then copy it back (`docker cp orthanc.json orthanc:/etc/orthanc/orthanc.json`).

Afterwards the container must be restarted (`docker restart orthanc`).

Orthanc under Windows

Verify that Python plugin is activated by opening your plugins page, for example: http://localhost:8042/app/explorer.html#plugins.

Copy meddream.py to any suitable location, for example, C:\Orthanc\meddream.py.

In the configuration file orthanc.json, verify path to the plugins directory and add the path to your script, for example:

```
"Plugins" : [ "../Plugins/" ],
"PythonScript" : "C:\\Orthanc\\meddream.py",
```

Afterwards the Orthanc service must be restarted.

Orthanc under Linux

Verify that Python plugin is activated by opening your plugins page, for example: http://localhost:8042/app/explorer.html#plugins.

Copy meddream.py to any suitable location, for example, /usr/local/share/orthanc/plugins/meddream.py.

In the configuration file orthanc.json, verify path to the plugins directory and add the path to your script, for example:

```
"Plugins" : [
    "/usr/share/orthanc/plugins", "/usr/local/share/orthanc/plugins"
],
"PythonScript" : "/usr/local/share/orthanc/plugins/meddream.py",
```

Afterwards the Orthanc service must be restarted.

Testing the script

1. Open any study in Orthanc and copy the value of “uuid” parameter from the browser address bar. For example, http://localhost:8042/app/explorer.html#study?uuid=dddddddd-cccccccc-bbbbbbb-00000000-aaaaaaaa.
2. Open a new tab and try to access the following URL (update the UUID with value from step 1): http://localhost:8042/studies/dddddddd-cccccccc-bbbbbbb-00000000-aaaaaaaa/info.

For example, if the study address was http://localhost:8042/app/explorer.html#study?uuid=c4dfb8ab-762b8b76-9a068673-4dbba15b-24167200, then the test URL shall be http://localhost:8042/studies/c4dfb8ab-762b8b76-9a068673-4dbba15b-24167200/info.

You should be able to see some JSON-formatted information.

3.3.2. DICOMweb

Universal HTTP-based communication over RESTful protocols like QIDO-RS and WADO-RS, and the legacy WADO-URI protocol.

3.3.2.1. DICOMweb notes

WADO-RS does not provide means to request a particular set of attributes. Study metadata from a server inclined to return all attributes can be enormous. It is advised to minimize the amount of data:

- Some non-standard parameter, like “dataset=meddream”, could be implemented in the server to support optimal and reduced attribute sets at the same time.
- The server could be configured for a reduced set if MedDream is the sole client.
- Some PACSes might return a smaller set of attributes via SearchForInstances of QIDO-RS, compared to RetrieveMetadata of WADO-RS. In that case, specify a full URL in `studyMetaUrl` (with as many “includefield” parameters as needed) and leave the `wadoRsUrl` option empty.

The DICOM Standard does not offer an attribute for the date/time when the study was received by the PACS. Similarly, there is some confusion regarding which attribute should indicate Application Entity Title of the sender. As a result, typical PACSes do not return this data and MedDream search columns “Received On” and “Source AE” remain empty. If you’ll find out what tags a particular PACS uses, then they can be configured via plugin options `...columns.receivedDate.tag` and `...columns.sourceAe.tag`.

The plugin supports the following authentication choices:

- no authentication;
- HTTP Basic Authentication;
- a custom HTTP POST (or, alternatively, GET) to a configurable address that accepts HTML FORM variables “login” and “password”, and returns HTTP 200/202 with a configurable cookie. This cookie is included in all subsequent requests and automatically refreshed when needed;
- Google Cloud service authentication for using a Google Cloud Healthcare data store instead of a PACS. It is enough to specify a JSON file downloaded from Cloud Console that contains keys and other parameters;
- Microsoft Azure authentication (based on client ID, client secret and OAuth2 URL that includes the tenant ID);
- AWS SigV4 authentication for downloading DICOM files from Amazon HealthImaging (as of Nov 2025, the server has limited support for search and study metadata).

When configuring a custom query in URLs, do not attempt to override some parameters added by the plugin dynamically; as the plugin does not check for duplicates, the server will likely accept the second occurrence (the duplicate from the plugin) instead.

Some servers require a reasonably accurate machine clock:

- a very large difference (tens of minutes) with respect to Google servers will result in an “Invalid JWT” error;
- Azure might have a similar effect because the token lifetime is just one hour;
- the SigV4 protocol of AWS tolerates clock differences of up to 5 minutes, and they will be detected during every request. The token itself is normally valid for 12 hours but that’s another matter – it only ensures that requests to STS will be quite rare.

3.3.2.2. Compatibility

Search operation uses SearchForStudies of QIDO-RS protocol, the header “Accept: application/dicom+json” is used by default (can be overridden). Attributes recognized in the response are:

- (0008,0020) Study Date
- (0008,0030) Study Time
- (0008,0050) Accession Number
- (0008,0061) Modalities In Study
- (0008,1030) Study Description
- (0010,0010) Patient Name
- (0010,0020) Patient ID
- (0020,000D) Study Instance UID
- a configurable custom attribute for AE Title(s) of senders contributing to the study
- a configurable custom attribute for study reception date/time

Most of them can also be used as query keys.

A certain filter, if enabled, checks the value of (0020,1208) Number Of Study Related Instances and silently aborts if the attribute is not present.

Fetching study metadata (the findStudy method) uses RetrieveMetadata of WADO-RS protocol, the header “Accept: application/dicom+json” is used by default (can be overridden). Alternatively, it is possible to configure a full URL that results in an equivalent list of instances and their attributes, like SearchForInstances of QIDO-RS. Attributes recognized in every SOP instance are:

- (0002,0010) Transfer Syntax UID
- (0008,0020) Study Date
- (0008,0016) SOP Class UID
- (0008,0018) SOP Instance UID
- (0008,0030) Study Time
- (0008,0050) Accession Number
- (0008,0060) Modality
- (0008,0090) Referring Physician’s Name
- (0008,1030) Study Description
- (0008,103E) Series Description
- (0010,0010) Patient Name
- (0010,0020) Patient ID
- (0010,0030) Patient Birth Date
- (0020,000E) Series Instance UID
- (0020,0010) Study ID
- (0020,0011) Series Number
- (0020,0013) Instance Number
- (0028,0008) Number Of Frames
- a configurable custom attribute for AE Title(s) of senders contributing to the study

Fetching detailed study metadata (as many attributes as possible via the findStudyAttributes method, not limited to the list above) must always use WADO-RS in order to function properly.

Download of a DICOM file can use WADO-URI protocol or RetrieveInstance of WADO-RS protocol. WADO-RS can return the file directly or wrapped in a "multipart/related" MIME container, however only the first body in the container, regardless of headers, is considered. The default Accept header is "application/dicom; transfer-syntax=*" (Google Cloud mode) or just "application/dicom" (others); the value can be overridden regardless of integration.

Uploading annotations and screenshots can use StoreInstances of STOW-RS protocol (disabled by default in favor of the global C-STORE based mechanism). The default header is "Accept: application/dicom+json" (can be overridden). The status codes 200 and 202 both mean a successful operation.

DICOM XML format for metadata and other responses is not supported.

3.3.2.3. MedDream configuration for DICOMweb

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to DICOMweb plugin. The documentation below is for plugin version 6.16.5; examples are from integrations with DCM4CHEE 5.25, Orthanc 1.5.8 with OrthancDicomWeb 1.0, PacsOne 7.2 and Google Cloud Healthcare as of June 2022.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Dicomweb`

     Use this specific value of `type` when connecting via DICOMweb.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=MainArchiveDW`

     Identifies the plugin/configuration pair in the search window drop-down menu. You can connect to multiple PACSes by using different values of `id` and remaining options.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.

     When targeting an older PacsOne installation that supports WADO-URI only, you can use `false` here and choose a different plugin (`PacsOne` or even `QR`) with same `id` and imageApiEnabled=false as the counterpart.

     Speaking of any PACS, this plugin's Image API can be used as an alternative if sharing DICOM files over the network is somehow not possible and slower operation is not an issue.

     Note: An equivalent of the legacy PHP-based `$pacs = 'WADO'` integration can be constructed from the QR plugin (Search API) and this plugin (Image API with wadoUriUrl).

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.

   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

(optional) `true`, together with `useSessionCache=true` and `com.softneta.preparation.useSessionCache=true`, enables cleaning of per-session cache subdirectories when the sessions end, and support for Clear Cache at the plugin side.

Default is `true` since 8.3.0.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storageApiEnabled=true`

  (optional) Specifies if this plugin/configuration pair is used to store annotations and screenshots (instead of the universal C-STORE based mechanism). Default is `false`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].prepareFromPacsMetadata=true`

  (optional) If true, Core will call a new plugin method intended for fetching as many instance-level attributes as possible, and will prepare instance metadata from it (not from the DICOM file as usual).

  Default is `false`. The plugin itself doesn't know about this setting.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].qidoRsUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs?fuzzymatching=true`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].qidoRsUrl=http://127.0.0.1:8042/dicom-web`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].qidoRsUrl=https://healthcare.googleapis.com/v1/projects/P1/locations/L2/datasets/D3/dicomStores/S4/dicomWeb?fuzzymatching=true`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].qidoRsUrl=https://my-azure-test-123.dicom.azurehealthcareapis.com/v1?fuzzymatching=true`

  Base URL of QIDO-RS API. Can include a query for additional or non-standard parameters. (The endpoint `/studies` and relevant dynamic parameters will be appended automatically.)

  In this example, DCM4CHEE 5, Google Cloud and Azure require the "fuzzymatching" parameter or else searching by Patient Name is not possible at all.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoRsUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoRsUrl=http://127.0.0.1:8042/dicom-web`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoRsUrl=https://healthcare.googleapis.com/v1/projects/P1/locations/L2/datasets/D3/dicomStores/S4/dicomWeb`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoRsUrl=https://my-azure-test-123.dicom.azurehealthcareapis.com/v1`

  Base URL of WADO-RS API. Can include a query for additional or non-standard parameters. (The endpoint `/studies` will be appended automatically.)

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyMetaUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies/{study}/metadata`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyMetaUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies/{study}/instances?includefield=0020000D&includefield=00080020&includefield=00080030&includefield=00100010&includefield=00100020&includefield=00100030&includefield=00100040&includefield=00200010&includefield=00080050&includefield=00080061&includefield=00080060&includefield=00081030&includefield=00200010&includefield=0020000E&`

`includefield=0008103E&includefield=00200011&includefield=00020010&includefield=00080018&&includefield=00080016&includefield=00280008&includefield=00200013`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyMetaUrl=http://127.0.0.1:8042/dicom-web/studies/{study}/instances?includefield=00100010&includefield=00100020&includefield=00100030&includefield=00080020&includefield=00080030&includefield=00080090&includefield=00280008`

  Alternative to wadoRsUrl. **This is a full URL, not its base part**; you must specify the query part, too, if needed. The placeholder `{study}` can appear anywhere.

  The second and third examples in fact mean QIDO-RS instead of WADO-RS. This makes no difference for the plugin, as long as the response format is the same.

  The second example contains all attributes needed for the study structure. Make sure to remove any line breaks after copying.

  > **Warning:** DCM4CHEE 5.x limits the number of instances (1000 by default) in a study that can be returned over QIDO-RS; in very large studies, some series might be incomplete or missing.
  >
  > If you chose the second example for performance reasons, then you'll also need to use **studyMetaUrlPageSize=990 (see further)** in order to enable paging. **Alternatively,** the limit can be increased in DCM4CHEE itself:
  >
  > 1. Log in to the DCM4CHEE UI.
  > 2. Open Configuration > Devices, click on "dcm4chee-arc" in the list.
  > 3. Click on "Child Objects".
  > 4. Click on "Network AEs" and select "DCM4CHEE".
  > 5. Click on "Edit extension".
  > 6. Click on "Attributes" at the end of the page.
  > 7. Find "Qido Max Number Of Results", enter a sufficiently large number (for example, 3000) and click "Save".
  >
  > The equivalent LDAP setting is dcmQidoMaxNumberOfResults.
  >
  > However, remember that the limit for this scenario was added in DCM4CHEE 5.10 to reduce memory requirements – the corresponding JVM might now require more.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyAttribUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies/{study}/metadata`

  (optional) The URL for a new attribute-rich scenario enabled with preparedFromPacsMetadata=true. As the default value is copied from wadoRsUrl and studyMetaUrl, the main purpose is to use WADO-RS here when studyMetaUrl uses QIDO-RS.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyCountersUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies?0020000D={study}&includefield=00201206&includefield=00201208`

  (optional) The URL for a dedicated query that retrieves only NumberOfStudyRelatedSeries and NumberOfStudyRelatedInstances. Do not use the standard endpoint .../studies/{study}/instances, as the PACS will repeat the values for every object in the study; the idea is to make the query as light-weight as possible.

  This dedicated query is used in multiple scenarios:

1. when structureListenerConfig in system.json enables polling of changes, and studyStructureCountersEnabled=true. Without studyCountersUrl, this polling will involve fetching the entire study structure from the PACS, where the counter values might still be missing (usually wadoRsUrl doesn't produce them);
2. when polling of changes is not used, the study structure is missing or outdated in the preparation cache, and cacheStudyMeta=true, then the plugin's findStudy method will fetch the counters in order to invalidate its own copy of cached study structure. Without studyCountersUrl, the plugin will just read a potentially outdated response from the cache, and only the temporary files cleaner (or Clear Caches with the "plugin caches" option) can force fetching a new structure from the PACS;
3. regardless of polling/caching mentioned above, the plugin always fetches the counters if studyCountersUrl is configured, then eventually updates the structure with their values if studyStructureCountersEnabled=true and existing values are either zeroes or missing. The update is performed for both online case (always fresh structure due to cacheStudyMeta=false) and cached case (structure is loaded from the disk – important if it was fetched via wadoRsUrl). Note that the latter is still be affected by No. 2 above, therefore the cached response will be up to date.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].metaAcceptHeader=application/json`

  (optional) Overrides the default Accept header `application/dicom+json` for search, study structure and attributes (prepareFromPacsMetadata=true).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].metaCustomHeader=x-test-1: value1`

  (optional) Adds custom header(s) for search, study structure and attributes (prepareFromPacsMetadata=true)

  Multiple entries are possible by using `|` as separator: `x-test-1: value1|x-test-2: value2|....`

  As of 8.5.0+, HIS-provided dynamic value placeholders `{custom1} ... {custom10}` (see further) can also be used here, however take care with the `|` character, as after the substitution MedDream will treat those as separators for multiple headers. Therefore, it is possible to pass multiple headers from HIS via a single dynamic parameter, but not to use this character in some header itself. No workarounds or escape sequences are offered at the moment.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoUriUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/wado`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoUriUrl=http://127.0.0.1:8042/wado`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoUriUrl=http://127.0.0.1/wado.php`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].wadoUriUrl=http://127.0.0.1/wado.php?username=UserLogin&password=UserPassword`

  Base URL of WADO-URI API. Can include a query for additional or non-standard parameters. (The relevant dynamic parameters will be appended automatically.)

  Google Cloud Healthcare and Azure DICOM service do not support WADO-URI, use dicomFileUrl instead.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomFileUrl=http://127.0.0.1:8042/wado?requestType=WADO&studyUID={study}&seriesUID={series}&objectUID={image}&contentType=application%2Fdicom`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomFileUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies/{study}/series/{series}/instances/{image}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomFileUrl=https://healthcare.googleapis.com/v1/projects/P1/locations/L2/datasets/D3/dicomStores/S4/dicomWeb/studies/{study}/series/{series}/instances/{image}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomFileUrl=https://my-azure-test-123.dicom.azurehealthcareapis.com/v1/studies/{study}/series/{series}/instances/{image}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomFileUrl=https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/e4558da0c596aa16ac2483858d155c1b/studies/{study}/series/{series}/instances/{image}`

  Alternative to wadoUriUrl. **This is a full URL, not its base part**; you must specify the query part, too, if needed. Placeholders `{study}`, `{series}` and `{image}` can appear anywhere.

  The examples 2. . . 5 in fact mean WADO-RS instead of WADO-URI. This makes no difference for the plugin, as long as the PACS still returns the DICOM file directly – or in the "multipart/related" format that is also indicated by the Content-Type header. (DCM4CHEE 5 supports only multipart from RetrieveInstance, see `fileAcceptHeader`.)
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].fileAcceptHeader=application/dicom`

  (optional) Overrides the default Accept header for file downloads (`application/dicom; transfer-syntax=*` if googleCloudConfigFile is configured, or `application/dicom` otherwise).

  To request a multipart format from PACSes that don't support other formats, try `multipart/related; type="application/dicom"`.
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].fileCustomHeader=x-test-1: value1`

  (optional) Adds custom header(s) for file downloads.

  Multiple entries are possible by using `|` as separator: `x-test-1: value1|x-test-2: value2|...`.

  As of 8.5.0+, HIS-provided dynamic value placeholders `{custom1} ... {custom10}` (see further) can also be used here, however take care with the `|` character, as after the substitution MedDream will treat those as separators for multiple headers. Therefore, it is possible to pass multiple headers from HIS via a single dynamic parameter, but not to use this character in some header itself. No workarounds or escape sequences are offered at the moment.
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].stowRsUrl=http://127.0.0.1:8042/dicom-web`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].stowRsUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].stowRsUrl=https://healthcare.googleapis.com/v1/projects/P1/locations/L2/datasets/D3/dicomStores/S4/dicomWeb`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].stowRsUrl=http://127.0.0.1/stowrs.php`

  Base URL of STOW-RS API. Can include a query for additional or non-standard parameters. (The relevant dynamic parameters will be appended automatically.)

  Azure DICOM service doesn't support the optional Study Instance UID that is automatically appended to this URL; use uploadUrl instead.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadUrl=http://127.0.0.1:8042/dicom-web/studies/{study}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadUrl=http://127.0.0.1:8080/dcm4chee-arc/aets/DCM4CHEE/rs/studies/{study}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadUrl=https://healthcare.googleapis.com/v1/projects/P1/locations/L2/datasets/D3/dicomStores/S4/dicomWeb/studies/{study}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadUrl=http://127.0.0.1/stowrs.php/studies/{study}`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadUrl=https://my-azure-test-123.dicom.azurehealthcareapis.com/v1/studies`

  Alternative to stowRsUrl. **This is a full URL, not its base part**. The `{study}` placeholder can appear anywhere. By the way, this parameter provides an opportunity to omit the standard "/studies/{study}" part because the latter is always automatically appended to stowRsUrl.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].sendAcceptHeader=application/json`

  (optional) Overrides the default Accept header (`application/dicom+json`) for uploading annotations and screenshots.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].sendCustomHeader=x-test-1: value1`

  (optional) Adds custom header(s) for uploading annotations and screenshots.

  Multiple entries are possible by using `|` as separator: `x-test-1: value1|x-test-2: value2|...`.

  As of 8.5.0+, HIS-provided dynamic value placeholders `{custom1} ... {custom10}` (see further) can also be used here, however take care with the `|` character, as after the substitution MedDream will treat those as separators for multiple headers. Therefore, it is possible to pass multiple headers from HIS via a single dynamic parameter, but not to use this character in some header itself. No workarounds or escape sequences are offered at the moment.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.tag=1048624`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.type=string`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.level=STUDY`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.strict=false`

  (optional) Remapping of attributes sent to the PACS as query/return keys and collected from the response. Also see 5.1.21 Custom columns in Patient History / Patient Studies for the overall idea.

  Fully tested values of <name> for this plugin are `modality`, `sourceAe`, `receivedDate`, `custom1 ... custom10`. More will be documented in the future.

  .tag specifies the tag of a DICOM attribute – an integer in decimal notation, e. g., (0123,4567) => 0x01234567 => 19088743. The columns "sourceAe" (was configured via sourceAeTitleTag before MedDream 8.2) and "receivedDate" (was configured via studyReceivedDateTag) are ignored by default as their .tag is zero. The special behavior of "modality" (especially collection of unique series-level values into a study-level value) is disabled if this column is reassigned with a non-default tag.

.type specifies additional processing of the value. `object` (default) means none. `number` is converted to Integer or Double **and works only for names custom1...custom10 (combination with other names is an error)**. For `string`, missing value is replaced with "". For `date`, a date formatted as yyyyMMdd is converted to yyyy-MM-dd. For `time`, a time formatted as HHmmss is converted to HH:mm:ss. `personName`, when dicomPersonName=false, replaces every "^" with a space. You can safely use `string` with the columns "sourceAe" and "receivedDate" mentioned above.

.level specifies where to request the attribute and expect it in the response. Default is STUDY, can be changed to SERIES or IMAGE.

.strict (false by default) can disable adding of wildcards by default when this attribute is used as a search key. The overall scenario and strictSearchIsEnabled can still override it. See also nonStrictSearchPattern, searchSymbols.

Migration for "sourceAe" and "receivedDate" from pre-8.2 versions looks as follows:

```
#com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].
↪sourceAeTitleTag=1048624
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.
↪sourceAE.tag=1048624
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.
↪sourceAE.type=string

#com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].
↪studyReceivedDateTag=19106219
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.
↪receivedDate.tag=19106219
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.
↪receivedDate.type=string
```

In 8.2+ it's still possible to specify any study-level attribute for .columns.sourceAE.tag or .columns.receivedDate.tag that the PACS is able to return. sourceAE should also work as a search key; receivedDate is only a return key. But, the column in search results will still have the original name which might confuse insufficiently trained end users. The only workaround (when using a single plugin) is to rename the text in the translation file; for English it's `...,"sourceAE":"Source AE",...` or `...,"receivedOn":"Received On",...` in sys/locales/en/translation.*.json.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].metadataTagsMapping=09992001:00020010;09992002:00280008`

  (optional) findStudy and findStudyAttributes can change any tag number in the server's response just before processing it. A typical use case is to rename private tags when their values are more adequate than of standard tags, or the corresponding standard tags are missing.

  The effect is similar to `columns` but is applied earlier, at lower level and without any conversion, destination values are unlimited, findStudies is not affected.

  The syntax for each semicolon-delimited entry is FROM:TO. The example above renames a few private tags 0999200x to the standard ones expected by the plugin.

  Note:

  - The rename operation changes the tag number only. The original tag (FROM) disappears from the dataset being parsed. If the destination tag (TO) already exists, it is overwritten. The plugin attempts to log an existing original tag at TRACE level for investigation; however, not all data types are fully supported.
  - A renamed tag is still parsed in the same way, therefore in most cases VR, VM and value encoding must match. You can't rename some byte stream with VR="UN" to a standard

SQ tag and expect its successful parsing as a sequence. Due to the flexibility of DICOM JSON encoding, you might succeed with mismatching VRs of string-like attributes, such as CS and LO.

– JSONs cached due to cacheStudyMeta=true are not affected by this setting – they still contain the original tag numbers. In scenarios where findStudy or findStudyAttributes are called without cleaning the plugin cache, they will load the cached response and remap the tags again. There is no second-level cache that prevents repeated processing related to alternateId, remapping, etc.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].multipleModalitySupport=true`

  (optional) `false` will yield an error message when the user selects multiple modalities. Use this if the PACS does not support multiple modalities and search results confuse the users (nothing is found).

  8.5.0+ contains a more user-friendly alternative when a single plugin is used: "Single modality support" in the Settings window (under Search).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].multivalueSeparatorIsComma=false`

  (optional) Since 4.3.0, the Modality filter is comma-separated by default, as recent versions of the Standard require this more clearly and more PACSes are following it. Earlier the Standard was referring to Query/Retrieve semantics; some PACSes (or their older versions) might still expect a backslash-separated filter in QIDO-RS queries, and you'll need to use `false` in those cases.

  A couple of tested PACSes return no results when this setting is incorrect and more than one modality is selected in the Search window. But, this still won't help if multiple modalities aren't supported at all (see `multipleModalitySupport`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomCacheDirectory=${com.softneta.meddream.tempDir}/dcm/MainArchiveDW`

  (optional) A directory for caching images. Without this setting a file is downloaded anew every time and examining its contents for troubleshooting becomes difficult.

  This plugin supports URIs (starting with `file://`). Making them relative to tempDir or another setting is encouraged to minimize human errors.

  Multiple instances of this plugin should have different directories in order to not mix up potentially different versions of the file with the same Study/Series/SOP Instance UIDs.

  As of MedDream 7.5.1+, this directory is not cleaned automatically and you need to configure a separate entry under `com.softneta.temp-cleaner.tempItems[]` (see *Temporary files cleaner*). `useSessionCache` below mitigates the situation to some extent.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheStudyMeta=true`

  (optional) If true, study structure and attributes responses will be cached for reuse. (Currently, both structure and attributes are stored in the same cached file named "study".)

  The content needs to be cleaned together with outdated DICOM files by Core's temporary files cleaner. This will work automatically when using the suggested pattern, . . . tempItems[. . .].pattern=*; however, a more specific pattern like *.dcm will prevent cleaning.

  This setting is effectively incompatible with studyAttribUrl, as findStudy with a QIDO-RS URL will initialize the cached file "study", then findStudyAttributes will immediately load an insufficient amount of attributes from it.

  On the other hand, it combines fairly well with wadoRsUrl: findStudy will take a longer time due to verbosity of WADO-RS, however afterwards findStudyAttributes will quickly load a cached response.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionCache=true`

  (optional) Enables per-session cache subdirectories. Default is `false`.

  The main benefit is that together with `eventApiEnabled=true` and `com.softneta.preparation.useSessionCache=true`, a session subdirectory is cleaned automatically when the session ends (for example, the user logs off). As a result, the DICOM files are cached for the shortest time possible. The temporary files cleaner is then needed mainly for unforeseen situations like application crashes or server restarts.

  On the other hand, when one user opens a particular study, then afterwards other users won't experience faster loading of the same study as the entire study is downloaded again for each of them.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=false`

  (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` forces wildcard matching everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window. `null` or unset is the default: exact matching for Patient History and HIS integration, wildcard matching for Search window.

  `true` is required for Google Cloud Healthcare due to its limitations. Furthermore, Google Cloud does not support searching by Study Description and MedDream is unable to handle this gracefully.

  `true` is required for Azure DICOM service, too. As of August 2022, the service supports fuzzy matching only for the Patient Name attribute and all other attributes must match exactly (wildcards are considered a literal part of the query).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].badUtf8Filter=true`

  (optional) `true` enables a workaround for incorrectly encoded UTF-8 characters in JSON responses related to qidoRsUrl, wadoRsUrl, studyMetaUrl and studyAttribUrl.

  The issue should be fixed in the PACS instead, as the workaround costs some performance overhead. Enable it only if search operations, or attempts to open a certain study, are failing and the application log contains a JsonMappingException complaining about "Invalid UTF-8".

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].trackUnexpectedEofByteWise=false`

  (optional) `false` or `true` enables a workaround for the "Premature EOF" exception when reading a response in "chunked" format from Azure servers. No workaround when unconfigured (default value).

  The value `false` ignores the exception at block level and can lose a few last KB of file content.

  The value `true` ignores the exception at byte level and is almost twice slower.

  We recommend "false" and to force the multipart format via fileAcceptHeader because then the plugin itself can detect a truncated response. Later absence of some data at the end of DICOM file can confuse the parser, which will be more difficult to troubleshoot.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].sCSeriesLowerPriority=true`

  (optional) `true` shifts series with Secondary Capture objects (classes ...1.1.7, ...1.1.7.*) further in the thumbnail strip by adding 10000 to the Series Number attribute of those objects. This is disabled by default.

  Since 8.3.1 the alternative is the Core setting `com.softneta.study.lowerPrioritySeriesModalities`: it works for all plugins, selects series by value of

Modality and moves them more effectively (there is no hardcoded increment of Series Number that might be insufficient for some studies).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].allowDuplicateObject=NO`

  (optional) Graceful handling of duplicate objects in the same series.

  MedDream uses the SOP Instance UID as a unique key when downloading a DICOM file, therefore the study structure (and response from wadoRsUrl/studyMetaUrl) can't have multiple entries with the same SOP Instance UID. However, it is possible to selectively ignore some of them.

  If the PACS always returns a more recent copy earlier, then you can safely choose `KEEP`: MedDream will use the first entry and ignore further duplicates.

  If the PACS always returns a more recent copy later, then you can safely choose `REPLACE`: MedDream will update the object attributes from every duplicate, effectively using the last entry.

  The risk behind a wrong choice is that MedDream will prepare itself for the value of an attribute found in the study structure, but later the PACS will return a DICOM file with a different value of the same attribute. Results might be unpredictable as MedDream is not tested with this kind of misinformation.

  The default value is `NO` and duplicates will cause an error like before plugin version 4.9.0.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].logResponseAmount=256`

  (optional) Change how many bytes of the server response are logged at DEBUG level.

  The default value of zero corresponds to 512 for file downloads and 524288 for search/metadata. A configured value replaces both limits at once.

  A common usage is to reduce size of the application logs due to a smaller value, like 32. A value larger than 512K can help when troubleshooting a crash related to parsing of JSON data.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomPersonName=true`

  (experimental) Return Patient Name and Referring Physician Name with DICOM separators (`^`) and include all five components. If `false`, return only last name and first name separated by a space. The ideographic and phonetic component groups are never returned, though.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].multivalueSeparatorIsComma=true`

  (optional) When `false`, multiple values of a search key (e. g., Modalities In Study) are delimited with backslashes and not commas. This might be needed by older PACSes.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].nonStrictSearchPattern=00100020:*{value}*;00080050:*{value}*`

  (optional) Specifies where to put wildcard characters when implementing a non-strict search. The value in the example is used by default, and yields the same behavior as before.

  Entries are separated with `;`. Format of the entry is `<tag code>:<value pattern>`. The `<value pattern>` part must contain a placeholder `{value}` that will be replaced with the search key; furthermore, space characters in the key are replaced with `*`. Currently recognized values for `<tag code>` are `00100020` and `00080050`.

  This would match only values beginning with the search key: `00100020:{value}*;00080050:{value}*`.

  This forces exact matching for Patient ID and Accession Number even with strictSearchIsEnabled=false: `00100020:{value};00080050:{value}`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchSymbols=*,?`

  (optional) Comma-separated list of wildcard characters. They aren't allowed in strict search scenarios (except `otherStrictSearchTags`), and will yield an error message.

  The default value with two characters from the Standard is a security improvement: it makes sure that the PACS won't return studies to which the end user has no access. An empty string means pre-8.0 behavior.

  In non-strict scenarios, including strictSearchIsEnabled=false or keys like Study Description, these characters are used to decide that the search key is good enough — namely, it doesn't need formatting by `nonStrictSearchPattern` and spaces aren't to be replaced with asterisks. When the end user is using the Search window, (s)he can explicitly add wildcards where deemed appropriate: for example, `PID001` is sent to the PACS as `*PID001*`, `PID 001` – `*PID*001*`, `PID001*` – `PID001*`, `PID?001` – `PID?001`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].otherStrictSearchTags=524384,19088743`

  (optional) Override which non-integration attributes are used as strict search filters. Tag numbers must be in decimal notation and separated by commas.

  Supported values are Modality (specified by 524384, which is the same as 0x00080060) and Source AE Title (specified by the same nonzero value as in `sourceAeTitleTag`).

  Modality is included by default; therefore you can use an empty string for a non-strict search by modality. Keep in mind afterwards that some standard modality codes include each other, like "US" will also match "IVUS", "OP" will also match "OPV", etc. This will be useful for PACSes that are unable to find, say, "MR" in a list like "KO,MR,PR".

  Both attributes are partially subject to `searchSymbols`. A query key is allowed to contain this kind of wildcard character; if there is one, and `otherStrictSearchTags` does not configure a strict search for the key, then leading and trailing `*` characters are added unconditionally – without using `nonStrictSearchPattern`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchPageSize=500`

  (optional) Maximum value of the QIDO-RS "limit" parameter (hard limit on search results) related to qidoRsUrl, when MedDream's legacy value, 1000, is too large for the server.

  If less than 1000, then MedDream performs multiple requests with corresponding "offset" and "limit" parameters until up to 1000 results are collected in total.

  By default, it's 200 when `azureAuthUrl` is not empty, and 1000 otherwise.

  A value not in range 1...1000 yields a warning and is replaced with a default. The Azure DICOM server, in turn, supports a shorter range – 100...200.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyMetaUrlPageSize=190`

  (optional) Maximum value of the QIDO-RS "limit" parameter related to studyMetaUrl in a QIDO-RS (not WADO-RS) context, when the server enforces some limit and therefore can't enumerate objects of a large study at once.

  If configured (larger than 0), then MedDream performs multiple requests with corresponding "offset" and "limit" parameters until the server returns a different number of results than requested.

  Zero by default, which corresponds to the legacy behavior (no "limit" parameter in structure requests). Known limit of Azure servers is not applied as a default, in favor of installations where studyMetaUrl is configured for WADO-RS.

Warning: A very rare misbehavior is possible:

1. Suppose the server doesn't support "offset" or "limit" (highly likely for a WADO-RS URL, as the Standard doesn't define them there), but you don't know that yet.
2. Furthermore, the server just ignores any unrecognized request parameters instead of reporting an error.
3. You try to open a study that *incidentally* contains exactly studyMetaUrlPageSize objects.
4. The server always returns the same result to every request despite different "offset", the plugin collects them and doesn't know when to stop, eventually JVM is out of memory unless you kill it earlier.

The plugin can only warn if you configure studyMetaUrlPageSize together with wadoRsUrl by mistake. You must test the pagination with large enough studies: studies smaller than studyMetaUrlPageSize won't prove anything, while an exact match will be unpleasant.

If the server's default limit, like 200 for Azure, is known in advance, then it's safer to configure studyMetaUrlPageSize with a slightly smaller value (smaller by one is enough). The plugin can then immediately detect that a particular URL doesn't support pagination because it returns more results than requested.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchSkipsSingleObjOfMods=KO,PR`

  (optional) Search results will not contain records where Modalities In Study has a single value matching any entry in this comma-separated list and Number Of Study Related Instances is 1.

  This option was created as a workaround for a certain PACS that lists studies consisting of a single certain SR object, yet is unable to return study metadata for it and the error message confuses the end users.

  It could also be used to hide studies consisting of a single non-image object like MedDream's own KO, PR or RTSTRUCT: studies like these will result in a message about an empty study.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].errorHeaderName=X-Error-Occurred`

  (optional) Name of a custom header (case sensitive) that provides a user-friendly explanation in case of an error reported by the server. Not configured by default.

  The contents of this header will be visible in the user interface. Commas in value (header value separators) are converted to hard line breaks (character code 10). The value must be plain ASCII or encoded in ISO-8859-1 (Latin-1), a default for HTTP headers.

  This setting is supported by Storage API, Image API, Search API and authType=custom. The latter might interface a custom-built software where such a header is deliberately implemented; there is no support in authType=dcmsys and authType=azure scenarios, though, until there is no known header that could be useful for diagnostics of a login failure.

  Before using this functionality, you should do a security audit of server software and examine what content can come in this header. Ideally it should be a small tightly controlled set of messages. If dynamic values in them are strings, these need to be audited further.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=5`

  (optional) A timeout in seconds, used both for socket connect and socket read operations. (If the server somehow manages to delay both, then the actual pause until failure will be twice as long.)

  The default behavior depends on JRE implementation for a particular operating system. There might be additional limitations: for example, Oracle JRE under Windows uses approx.

21 second for a connection timeout, and larger values of `timeout` are ignored (one can only make the timeout smaller).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables adding of NumberOfStudyRelatedSeries/NumberOfStudyRelatedInstances to the study structure; is required for structureListenerConfig in system.json. Disabled by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].alternateIdPrefixToRemove=/aets/{custom5}/`

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].alternateIdSuffixToAppend=?c={custom6}&f={{frameId}}`

  (optional) Similar to alternateURITemplate in the AWSHealthImaging plugin: if not empty, then every image in the structure will have the "alternateID" field derived from image's 7fe0,0010 BulkData JSON attribute with `alternateIdPrefixToRemove` removed and `alternateIdSuffixToAppend` appended. (The example appends "?c=... &f={{frameId}}" to alternateID.)

  Supports placeholders {custom1} ... {custom10} for dynamic values from the token, but ignores the placeholder {{frameId}} which will be filled by the frontend. No value by default.

  When these settings are in effect, the attribute (0028,3010) VOI LUT Sequence is validated. If present, it must contain (0028,3002) LUT Descriptor, (0028,3003) LUT Explanation, and (0028,3006) LUT Data) or else "alternateID" is still not added. Additional consequence of this invalid VOI LUT is that (0002,0010) Transfer Syntax UID returned from findStudy and findStudyAttributes will be set to "", as a hint for the frontend to not use the alternateID scenario.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].alternateIdURITemplate={custom7}/studies/{study}/series/{series}/images/{image}?f={{frameId}}`

  (optional) Equivalent of alternateURITemplate in the AWSHealthImaging plugin. If not empty, then every object in the structure will have the "alternateID" field built from this template.

  The template can contain placeholders {study}, {series}, {image}, {custom1} ... {custom10} that will be filled before alternateID reaches the frontend. The placeholder {{frameId}} is deliberately ignored and remains as it is. You can use any suitable web path that will be handled by the reverse proxy.

  Not compatible with alternateIdPrefixToRemove / alternateIdSuffixToAppend (the plugin then refuses to continue).

  When this setting is in effect, the attribute (0028,3010) VOI LUT Sequence is validated. If present, it must contain (0028,3002) LUT Descriptor, (0028,3003) LUT Explanation, and (0028,3006) LUT Data) or else "alternateID" is still not added. Additional consequence of this invalid VOI LUT is that (0002,0010) Transfer Syntax UID returned from findStudy and findStudyAttributes will be set to "", as a hint for the frontend to not use the alternateID scenario.

  Note: As of MedDream 8.8.0+, alternateID is added to any kind of object in the study structure, including non-images like SR. As their metadata doesn't contain the 7fe0,0010 DICOM attribute at all, those objects don't get alternateID when using the older alternateId*To* options.

  This is considered a minor nuisance, as the frontend can already recognize which objects are images and therefore ignore the redundant field for others. Even the traditional scenario (without alternateID) uses different endpoints for non-images.

4. Review authentication parameters (below).

5. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for chosen method.

Authentication

Since MedDream 8.4 the type of authentication can be specified explicitly via the `authType` parameter.

- none

  The server must accept fully anonymous connections.

  At the moment this is *not yet* the default value when authType is missing. Now a missing authType means that the legacy auto-detection will be performed, and everything will work as earlier (except annoying warnings in the log).

- basic

  HTTP Basic Authentication. The header Authorization: Basic is added to every DICOMweb request.

  Requires parameters `username` and `password` that will be encoded in the header.

- custom

  Simulates an HTML FORM action (POST by default, or GET when `loginWithGET` is true) to `loginUrl`. Uses hardcoded parameters "login" (for `username`) and "password" (for `password`), they are passed via Body (POST, in x-www-form-urlencoded format) or query string (GET). The cookie with name configured via `loginCookie` is expected in the response, and will be added to every DICOMweb request. The sequence is repeated automatically when the cookie expires.

  Requires parameters `loginUrl`, `username`, `password`. Optional ones are `loginWithGET` and `loginCookie`.

- google

  Google Cloud platform authentication via the "google-auth-library-oauth2-http" third-party library. The obtained access token is added to every DICOMweb request via Authentication: Bearer header, and is refreshed automatically when it expires. Involves a different default set of Accept headers.

  Requires the parameter `googleCloudConfigFile`.

  Please remember to enter `dicomFileUrl` in WADO-RS format instead of WADO-URI format (the latter is not supported by Google).

- azure

  OAuth2 implementation in Microsoft Azure. Performs a GET request to `loginUrl` (must include the tenant/directory ID); the body encodes a JSON payload with fields "client_id" (from `clientId`), "client_secret" (from `clientSecret`), "grant_type" (hardcoded value "Client_Credentials"), "resource" (hardcoded value of a certain Azure URL). This is indeed a GET request (not POST) with an HTTP Body. The expected response is HTTP 200 with a JSON payload that includes fields "token_type" (equal to "Bearer"), "expires_in" and "access_token". The value of access_token is included in all subsequent DICOMweb requests via the Authorization: Bearer header. The sequence is repeated automatically when the token expires.

  Requires parameters `loginUrl`, `clientId`, `clientSecret`.

  For some time, the legacy convention will be supported together with explicit authType=azure: `azureAuthUrl` can be used instead of `loginUrl`, `username` – instead of `clientId`, `password` – instead of `clientSecret`, although warnings about obsolete parameters will be logged. After the obsoletion period ends, `azureAuthUrl` will be removed, while `username` and `password` will be reserved for other login methods.

- dcmsys

Another proprietary implementation of OAuth2. Performs a POST request to `loginUrl`, the body encodes a JSON payload with fields "client_id" (from `clientId`), "client_secret" (from `clientSecret`), "grant_type" (hardcoded value "password"), "username" (from `username`), "password" (from `password`). The expected response is HTTP 200 with a JSON payload that includes fields "token_type" (equal to "Bearer"), "expires_in" and "access_token". The value of access_token is included in all subsequent DICOMweb requests via the Authorization: Bearer header. The sequence is repeated automatically when the token expires.

Requires parameters `loginUrl`, `clientId`, `clientSecret`, `username`, `password`.

- aws

  Amazon Web Services authentication (SigV4 as implemented in the AWS SDK) for accessing DICOM files stored in a HealthImaging service. Credentials consist of an access key ID and a secret access key (the preconfigured session token is ignored), they are system-wide and automatically fetched in various ways, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/credentials-chain.html. A new session token, obtained via GetSessionToken from STS, is used to "sign" every DICOMweb request via corresponding headers, and is refreshed automatically when it expires.

  Requires the parameter `awsRegion`.

  Please remember to use *dicomFileUrl* in WADO-RS format (AWS doesn't support WADO-URI) and also add *fileAcceptHeader* that specifies one of two supported transfer syntaxes.

  > **Warning:** At time of this writing (Oct 2024), AWS does not support QIDO-RS, RetrieveMetadata of WADO-RS, STOW-RS. SearchAPI and StorageAPI from a different plugin will be needed.

The legacy logic (when `authType` is missing) detects certain types in this order:

- google: not empty googleCloudConfigFile;
- azure: azureAuthUrl is not empty and loginUrl is empty;
- custom: loginUrl is not empty and azureAuthUrl is empty;
- basic: loginUrl and azureAuthUrl are empty, however both username and password are not empty;
- none: when no condition above matches.

The detection still works as of 8.4 but is now obsolete and yields warnings in the log; you should update the configuration instead of waiting until it isn't accepted at all. There is no legacy equivalent for authType=dcmsys and authType=aws.

A reference for all authentication parameters:

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].authType=azure`

  (optional) The preferred way to specify authentication type. If missing, the legacy autodetection is performed. If present, then other authentication parameters are validated more strictly.

  Possible values: none, basic, custom, azure, google, dcmsys.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].clientId=123e4567-e89b-12d3-a456-426652340000`

  Value for the "client_id" request parameter. This is the only way for authType=dcmsys, and the preferred way for authType=azure (instead of `username`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].clientSecret=SW4gdGhlIGJlZ2lubmluZyB3YXMgdGhlIFdvcmQ=`

  Value for the "client_secret" request parameter. This is the only way for authType=dcmsys, and the preferred way for authType=azure (instead of `password`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].username=UserLogin`

  Username for Basic Authentication, custom form authentication or Azure authentication (in the latter case this parameter expects the client ID and is obsolete).

  Not needed for default configuration of DCM4CHEE 5. Definitely needed for Orthanc with its AuthenticationEnabled=true. Needed for PacsOne by default unless credentials are included in the URL (less secure).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].password=UserPassword`

  Password for Basic Authentication, custom form authentication or Azure authentication (in the latter case this parameter expects the client secret and is obsolete).

  Not needed for default configuration of DCM4CHEE 5. Definitely needed for Orthanc with AuthenticationEnabled=true. Needed for PacsOne by default unless credentials are included in the URL (less secure).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].loginUrl=http://server/path`

  Legacy convention: URL for a custom POST/GET request. Present convention: URL for a custom POST/GET request, for Azure token request, for "dcmsys" token request.

  Obviously not applicable to DCM4CHEE, Orthanc or PacsOne.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].loginWithGET=true`

  (optional) When `true`, loginUrl is used to perform a GET request for the custom form authentication instead of POST. Credentials (`login` and `password`) are automatically added to the URL.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].loginCookie=MySessionCookie`

  (optional) Cookie name for the custom form authentication. Default value is "suid".

  Obviously not applicable to DCM4CHEE, Orthanc or PacsOne.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].azureAuthUrl=https://login.microsoftonline.com/936da01f-9abd-4d9d-80c7-02af85c822a8/oauth2/token`

  (obsolete) Enables the Azure authentication (`loginUrl` must be empty when not using authType). The URL must be in form shown above; the GUID is to be replaced with your tenant ID on Azure platform.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].googleCloudConfigFile=C:\MedDreamPACS-Premium\MedDream\gc-login.json`

  Google Cloud Healthcare integration mode: a certain authentication and a different default Accept header for file downloads.

  The file must contain a service account private key in JSON format, as created by the IAM & Admin console.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsRegion=us-east-1`

  Amazon HealthImaging integration mode: AWS region string. Will be validated by the AWS SDK.

3.3.2.4. HIS-provided dynamic values in configuration strings

The parameters `qidoRsUrl`, `wadoRsUrl`, `studyMetaUrl`, `wadoUriUrl`, `dicomFileUrl`, `stowRsUrl`, `uploadUrl`, `metaCustomHeader`, `fileCustomHeader`, `sendCustomHeader`, `googleCloudConfigFile`, `alternateIdPrefixToRemove` and `alternateIdSuffixToAppend` can have 10 placeholders "custom1" ... "custom10" for integration-specific dynamic values:

- `{custom1}`
- `{custom1:DEFAULT_VALUE}`
- `{custom1:}`
- ...
- `{custom10:DEFAULT_VALUE}`

(in the 3rd example, the default value is an empty string).

Only those ten are supported. Other documented placeholders – "{study}", "{series}" and "{image}" – are handled elsewhere in the plugin. Another special placeholder, "{{frameId}}", is explicitly ignored as it will be handled in the frontend. Any other (unknown) placeholder name will result in a fatal error.

Dynamic values come from the "storageConfiguration" part of the HIS integration token. The corresponding parameters there are called DICOMWEB_CUSTOM_1 ... DICOMWEB_CUSTOM_10.

If the default value isn't specified together with the placeholder, then a corresponding dynamic value must be provided in the token. This means that when MedDream is used both interactively and from HIS, the default values are unavoidable.

For example, the setting `...[0].googleCloudConfigFile=/opt/meddream/gc-{custom1:interactive}.json` refers to a file "gc-interactive.json" for scenarios with MedDream login and search, while a token

```
{
    "items": [ ],
    "storageConfiguration": [ { "storage": "gc1", "parameters": [
        { "name": "DICOMWEB_CUSTOM_1", "value": "his" }
    ] } ]
}
```

requests to use a file "gc-his.json" instead.

Note: The bundled token service doesn't recognize DICOMWEB_CUSTOM_* by default and needs the following configuration string in its own application.properties file:

```
validate.storage-configurations.parameter-list=DICOMWEB_CUSTOM_1,DICOMWEB_CUSTOM_2,\
    DICOMWEB_CUSTOM_3,DICOMWEB_CUSTOM_4,DICOMWEB_CUSTOM_5,DICOMWEB_CUSTOM_6,\
    DICOMWEB_CUSTOM_7,DICOMWEB_CUSTOM_8,DICOMWEB_CUSTOM_9,DICOMWEB_CUSTOM_10
```

3.3.3. Amazon S3

S3 is a generic cloud object storage solution from Amazon.

By itself S3 doesn't provide metadata that is needed by MedDream early – during initial stages of study loading. In comparison, the "File system mode" somewhat gets away by extracting a minimum amount of metadata from files that are immediately available on a rather quick local file system, therefore the initial delay remains bearable. With S3, however, the latency is too high; neither an `s3fs` mount, nor a custom client for iterating objects below the given prefix, can result in acceptable responsiveness of MedDream.

We suggest a custom manifest in JSON format (a ready to use "study structure") to speed up the collection of metadata.

1. MedDream is started with the "study" parameter (legacy URL-based HIS integration), or with an equivalent token (token-based HIS integration).
2. An HTTP GET request downloads the JSON manifest for a particular Study Instance UID that was passed in Step 1.
    1. (In fact, there are two identical requests: one basically verifies that this study is present, and the second one converts the manifest to a study structure. 8.8.0+ is able to call a different, potentially more efficient, endpoint in the first request.)
3. Based on the manifest, MedDream displays the initial empty study structure and related metadata. Image-level URLs to DICOM files listed in the manifest are cached in memory every time a manifest is downloaded.
4. DICOM files are downloaded on demand (depending on whether the thumbnail is visible, or if the "Preload" function is used, etc.) and cached on disk. The cached copies are reused later until the *Temporary files cleaner* removes them.

3.3.3.1. Amazon S3 notes/limitations

The Search window is not supported. If enabled, its contents will remain empty. This is a typical HIS integration.

Without useS3SDK=true, there is no specific authentication/authorization between MedDream and S3, or between MedDream and the manifest service. In general, DICOM files in the S3 bucket must be either readable anonymously (the `s3:ListBucket` permission should be denied, though), or accessible due to `X-Amz-Credential` and similar options in the URL. The implementer can choose any convention that results in short-lived URLs. The manifest service should be accessible only by MedDream – due to a firewall, to an IP address whitelist, or just by listening on a loopback address in the same machine.

It's a tempting idea to return short-lived URLs to DICOM files from the study structure service. A reasonable time is a few hours. Minutes might be insufficient, as MedDream does not always download every file in the study (only Preload Study/Series takes care of that) and even otherwise a combination of network bandwidth and study size sometimes means more time. An age limit smaller than com.softneta.cache.studyStructureSec (300 seconds by default) is harmful, as Core would be using cached URLs that aren't working anymore.

Starting from 8.7.0, it is possible to configure a REST-like URL for downloading the manifest as a static resource on the same S3 bucket.

Other manifest formats, like Basic Directory IOD (DICOMDIR), are not supported at the moment.

The downloaded DICOM files are always cached on disk, and the corresponding configuration option (cache directory) is mandatory.

Do not change the two settings common to most plugins, `searchApiEnabled` and `imageApiEnabled`. The values are true by default, and `false` will not work properly: particularly, ImageAPI of this plugin depends on URLs cached by its SearchAPI.

Saving annotations will not work unless the customer's solution includes a DICOM Storage SCP that propagates the changes both to the S3 and the manifest service.

3.3.3.2. MedDream configuration for Amazon S3

1. **Perform steps in the 5.1.1 Essential configuration and the first run chapter.**
    1. Configure the HIS integration (preferably token-based) in `application.properties` file:

```
spring.profiles.include=auth-his
#authentication.his.validHisParams=study
authentication.his.token-service-address=http://127.0.0.1:8088/v2/validate
authorization.defaultHisPermissions=EXPORT_ISO,EXPORT_ARCH,FORWARD,\
    UPLOAD_DICOM_LIBRARY,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
```

2. Edit `application.properties` file and update the options related to Amazon S3 plugin. The documentation below is for plugin version 3.8.1.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=AmazonS3`

  Use this specific value of `type`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=S3`

  Identifies the plugin configuration. You can use multiple manifest services etc. by using different values of `id` and remaining options.

  An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

  (optional) `false` will turn off Clear cache support in the plugin. True by default.

  Note that this is not related to automatic cleaning of the cache when the session ends. This plugin, unlike some others, doesn't support the useSessionCache setting and its cache is always global.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storageApiEnabled=true`

  (optional) Specifies if this plugin is used to store annotations and screenshots (instead of the universal C-STORE based mechanism). Requires configuration of more related properties. Default is `false`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureServiceUrl=http://internal-production-alp.us-east-2.elb.amazonaws.com/Integration/GetStructureByStudy?studyInstanceUID={study}`

  The endpoint that returns a manifest JSON given the Study Instance UID. The latter must be specified explicitly using the `{study}` placeholder. The placeholders "{custom1}" ... "{custom10}" (see the next chapter) are also supported.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studiesServiceUrl=http://internal-production-alp.us-east-2.elb.amazonaws.com/Integration/GetStructureByStudy?studyInstanceUID={study}`

  The endpoint used to validate presence of the study: it is enough to return an almost empty study structure (only the StudyInstanceUID attribute). See the manifests for "findStudies" instead of "findStudy" in further chapters.

  By default is initialized with studyStructureServiceUrl.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].uploadServiceBaseUrl=http://internal-production-alp.us-east-2.elb.amazonaws.com/api/upload`

  Base URL to the upload service for annotations and screenshots. Required when storageApiEnabled=true. Not compatible with studyStructureType=DEFAULT.

  1. The plugin performs a GET request to ${uploadServiceBaseUrl}/link. There is a "fileName" query parameter; however, due to architectural limitations, the plugin uses a random, UUID-based, file name with the ".dcm" extension. The setting `sendCustomHeader` can provide additional HTTP headers. The server must return a JSON with mandatory non-empty fields "presignedUrl", "bucketName" and "objectKey".
  2. Then the plugin uses the traditional S3 API to perform a PUT request to presignedUrl (no additional authentication needed) and therefore upload contents of the DICOM file. HTTP 200 and 201 mean success.
  3. The customer's system is now supposed to import the file so that it becomes available to the study structure service. The plugin initiates the import job by calling GET ${uploadServiceBaseUrl}/validateAndArchive where Body is a JSON with fields from the first response, "bucketName" and "objectKey", and additionally "patientId" and

"accessionNumber" extracted from the DICOM file. The setting `sendCustomHeader` is used again if configured. The server must return an empty response with Status 200.

4. A copy of the uploaded file is cached under `dicomCacheDirectory` in advance.
5. The import is expected to be rather slow and asynchronous: a success response from validateAndArchive doesn't mean that the new object will be in the study structure after fetching a new version of it. Unless your implementation can complete the import in reasonable amount of time and then return from validateAndArchive, **you must configure** com.softneta.meddream.secToWaitAfterCreatedDICOMSend and `checkInstanceAvailableUrl`. That way Core will wait for some more time until the object appears in the study structure.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].checkInstanceAvailableUrl=http://internal-production-alp.us-east-2.elb.amazonaws.com/api/gateways/query/images`

  A final URL for a POST request that accepts a JSON-formatted reference in Body, then returns HTTP 200 where Body indicates whether the referenced object has been imported. An empty Body, or `[]`, is treated as "not imported yet"; any other content, even a single byte, indicates a finished import.

  The request body contains fields "studyInstanceUID", "seriesInstanceUID", "sopInstanceUID" (already known to Core and plugin) and "patientId" (comes from `customParameters` property that must contain it, like `patientId: {custom3}|someOtherPrm: ...`).

  Headers from `checkInstanceCustomHeader` will be added to the request if configured.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyCountersUrl=http://internal-production-alp.us-east-2.elb.amazonaws.com/api/gateways/query/counters?study={study}`

  A final URL for the counters service; mandatory when studyStructureCountersEnabled=true. Must return a JSON with numeric fields "numberOfStudyRelatedSeries" and "numberOfStudyRelatedInstances".

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureType=DEFAULT`

  (optional) What kind of JSON structure is expected from the structure service. Currently, two values are supported: "DEFAULT" (used by default) and "TELEPAXX".

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].metaCustomHeader=x-test-1: value1`

  (optional) Adds custom header(s) in requests to the structure service.

  Multiple entries are possible by using `|` as separator: `x-test-1: value1|x-test-2: value2|...`.

  HIS-provided dynamic value placeholders `{custom1} ... {custom10}` (see the next chapter) can be used here, however take care with the `|` character, as after the substitution MedDream will treat those as separators for multiple headers. Therefore, it is possible to pass multiple headers from HIS via a single dynamic parameter, but not to use this character in some header itself. No workarounds or escape sequences are offered at the moment.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].fileCustomHeader=x-test-2: value2`

  (optional) Adds custom header(s) in file download request when using useS3SDK=false. With studyStructureType=TELEPAXX, the server must also return bearerTokenNeeded=true for a particular image, or else this setting is ignored.

  Multiple entries are possible by using `|` as separator: `x-test-1: value1|x-test-2: value2|...`.

HIS-provided dynamic value placeholders `{custom1} ... {custom10}` (see the next chapter) can be used here, however take care with the `|` character, as after the substitution MedDream will treat those as separators for multiple headers. Therefore, it is possible to pass multiple headers from HIS via a single dynamic parameter, but not to use this character in some header itself. No workarounds or escape sequences are offered at the moment.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].sendCustomHeader=x-test-2: value5`

  (optional) Adds custom header(s) during the file upload (for `uploadServiceBaseUrl`). Same format as in metaCustomHeader and fileCustomHeader.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].checkInstanceCustomHeader=x-test-2: value3`

  (optional) Adds custom header(s) during the instance availability check (for `checkInstanceAvailableUrl`). Same format as in metaCustomHeader and fileCustomHeader.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyCountersCustomHeader=x-test-2: value4`

  (optional) Adds custom header(s) when calling the study counters service (for `studyCountersUrl`). Same format as in metaCustomHeader and fileCustomHeader.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].customParameters=patientId: {custom3:}`

  Additional parameters for plugin logic that can get the values from HIS via the "{customN}" placeholders.

  As of MedDream 8.9.0+, this option provides the value of Patient ID for the instance availability check. Is required and must contain the "patientId" entry when `checkInstanceAvailableUrl` is configured. Same format as in metaCustomHeader and fileCustomHeader.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useS3SDK=true`

  (optional) Use the Amazon S3 SDK instead of simple HTTP requests. This allows managing access via IAM roles at the server side. Disabled by default.

  When `true`, the manifest JSON must contain `s3://...` URLs.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsRegion=eu-central-1`

  Explicitly specifies the AWS region. Required by `useS3SDK=true`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imapProfile=ENVIRONMENT_VARIABLE_CREDENTIALS_PROVIDER`

  (optional) Specify how to fetch credentials for the SDK.

  Possible values are:

  - `ENVIRONMENT_VARIABLE_CREDENTIALS_PROVIDER` – expects environment variables `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN`;
  - `INSTANCE_PROFILE_CREDENTIALS_PROVIDER` – uses the EC2 metadata service (EC2 installations only).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomCacheDirectory=${com.softneta.meddream.tempDir}/dcm/AmazonS3`

  Cache directory for downloaded DICOM files.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomUrlMappingItemsCount=300000`

  (optional) Size of the URL cache. The default value is 250000.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) Enable support for (potentially lightweight) calls that fetch NumberOfStudyRelatedSeries and NumberOfStudyRelatedInstances. False (disabled) by default. `true` also requires `studyCountersUrl`.

  The setting is needed when Core is configured to poll the counters according to structureListenerConfig in system.json, or perform on-demand validation according to com.softneta.cache.studyStructureExpiresOnChange in application.properties.

  In most other plugins, this setting just enhances the study structure queries (or parsing of their result) so that both counters are returned in the structure. A dedicated lightweight query to the PACS is normally performed only when Core requests just the counters.

  AmazonS3, however, calls studyCountersUrl in all cases: when Core requests the full structure and not just counters, the counters in the structure will come from the studyCountersUrl call. There is no attempt to add the two tags from the structure service response, and we didn't update the manifests accordingly. The manifest approach would save 1-2 "redundant" calls to studyCountersUrl only during the moment when the polling/validation logic decides that a full structure needs to be fetched again. This is negligible compared to vast amount of studyCountersUrl calls spent before the change happens. There might be a smaller overall number of studyCountersUrl calls in case of studyStructureExpiresOnChange; but, their share is still small.

  If you don't intend to implement a separate study counters service, then studyCountersUrl can have the same value as studyStructureServiceUrl or studiesServiceUrl, provided that the service also returns JSON fields "numberOfStudyRelatedSeries" and "numberOfStudyRelatedInstances" at top level (any other fields will be ignored).

3.3.3.3. HIS-provided dynamic values in configuration strings

The parameters `studiesServiceUrl`, `studyStructureServiceUrl`, `metaCustomHeader` and `fileCustomHeader` can contain 10 placeholders "{custom1}" . . . "{custom10}" for integration-specific dynamic values:

- `{custom1}`
- `{custom1:DEFAULT_VALUE}`
- `{custom1:}`
- . . .
- `{custom10:DEFAULT_VALUE}`

(in the 3rd example, the default value is an empty string).

Only those 10 are supported. The "{study}" placeholder is handled elsewhere. Any other (unknown) placeholder name will result in a fatal error.

Before inserting into studieServiceUrl or studyStructureServiceUrl, the value is URL-encoded (to avoid problems with values containing spaces, etc.). No encoding is done for other destinations.

Dynamic values come from the "storageConfiguration" part of the HIS integration token. The corresponding parameters there are called identically, "custom1" . . . "custom10".

If the default value isn't specified together with the placeholder, then a corresponding dynamic value must be provided in the token.

Note: The bundled token service doesn't recognize "custom*" by default and needs the following configuration string in its own application.properties file:

```
validate.storage-configurations.parameter-list=custom1,custom2,custom3
```

3.3.3.4. Specification of the “DEFAULT” manifest

The following JSON file illustrates the supported metadata for studyStructureType=”DEFAULT”. Attributes are grouped into study level and instance+series levels (MedDream afterwards uses series-level attributes in the latter for proper grouping into series).

The plugin ignores the Content-Type response header and always parses the payload as JSON.

```
{
  "instanceUid": "1.2.3",
  "studyDate": "20140710",
  "studyTime": "084944.000",
  "studyReceivedDate": "",
  "studyId": "1",
  "studyDescription": "PLC CT CHE-ABD-PEL-CORONARY CAL SCORE-VIRTUAL COLONOSCOPY",
  "accessionNumber": "1000801048",
  "referringPhysicianName": "DOE^JOSEPH^X.",
  "modalitiesInStudy": "CT",
  "sourceAETitle": "",
  "patientId": "200091016",
  "patientName": "DOE^JOHN",
  "patientBirthDate": "19710107",
  "sops": [
    {
      "seriesInstanceUid": "4.5.6.70",
      "seriesDescription": "Cardiac 3.0 Cardiac/Score",
      "modality": "CT",
      "seriesNumber": "2",
      "sopClassUid": "1.2.840.10008.5.1.4.1.1.2",
      "transferSyntaxUid": "1.2.840.10008.1.2",
      "instanceNumber": "4",
      "sop": "8.9.10.100",
      "frameCount": 1,
      "dicomUrl": "https://some-server.s3.us-east-2.amazonaws.com/dicom/1.2.3/4.5.6.70/8.9.
↪10.100.dcm?X-Amz-Credential=...&X-Amz-Signature=..."
    },
    {
      "seriesInstanceUid": "4.5.6.71",
      "seriesDescription": "Body 5.0",
      "modality": "CT",
      "seriesNumber": "3",
      "sopClassUid": "1.2.840.10008.5.1.4.1.1.2",
      "transferSyntaxUid": "1.2.840.10008.1.2",
      "instanceNumber": "2",
      "sop": "8.9.10.100",
      "frameCount": 1,
      "dicomUrl": "https://some-server.s3.us-east-2.amazonaws.com/dicom/1.2.3/4.5.6.71/8.9.
↪10.101.dcm?X-Amz-Credential=...&X-Amz-Signature=..."
    }
  ]
}
```

JSON field	Required?	Meaning
instanceUid	Y	*(0020,000D) Study Instance UID*
studyDate	.	*(0008,0020) Study Date*
studyTime	.	*(0008,0030) Study Time*
studyReceivedDate	.	Implementation-specific: when the study has been received
studyId	.	*(0020,0010) Study ID*
studyDescription	.	*(0008,1030) Study Description*
accessionNumber	.	*(0008,0050) Accession Number*
referringPhysicianName	.	*(0008,0090) Referring Physician's Name*
modalitiesInStudy	.	Either *(0008,0061) Modalities In Study*, or unique values of *(0008,0061) Modality* from all objects
sourceAETitle	.	Implementation-specific: AE Title of the sender
patientId	.	*(0010,0020) Patient ID*
patientName	.	*(0010,0010) Patient's Name*
patientBirthDate	.	*(0010,0030) Patient's Birth Date*
sops	Y	List of objects (at least one) in the study
seriesInstanceUid	Y	*(0020,000E) Series Instance UID*
seriesDescription	.	*(0008,103E) Series Description*
modality	.	*(0008,0060) Modality*
seriesNumber	Y	*(0020,0011) Series Number*
sopClassUid	Y	*(0008,0016) SOP Class UID*
transferSyntaxUid	Y	*(0002,0010) Transfer Syntax UID*
instanceNumber	Y	*(0020,0013) Instance Number*
sop	Y	*(0008,0018) SOP Instance UID*
frameCount	.	*(0028,0008) Number Of Frames*
dicomUrl	Y	URL for downloading the entire DICOM file

A reduced response is expected by the plugin's method findStudies (scenarios related to studiesServiceUrl). Basically, the "sops" object is not required and would needlessly waste network/server resources.

3.3.3.5. Specification of the "TELEPAXX" manifest for findStudy

```
{
  "patient": {
    "fullName": "John^Doe^^",
    "patientId": "111111",
    "sex": "M",
    "birthdate": "1945-02-25",
    "study": {
      "instanceUid": "1.2.276.0.7230010.3.33065",
      "description": "CT Schaedel",
      "dateTime": "2006-04-27T07:03:21Z",
      "accessionNumber": "33065",
      "studyId": "50763",
      "series": [
        {
          "instanceUid": "1.2.392.200036.9116.2.1.5.14468257.50762.1001",
          "modality": "CT",
          "description": null,
          "number": 1,
          "images": [
            {
              "filename": "http://some-server:8080/path/to/dicom/1111111",
              "instanceUid": "1.2.392.200036.9116.2.1.5.14468257.50762.1001.1",
```

(continues on next page)

(continued from previous page)

```
            "classUid": "1.2.840.10008.5.1.4.1.1.2",
            "sequenceNumber": 1,
            "numberOfFrames": 0,
            "transferSyntax": null,
            "bearerTokenNeeded": true
          }
        ]
      },
      {
        "instanceUid": "1.2.392.200036.9116.2.1.5.14468257.50763.1001",
        "modality": "CT",
        "description": null,
        "number": 2,
        "images": [
          {
            "filename": "http://some-server:8080/path/to/dicom/1111112",
            "instanceUid": "1.2.392.200036.9116.2.1.5.14468257.50763.1001.1",
            "classUid": "1.2.840.10008.5.1.4.1.1.2",
            "sequenceNumber": 1,
            "numberOfFrames": 0,
            "transferSyntax": null,
            "bearerTokenNeeded": true
          }
        ]
      }
    ]
  }
}
```

JSON field	Required?	Meaning
patient	Y	Patient-level attributes and lower-level objects
fullName	.	*(0010,0010) Patient's Name*
patientId	.	*(0010,0020) Patient ID*
sex	.	*(0010,0040) Patient Sex*
birthdate	.	*(0010,0030) Patient's Birth Date*
study	Y	Study-level attributes and lower-level objects
study.instanceUid	.	*(0020,000D) Study Instance UID*
study.description	.	*(0008,1030) Study Description*
dateTime	.	*(0008,0020) Study Date* and *(0008,0030) Study Time*
accessionNumber	.	*(0008,0050) Accession Number*
studyId	.	*(0020,0010) Study ID*
series	.	Series-level attributes and lower-level objects
series.instanceUid	.	*(0020,000E) Series Instance UID*
modality	.	*(0008,0061) Modality*
series.description	.	*(0008,103E) Series Description*
number	.	*(0020,0011) Series Number*
images	.	Image-level attributes
filename	.	Address of a DICOM file on Amazon S3
images.instanceUid	.	*(0008,0018) SOP Instance UID*
classUid	.	*(0008,0016) SOP Class UID*
sequenceNumber	.	*(0020,0013) Instance Number*
numberOfFrames	.	*(0028,0008) Number Of Frames*
transferSyntax	.	*(0002,0010) Transfer Syntax UID*
bearerTokenNeeded	.	If `true`, then fileCustomHeader is added to the request

3.3.3.6. Specification of the "TELEPAXX" manifest for findStudies

This reduced response is expected in scenarios related to studiesServiceUrl.

```
{
  "patient": {
    "fullName": "Wilson^Robert",
    "patientId": "1.2.826.0.1.3680043.8.498.42786523069543372024391294925156202159",
    "sex": "M",
    "birthdate": "1981-04-20",
    "study": {
      "instanceUid": "2.25.76535811009362561901722021937332161972",
      "description": "CT Head",
      "dateTime": "2004-12-15T18:01:56Z",
      "accessionNumber": "A864710",
      "modalitiesInStudy": ["CT", "DOC"]
    }
  }
}
```

JSON field	Required?	Meaning
patient	Y	Patient-level attributes and lower-level objects
fullName	.	*(0010,0010) Patient's Name*
patientId	.	*(0010,0020) Patient ID*
sex	.	*(0010,0040) Patient Sex*
birthdate	.	*(0010,0030) Patient's Birth Date*
study	Y	Study-level attributes
instanceUid	.	*(0020,000D) Study Instance UID*
description	.	*(0008,1030) Study Description*
dateTime	.	*(0008,0020) Study Date* and *(0008,0030) Study Time*
accessionNumber	.	*(0008,0050) Accession Number*
modalitiesInStudy	.	*(0008,0061) Modalities In Study* (distinct values of *(0008,0061) Modality* in entire study)

3.3.4. AWS HealthImaging

As of 2025, some server-side functionality of the AWS HealthImaging is still being developed. The recommended mode of integration is to use a pair of a dedicated plugin and the Dicomweb plugin.

3.3.4.1. AWS HealthImaging notes/limitations

Our long-term goal for this integration is the Dicomweb plugin. However, at the time of this writing (Jul 2025), the platform does not promise returning and indicating non-primary image sets via QIDO-RS, which by itself would be an extension of the standard. A dedicated plugin provides SearchAPI only. Example of the combined configuration:

```
com.softneta.meddream.pacs.configurations[0].type=AWSHealthImaging
com.softneta.meddream.pacs.configurations[0].id=aws_hi
com.softneta.meddream.pacs.configurations[0].datastoreId=6fa927e85331bdf3a0e9d0eb849f45c0
com.softneta.meddream.pacs.configurations[0].awsRegion=us-east-1
com.softneta.meddream.pacs.configurations[1].type=Dicomweb
com.softneta.meddream.pacs.configurations[1].id=aws_hi
com.softneta.meddream.pacs.configurations[1].searchApiEnabled=false
com.softneta.meddream.pacs.configurations[1].dicomFileUrl=https://dicom-medical-imaging.us-
```

(continues on next page)

(continued from previous page)

```
↪east-1.amazonaws.com/datastore/6fa927e85331bdf3a0e9d0eb849f45c0/studies/{study}/series/
↪{series}/instances/{image}
com.softneta.meddream.pacs.configurations[1].fileAcceptHeader=application/dicom; transfer-
↪syntax=1.2.840.10008.1.2.1
com.softneta.meddream.pacs.configurations[1].authType=aws
com.softneta.meddream.pacs.configurations[1].awsRegion=us-east-1
com.softneta.meddream.pacs.configurations[1].dicomCacheDirectory=${com.softneta.meddream.
↪tempDir}/dcm/aws
```

There is no sense to use searchApiEnabled=false for this plugin and the default value is already `true`. In turn, the plugin does not support other related settings "imageApiEnabled" etc. (Image API is not implemented at all) and will report `false` to Core in order to minimize confusion.

The plugin uses the default authentication scenario from AWS SDK. Probably the simplest way to pass credentials is to use the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.

Name of every plugin setting is case-insensitive, therefore one can pass a plugin setting via environment variables, for example, COM_SOFTNETA_MEDDREAM_PACS_CONFIGURATIONS_0_AWS_REGION = us-east-1.

3.3.4.2. MedDream configuration for AWS HealthImaging

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure a universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to AWS HealthImaging plugin. The documentation below is for plugin version 1.3.11. Do not forget to add the Dicomweb plugin with the same .id and .searchApiEnabled=false, or else Core won't start.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=AWSHealthImaging`

     Use this specific value of `type`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=aws_hi`

     Identifies the plugin configuration. You can use multiple datastores etc. by using different values of `id` and remaining options. Corresponding instances of the Dicomweb plugin are still required.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

     (optional) The value `false` disables support for "Clear caches" functionality. Although this plugin doesn't download DICOM files, it still can cache structure-related server responses for better performance, and "Clear caches" is able to remove them.

     Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].prepareFromPacsMetadata=true`

     (optional) If true, Core will call a new plugin method intended for fetching as many instance-level attributes as possible, and will prepare instance metadata from it (not from the DICOM file as usual).

     Default is `false`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].datastoreId=6fa927e85331bdf3a0e9d0eb849f45c0`

  The identifier of the data store where the image sets reside.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].awsRegion=us-east_1`

  Explicitly specifies the AWS region.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDirectory=${com.softneta.meddream.tempDir}/dcm/aws`

  (optional) A cache directory for server responses. If not configured, or is missing on disk and its creation has failed, then no caching is performed, therefore performance might be worse. On the other hand, if you use the cache, then you need to set up a separate entry under `com.softneta.temp-cleaner.tempItems[]` (see *Temporary files cleaner*).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].directAHIforPixels=true`

  (optional, experimental) If `true`, the study structure will include data needed by the frontend for fetching pixels and thumbnails directly from AWS (work in progress). Default is `false`.

  Currently, this "direct" access involves requests to non-existent endpoints in Core; these requests must be routed to AWS via a reverse proxy.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].alternateURITemplate= /store/{datastoreId}/imageSet/{imageSetId}/frameId/{{frameId}}/region/{regionId}`

  (optional) Configures a URL for directAHIforPixels=true (its equivalent was hardcoded before MedDream 8.8.0). The default value is shown in the example.

  Placeholders with {single} brackets are filled by the plugin, while the ones with {{double}} brackets are filled by the frontend. Only these 4 names in the example are supported.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].excludeNonPrimaryImageSets=true`

  (optional) If `true` (default), the study structure won't include any non-primary image sets. Those have conflicting data elements, like a different Series Number together with the same Series Instance UID, or different patient demographics.

  The value `false` is incompatible with retrieval of images via the Dicomweb plugin, as HealthImaging only returns non-primary image sets over WADO-RS when an image set ID is specified in the request. This ID is not specified by WADO-RS and not present in MedDream document model, where a study/series/image UID triplet is sufficient to identify a DICOM resource.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].maxStringLengthOfJsonField=25000000`

  (optional) Adjust a certain limitation of the Jackson library if opening a study fails with the error "String value length . . . exceeds the maximum allowed".

  As of MedDream 8.9.0+, the default value of this setting is 25 000 000, while the library-internal default is 20 000 000.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for chosen method.

QIDO-RS support

The following parameters and their implementation was copied from the Dicomweb plugin, with minor adjustments.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].qidoRsUrl=https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/6fa927e85331bdf3a0e9d0eb849f45c0`

  (optional) If configured, the plugin will use QIDO-RS instead of the traditional HealthImaging API. Empty by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchPageSize=1000`

  (optional) Override "limit" parameter for qidoRsUrl. 50 by default, must not exceed 1000 because other parts of MedDream are unable to handle more.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=120`

  (optional) Socket connect and read timeout in seconds. Zero (default) means a system-dependent default that is applied by the JVM as before.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].dicomPersonName=false`

  (optional) true (default): patient and referring physician names are returned with DICOM component/group separators already supported in Core. If false, the separators are converted to spaces.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].nonStrictSearchPattern=00100020:{value}*;00080050:{value}`

  (optional) Specifies where to add the wildcard characters in a non-strict search context. Applies to Patient ID and Accession Number. Use `{value}` as the placeholder for the actual search key. The example above forces the "begins with" and "equals" matching rules instead of "contains".

  Default value is `00100020:*{value}*;00080050:*{value}*` – with a "contains" rule in both cases.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchSymbols=*,?`

  (optional) Comma-separated list of characters that are used when partial search filter is enabled via supportsPartialSearchFilter=true. The end user can include them explicitly in the search key, in which case nonStrictSearchPattern is not applied.

  When a partial search filter is disabled (supportsPartialSearchFilter=false), those characters are not allowed in end user's input and will yield an error message.

  The value in the example is the default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=true`

  (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` (default) forces wildcard matching by those attributes everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window.

  Unlike in other plugins, empty/null is not a default, and this third state is not supported.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

(optional) `true` enables support for monitoring study structure changes via structureListenerConfig in system.json. Requires non-empty studyCountersUrl.

Disabled by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyCountersUrl=https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/6fa927e85331bdf3a0e9d0eb849f45c0/studies?0020000D={study}&includefield=00201206&includefield=00201208`

  (optional) A lightweight query for fetching only two DICOM attributes, NumberOfStudyRelatedSeries and NumberOfStudyRelatedInstances.

  Empty by default.

3.4. DICOM mode

MedDream is able to communicate over DICOM 3.0 native interface (Upper Layer Protocol, ULP) with any PACS that supports Query/Retrieve and Storage.

3.4.1. DICOM mode notes

The Query/Retrieve SCU identifies itself as "MEDDREAM" by default. (Do not forget to configure the PACS to accept connections from this AET.) This can be changed by the `localAET` setting of the plugin. Binding to a particular port is usually not required, and Java-based MedDream still doesn't support that.

The plugin also contains its own Store SCU that is used to send annotations and screenshots back to the PACS. (With other plugins, a common DICOM SCU in Java Core is used, see 3.1.2 Saving measurements, Key Objects, segmentations and screenshots.) If you choose this, then:

- the plugin-level settings `storeScpIp`, `storeScpPort` and `storeScpAet` (for the common DICOM SCU) are not needed – the plugin uses the same `remoteHost`, `remotePort` and `remoteAET` as for Query/Retrieve. `storeScpAet` might still be needed for in-advance preparation (see 5.1.18 Caching studies via DICOM Listener);
- its Calling AE Title is configured by the `localAET` plugin setting, instead of `storeScuAet` or `com.softneta.meddream.dcmsnd.bind`;
- uploading via the plugin also requires its setting storageApiEnabled=true.

Usage of same `remoteAET` for destination makes it incompatible with PACSes where different AETs are configured for Query/Retrieve SCP and for Storage SCP. Meanwhile the common SCU in Core uses a separate set of parameters for every plugin, therefore the destination can not only have a different AET but also a different address and port, or even the same address and port for all plugins.

The local Store SCP (receives files during a C-MOVE session) is implemented not in the plugin but in Java Core, and can function in parallel to any other plugin if needed. It **must** bind to a port (we recommend 11116) and by default listens on 127.0.0.1 (`com.softneta.dicomStoreService.address` can specify a single address or 0.0.0.0, a wildcard for any available IP address). The mandatory settings are:

- `com.softneta.dicomStoreService.localAETitle` – accepted Called AE Title (also multiple comma-separated values, or even empty),
- `com.softneta.dicomStoreService.port`,
- `com.softneta.dicomStoreService.acceptAETitles` – known Calling AE Titles,
- `com.softneta.dicomStoreService.saveDirectory` – base directory for received files.

3.4.2. MedDream configuration for DICOM mode

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Choose and configure some universal authentication method according to 5.1.6 Built-in authentication and authorization.
3. Edit `application.properties` file and update the options related to Query/Retrieve plugin. The documentation below is for plugin version 6.7.2.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=QR`

     Use this specific value of `type` when connecting to a PACS over the DICOM 3.0 protocol.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=AnonStudies`

     Identifies the plugin/configuration pair in the search window drop-down menu.

     An alphanumeric string (Latin lowercase/uppercase letters, digits, underscore) is expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair returns a study list for the Search window and provides study metadata. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].imageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair retrieves image data. Default is `true`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].eventApiEnabled=false`

     (optional) `true` by default. If set to true (or commented out) together with `useSessionCache=true` and `com.softneta.preparation.useSessionCache=true`, the per-session cache is cleared automatically when session ends.

     If set to true (or commented out), support for Clear Cache at the plugin side is enabled. In case of "useSessionCache" settings mentioned above, will clean the user-specified part of both per-session and global caches.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].storageApiEnabled=true`

     (optional) Specifies if this plugin/configuration pair is used to send PR/KO/segmentation/screenshot objects back to its PACS (instead of using the mechanism common to all plugins). Default is `false`.

     You can try this option when there are some problems with PR/KO sending, as the Store SCU in the plugin is a bit different. The parameters storeScpAet, storeScpIp, storeScpPort and storeScuAet are then not used.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteAET=PACSONE`

     The AE Title of the remote machine.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remoteHost=127.0.0.1`

     IP address or hostname of the remote machine.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].remotePort=104`

     Port number of the remote machine.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].localAET=MEDDREAM`

  The AE Title of the DICOM C-FIND/C-MOVE/C-STORE clients implemented in this plugin. Also used as the C-MOVE destination.

  Multiple values (separated by comma) can be entered if the local Store SCP is configured with multiple Called AE Titles. This might improve performance due to parallel sessions.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cFindLocalAET=MDR`

  (optional) If not empty, overrides `localAET` for C-FIND sessions (C-MOVE sessions always remain unchanged).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDir=${com.softneta.dicomStoreService.saveDirectory}/PACSONE/`

  A directory with images that are received by the *local C-STORE SCP* in the background. **The trailing slash/backslash is mandatory.**

  The last subdirectory must be equal to `remoteAET`, as the local SCP adds one automatically to the configured base directory.

  This plugin supports URIs (starting with `file://`). Making them relative to receiver's saveDirectory is encouraged to minimize human errors.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=90000`

  (optional) DICOM file receive timeout in milliseconds (starting from the C-MOVE request).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].retryDelay=1000`

  (optional) DICOM file existence check period in milliseconds.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].strictSearchIsEnabled=true`

  (optional) `true` forces exact matching by Patient ID and Accession Number everywhere, including the Search window. `false` forces wildcard matching everywhere, including HIS integration by Patient ID or Accession Number, and the Patient History window. `null` or unset is the default: exact matching for Patient History and HIS integration, wildcard matching for Search window.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].multipleModalitySupport=true`

  (optional) `false` will yield an error message when the user selects multiple modalities. Use this if the PACS does not support multiple modalities and search results confuse the users.

  If explicitly configured with `true`, the plugin performs a workaround for a certain PACS: a filter by modality "ECG" will also match a non-standard modality "ECGEP". Keep the option commented out if this is undesired.

  When this option is commented out (default), both the error message and the modality workaround are turned off.

  8.5.0+ contains a more user-friendly alternative when a single plugin is used: "Single modality support" in the Settings window (under Search).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true`

  (optional) `true` enables support for monitoring study structure changes (see structureListenerConfig in system.json). Disabled by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].defaultStartDate=20181107`

(optional) Default "from" date in `YYYYMMDD` format when empty in the user interface. This is for some PACSes that require the Study Date search key in every C-FIND request.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].defaultEndDate=20181108`

  (optional) Default "to" date in `YYYYMMDD` format when empty in the user interface. This is for some PACSes that require the Study Date search key in every C-FIND request.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].studyLevelRequest=false`

  (optional) Value for query attribute (0008,0052) Query/Retrieve Level: `STUDY` if true (default), `SERIES` if false.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchCharset=ISO_IR 144`

  (optional) Convert search keys from UTF-8 to this encoding. Default is empty (no conversion).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].structureResultCharset=ISO_IR 144`

  (optional) If the PACS didn't specify result encoding, then convert search results from this encoding to UTF-8. Default is empty (no conversion).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useSessionCache=true`

  (optional) Enables session-scope caching. When true, every session gets its own cache subdirectory (the plugin moves, or copies, globally received files to session-related subdirectories). It is forcefully cleaned when the session ends (cleaning also requires `eventApiEnabled` and `com.softneta.preparation.useSessionCache`).

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.requestSeriesFirstInstances=0`

  (optional) If nonzero, performs an image-level C-MOVE for this number of objects at beginning of each series, and then proceeds with remaining images via either study- or series-level C-MOVE.

  This might help to display the thumbnails faster. For example, when MedDream displays only the first image of every CT series, then cmove.requestSeriesFirstInstances=1 ensures that the thumbnail of series #2 doesn't need to wait until the entire series #1 is received, and so on.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.requestAllSeriesIfInstanceCountLowerThen=0`

  (optional) If a series contains less objects than specified here, then it is not downloaded partially (via an instance-level C-MOVE); it is instead downloaded completely, via a series-level C-MOVE.

  The value must be equal to, or larger than, cmove.requestSeriesFirstInstances. Otherwise, it will be increased accordingly.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.requestAfterStructure=true`

  (optional) If true, a C-MOVE of the entire study or every series begins in background automatically when the study structure is being returned to the frontend. This will work even if imageApiEnabled=false. A side effect is that Export/ZIP will always trigger a refresh of the study.

  False by default.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].requestStructureInEveryLevel=1000`

  (optional) When to collect the study structure incrementally (particularly, instance attributes are enumerated separately for each series). The default attempt to fetch attributes from multiple levels might cause some PACSes to hit a hardcoded number of query results and the only partial workaround is to enumerate instances from a single series at once. This will, of course, not help if even a single series is larger than the limit; `requestStructureInstanceLevelResultLimit` might then help further.

  An integer means retrying in incremental mode after finding out that the study structure contains this many images. This special number is to be found experimentally: every large study then looks "truncated" at the same size. (The true size can be found by starting study-level C-MOVE from a third-party tool and counting the number of received files.)

  `true` forces the incremental mode regardless of study size. This decision should be based on performance measurements as default behavior might be more efficient with some PACSes.

  Default is zero and means "never".

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].requestStructureInstanceLevelResultLimit=1000`

  (optional) Refines behavior of `requestStructureInEveryLevel`. If any query in incremental mode again hits this limit (not necessarily the same as specified by requestStructureInEveryLevel), then values of the Instance Number attribute are analyzed for gaps and missing ones are enumerated via queries with a filter on Study Instance UID / Series Instance UID / Instance Number. A gap is assumed at the beginning if the minimum Instance Number is not 1. Another gap is always assumed at the end, starting from the maximum Instance Number plus one. Every gap is queried by increasing values of Instance Number, and this terminates prematurely on empty result.

  Default is zero and disables this additional filter on Instance Number.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.filterModalities=CT,SR`

  (optional) Alternative to `cmove.requestSeriesFirstInstances`: a comma-separated list of Modality values that identify important series to be C-MOVE'd before the remaining ones. Requires cmove.requestAfterStructure=true, cmove.requestAfterStructureLevel=SERIES and nonzero cmove.filterModalitiesIfStudyHasMoreInstance.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.filterModalitiesIfStudyHasMoreInstance=500`

  (optional) The cmove.filterModalities logic is enabled only when number of instances in the study exceeds this amount.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cmove.requestAfterStructureLevel=SERIES`

  (optional) If "SERIES", fetches each of the missing or incomplete series separately (performs a series-level C-MOVE). "STUDY" will fetch the entire study at once.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchEscapeSpaceSymbol=*`

  (optional) Spaces in the search key (for example, entered by the user in the Search window) will be replaced with this value.

  The default value corresponds to pre-8.4 behavior: it had a hardcoded step of replacing spaces with asterisks. Now it is possible, for example, to replace spaces in patient name with `^`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].nonStrictSearchPattern=00100020:*{value}*;00080050:*{value}*;00100010:*{value}*;00081030:*{value}*`

(optional) Specifies where to put wildcard characters when implementing a non-strict search. The value in the example is used by default, and yields the same behavior as before. Other tags you can use are 00100040 and the same value as in . . . columns.sourceAE.tag.

Entries are separated with `;`. Format of the entry is `<tag code>:<value pattern>`. The `<value pattern>` part must contain a placeholder `{value}` that will be replaced with the search key; furthermore, space characters in the key are replaced with searchEscapeSpaceSymbol. Currently recognized values for `<tag code>` are `00100020`, `00080050`, `00100010`, `00081030`.

This would match only values beginning with the search key: `00100020:{value}*; 00080050:{value}*`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].searchSymbols=*,?`

  (optional) Comma-separated list of wildcard characters. They aren't allowed in strict search scenarios, and will yield an error message. Applies to the same set of DICOM Attributes as nonStrictSearchPattern: Patient ID, Accession Number, Patient Name, Study Description.

  The default value with two characters from the Standard is a security improvement: it makes sure that the PACS won't return studies to which the end user has no access. An empty string means pre-8.0 behavior: if the user-supplied search key contains an asterisk, then the PACS might return too many results.

  In non-strict scenarios, including strictSearchIsEnabled=false or keys like Study Description, these characters are used to decide that the search key is good enough — namely, it doesn't need formatting by `nonStrictSearchPattern` and spaces aren't to be replaced with searchSpaceEscapeSymbol. When the end user is using the Search window, (s)he can explicitly add wildcards where deemed appropriate: for example, `PID001` is sent to the PACS as `*PID001*`, `PID 001` – `*PID*001*`, `PID001*` – `PID001*`, `PID?001` – `PID?001`.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].allowDuplicateObjects=NO`

  (optional) Handling of objects with the same SOP Instance UID found in same series. Objects like these indicate a serious flaw in the PACS, especially if they have different attributes: you are no more sure which version of the object will be sent to MedDream later when requested by the same SOP Instance UID (attributes in the structure might be misleading).

  NO (default) disallows the duplicates and the study doesn't open.

  KEEP ignores any duplicates, therefore attributes of the first occurrence are preserved. If the PACS wants to return all versions of the object and they are properly sorted **in descending order by reception time**, then this is the safe value.

  REPLACE updates the attributes of the earlier entry. If the PACS wants to return all versions of the object and they are properly sorted **in ascending order by reception time**, then this is the safe value.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.tag=1048624`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.type=string`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.level=STUDY`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.<name>.strict=false`

  (optional) Remapping of attributes sent to the PACS as query/return keys and collected from the response. Also see 5.1.21 Custom columns in Patient History / Patient Studies for the overall idea.

Fully tested values of <name> for this plugin are `sourceAe`, `receivedDate`, `report`, `custom1 ... custom10`. More will be documented in the future.

.tag specifies the tag of a DICOM attribute – an integer in decimal notation. The columns "sourceAe" (was configured via sourceAeTitleTag before MedDream 8.2) and "receivedDate" (was configured via studyReceivedDateTag) are ignored by default as their .tag is zero. The column `report` is a new functionality for read-only reports implemented by the customer and also ignored by default.

.type specifies additional processing of the value. `object` (default) means none. `number` is converted to Integer or Double **and works only for names custom1...custom10 (combination with other names is an error)**. For `string`, missing value is replaced with "". For `date`, a date formatted as yyyyMMdd is converted to yyyy-MM-dd. For `time`, a time formatted as HHmmss is converted to HH:mm:ss. `personName`, when dicomPersonName=false, replaces every "^" with a space. You can safely use `string` with the columns "sourceAe", "receivedDate" and "report" mentioned above.

.level specifies where to request the attribute and expect it in the response. Default is STUDY, can be changed to SERIES or IMAGE.

.strict (false by default) can disable adding of wildcards by default when this attribute is used as a search key. The overall scenario and strictSearchIsEnabled can still override it. See also nonStrictSearchPattern, searchSymbols.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].reportURL=http://127.0.0.1:8088/report/{studyUid}`

  (optional) A URL (or file system path) template for accessing customer-implemented reports (see 5.1.8.4 Remote reports for details). A placeholder `{studyUid}` is replaced with Study Instance UID, and `{storageId}` – with .id of this plugin configuration.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].maxConnections=5`

  (optional) Limit on C-MOVE sessions from a particular plugin instance (a single "storage"), shared between all concurrent user connections. 1000 by default.

  Does not affect C-FIND sessions so far; their number will depend on concurrent users as before.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].blacklistedSopClasses=1.2.840.113619.4.30`

  (optional) Comma-separated list of SOP Class UID values (empty by default) to be filtered out from the study structure. Similar to "blackListedSopClasses" in system.json but at a lower level: even PR, KO, etc can be completely hidden from MedDream this way.

  Objects with a matching attribute might also avoid a C-MOVE to MedDream. However, if not all images from a series were blacklisted but the plugin decides to do a series-level C-MOVE, or not all series were blacklisted but the plugin decides to do a study-level C-MOVE, then the PACS will still attempt to send the object among the "good" ones.

  The main purpose of this setting is to avoid C-MOVE sessions of objects (or series, or even studies) consisting entirely of listed objects. For example, a sub-association related only to private SOP Classes unsupported by the local C-STORE SCP usually fails with an error that is reported back to MedDream and interferes with loading of the study. In contrast, if that association also contains supported objects, transfer of the latter succeeds with just a warning which MedDream ignores.

4. Restart Core for changes to take effect, and navigate to `http://127.0.0.1/`. Use login credentials that are valid for configured method.

3.5. File system access mode

This MedDream mode allows you to view single DICOM files, or directories containing DICOM files (of a single study or multiple studies). In fact, no PACS is necessary.

However, there is no search functionality. **A HIS (or equivalent application) must track studies/images together with corresponding paths and offer hyperlinks to MedDream.**

3.5.1. File system mode notes

MedDream is opened with a relative path to a single file, or to a directory. Absolute paths are not allowed due to security concerns.

If a directory is specified, then DICOM files are collected directly below it by default. Number of scanned subdirectories can be increased by the `maxDepth` plugin option.

> **Warning:** As of 7.7.0+, there is only a limited support for copies of the same DICOM file (same Study/Series/SOP Instance UIDs) in different directories. These directories must be accessed via separate sessions with different /?file=... values (or their token-based equivalents); a single session must use different multiple entries in the "file" parameter. Accessing at once via a common value, like a parent directory, is not possible. The most obvious situation is where all files are kept together without subdirectories, and at least one file is duplicate; duplicates will then be ignored, warnings will only be seen in the log, it might be counter-intuitive which file is parsed first and therefore treated as a reference for finding duplicates.
>
> The Reporting function is not recommended at all: it will treat such copies as different studies, their reports might become mixed up after restarting the backend and reopening the studies again. Saving of key objects and measurements might be problematic, too.

The specified directory must contain DICOM files only, at least for the default ignoreUnparseable=false. Filtering by name pattern or extension is not supported.

- With the default useDicomdirForStruct=false, typical "DICOMDIR" files (the Basic Directory IOD, SOP Class "1.2.840.10008.1.3.10") will be ignored if encountered during a directory scan, and are specifically disallowed as a target of the "file" parameter.
- useDicomdirForStruct=true allows you to use explicitly specified DICOMDIR files as an alternative to directory scanning; the DICOM files won't be accessed at all when building the study structure. DICOMDIRs will still be ignored during a directory scan.
- With ignoreUnparseable=true, files that are unparseable or unreadable during a directory scan, can be ignored but the application log will contain more rubbish.

This plugin supports a custom HIS integration parameter "file". Due to architectural shortcuts, it also supports the ubiquitous parameter "study". However, the latter is not intended for HIS integration; if used this way, it will work only for a short time (until a particular in-memory cache expires) after successfully opening the corresponding studies via "file".

Do not set `searchApiEnabled` or `imageApiEnabled` in plugin configuration to `false`, as Image API of this plugin depends on data cached by its Search API. This plugin can't be combined with others.

The login form is not needed to view images and can therefore remain disabled together with the Search window. However, both features are needed to access the Settings dialog when the USER_SETTINGS permission is not suitable. For that one must log in interactively as a user with the "ADMIN" permission.

The error message "The DICOM file has not been parsed yet" often means that the file, or its reference in DICOMDIR, was evicted from the in-memory cache due to insufficient space there – you need a larger cacheMaxSize setting. Another related setting, cacheMaxAgeSec, is a rarer reason. When opening a study from DICOMDIR (MedDream 8.8.0+), this message might also mean that a file referenced by DICOMDIR is missing or unreadable; the real reason is logged at ERROR level.

3.5.2. MedDream configuration for File system mode

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Configure the HIS integration in `application.properties` file:

```
spring.profiles.include=auth-his
authentication.his.validHisParams=file
authentication.his.maxObjects=10
authorization.defaultHisPermissions=EXPORT_ISO,EXPORT_ARCH,FORWARD,\
    UPLOAD_DICOM_LIBRARY,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
```

   The "maxObjects" setting allows you to limit the number of studies that is allowed to open at once. If the directory specified in /?file=... contains more, the operation will fail. This prevents situations where too many objects would render the frontend unusable.

3. **Plan how MedDream settings will be changed:**
   - **(Option 1)** Choose and additionally configure some universal authentication method according to 5.1.6 Built-in authentication and authorization, then add both "ADMIN" and "SEARCH" permissions to some users. These users will be able to log in, ignore zero results in the Search window, then go to the Settings window. There is little sense in having users without the "ADMIN" permission, due to absent studies in the Search window.
   - **(Option 2)** Add the "ADMIN" permission to `authorization.defaultHisPermissions`. Remember that all users will then be able to change MedDream settings and might thus interfere with each other's work. As a short-time measure, however, this option might still be more attractive than the first one.
   - **(Option 3)** Add the "USER_SETTINGS" permission to `authorization.defaultHisPermissions`. Afterwards every user will be able to change his own copy of the settings file, and will need administrator's help to get the common settings back.
4. Edit `application.properties` file and update the options related to the plugin "FileSystem". The documentation below is for plugin version 7.3.0.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=FileSystem`

     Always set this to `FileSystem`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=File-System-Plugin`

     Any alphanumeric string that identifies this plugin instance.

     Latin lowercase/uppercase letters, digits, underscore are expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].rootDirectory=C:\\dicoms\\files`

     Full path to the base directory below which the DICOM files can be found.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].rootDirectoryOptional=true`

     (optional) When false (default), rootDirectory must be configured and /?file=... must contain relative paths (an absolute path will result in an error).

     When true, pre-8.6 behavior is back: absolute paths in /?file=... are used as is, while relative ones are based on rootDirectory. It is also possible to leave rootDirectory empty or unconfigured, which means relative paths will be based on the current directory of the Core process (normally the same as com.softneta.meddream.configRoot).
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].maxDepth=1`

(optional) The number of subdirectories traversed when collecting files. Minimum allowed value and default value is 0, which means the files will be collected in the directory specified by /?file=... but not deeper.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].ignoreUnparseable=true`

  (optional) Default (false): any unreadable or unparseable DICOM file fails the entire operation (study structure is not returned).

  True: the problem is only logged, the file isn't added to the study. This applies only to files found during a directory scan, not to a single file referenced by ?file=... or an equivalent entry in the token.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheMaxSize=1000`

  (optional) Maximum number of studies in the in-memory cache. Minimum allowed value is 1, default is 1000.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheMaxAgeSec=7200`

  (optional) Maximum age, in seconds, of study records in the in-memory cache. Minimum allowed value is 1, default is 7200 (2 hours).

  The actual value might appear higher as record expiration is triggered by a write activity. If new studies aren't opened, then memory usage will stay unchanged.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].timeout=10000`
- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].retryDelay=1000`

  (optional) How long to wait for a DICOM file to be available in case of multiple concurrent sessions with a non-unique "file" HIS integration parameter.

  The error "The DICOM file has not been parsed yet" is still possible even if cacheMaxSize and cacheMaxAgeSec are large enough. Cached data obtained from a particular path is removed before parsing that path anew, so other sessions trying to parse the same path in parallel might not find this data in the cache and will yield that error. As a workaround, MedDream now waits for up to `timeout` milliseconds, in `retryDelay` increments, for a corresponding cache entry (or a file referenced by it) to become available. **The best prevention is to always use a unique "file" parameter, even in the same session.**

  Default values are in the example. Zeroes, or very small values like 10, will effectively disable this waiting and return pre-8.8 behavior; you should try them if 8.8 has introduced unexplainable pauses with this plugin.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].useDicomdirForStruct=true`

  (optional) If true, you can specify a DICOMDIR file via /?file=... or its equivalent in the token, and build the study structure from it, which is faster.

  The default behavior (false) is slower not only because of a recursive directory scan: every found DICOM file is fully parsed and loaded into memory, except the PixelData attribute. When using a DICOMDIR as a "manifest", only this file is fully parsed and loaded.

  Note that MedDream requires (0008,103E) Series Description, and especially (0028,0008) Number Of Frames, for proper operation. However, these attributes are not mandatory in a DICOMDIR file. If you're generating DICOMDIRs by your own means, then it will be easier to adjust the process.

5. Test the functionality by browsing to `http://127.0.0.1/?file=`*`PATH_TO_YOUR_FILE`*

   For example, the directory `C:\PACS\dcm4chee\server\default\archive` contains DCM4CHEE 2.x archive tree, with deeper levels named like `year\month\day\hour\...`. This directory

is specified by `rootDirectory`. Then *PATH_TO_YOUR_FILE* could be `2012/2/4/0/43D7AA94/2569DF62/9242C40A`.

3.6. Non-DICOM file system mode

The "FileSystem-Non-Dicom" plugin allows you to view directories with non-DICOM files in a fashion similar to the traditional "filesystem" mode. A directory always results in a single study. Every file there appears as a series with a single viewable object. Patient demographics and other metadata must be provided in a dedicated JSON file.

There is no indexing, too, therefore the Search window cannot be used.

3.6.1. Non-DICOM mode notes

MedDream is opened with a full path to a directory. The legacy "unsafe" method via /?file=... is not supported so far, one must use token-enabled integration. Example token:

```
{
  "items": [
    {
      "studies": {
        "file" : "f:/tmp/study49405968726",
        "storage": "fs2"
      }
    }
  ]
}
```

Viewable objects are files with extensions "png", "bmp", "jpeg", "tiff", "pdf", "mp4", "txt". Files are sorted by name and are displayed in this order. Filtering by name pattern or extension is not supported. **Any subdirectories will crash the plugin.**

The metadata file must be named `study-patient.json`. Example contents:

```
{
  "date": "2021-10-04",
  "time": "14:36:30",
  "description": "Second opinion (scanned document)",
  "patientName": "John Doe",
  "patientId": "123",
  "sensitive": false,
  "restricted": false,
  "dob": "1975-04-12",
  "gender": "M"
}
```

The fields "sensitive" and "restricted" are reserved in MedDream 8.1.0+. The value "true" will cause specific behavior in later versions, and should not be used until then to avoid surprises.

The Modality attribute (visible in, for example, the Export window) is always set to "nonDicom". However, the Export window, Forward window and some other functions do not work with non-DICOM studies.

The plugin currently caches some data until application restart, for every unique value of the "file" parameter. Contrary to the FileSystem plugin, the same directory with updated content will not be rescanned when opening the study once more. You must use unique values of "file" every time (for example, generate the directory name randomly). As MedDream 8.5+ tracks file system changes below rootDirectory, small incremental changes of study content are possible in-place, without copying the entire file set elsewhere.

Do not set `searchApiEnabled` or `documentApiEnabled` in plugin configuration to `false`, as Image API of this plugin depends on data cached by its Search API. This plugin can't be combined with others.

Starting from MedDream 8.5.0, `eventApiEnabled` can't be set to false, too, because the plugin must get a particular event to release resources properly.

This plugin supports a custom HIS integration parameter "file". Due to architectural shortcuts, it also supports the ubiquitous parameter "study". However, the latter is not intended for HIS integration, its value is random and becomes unusable after restarting the JAR.

3.6.2. MedDream configuration for Non-DICOM file system mode

1. Perform steps in the 5.1.1 Essential configuration and the first run chapter.
2. Configure the HIS integration in `application.properties` file:

```
spring.profiles.include=auth-his
authentication.his.token-service-address=http://127.0.0.1:8082/v2/validate
#authorization.defaultHisPermissions=USER_SETTINGS
#authorization.defaultHisPermissions=ADMIN
```

3. **Plan how MedDream settings will be changed:**
   - **(Option 1)** Choose and additionally configure some universal authentication method according to 5.1.6 Built-in authentication and authorization, then add both "ADMIN" and "SEARCH" permissions to some users. These users will be able to log in, ignore zero results in the Search window, then go to the Settings window. There is little sense in having users without the "ADMIN" permission, due to absent studies in the Search window.
   - **(Option 2)** Add the "ADMIN" permission to `authorization.defaultHisPermissions`. Remember that all users will then be able to change MedDream settings and might thus interfere with each other's work. As a short-time measure, however, this option might still be more attractive than the first one.
   - **(Option 3)** Add the "USER_SETTINGS" permission to `authorization.defaultHisPermissions`. Afterwards every user will be able to change his own copy of the settings file, and will need administrator's help to get the common settings back.
4. Edit `application.properties` file and update the options related to the plugin "Filesystem-Non-Dicom". The documentation below is for plugin version 4.0.25.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].type=Filesystem-Non-Dicom`

     Always set this to `Filesystem-Non-Dicom`.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].id=NonDicom`

     Any alphanumeric string that identifies this plugin instance.

     Latin lowercase/uppercase letters, digits, underscore are expected. Plan in advance and avoid changing the value (even letter case) when the system has already been used for some time, as it is stored in report database and preparation cache.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].cacheDirectory=${com.softneta.meddream.tempDir}/nd`

     A directory for some temporary data.
   - Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].rootDirectory=/tmp/nd`

     A root directory for all paths to studies. Starting from MedDream 8.5.0, paths passed by the HIS must be relative to this setting and file system changes below this directory are

monitored, therefore you should use a dedicated directory instead of "/" or other location below which changes are unpredictable.

- Example: `com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].supportedMimeTypes=/etc/hisTypeWhitelist.lst`

  (optional) Path to a file that lists allowed MIME Type values. Files of other types will be silently ignored.

5. Generate a token with `"file":"{PATH_TO_YOUR_DIRECTORY}"`.
6. Test the functionality by browsing to `http://127.0.0.1/?token={YOUR_TOKEN}`.

4. Image Access from hospital information system (HIS)

It is recommended to **use the token-based integration mode**:

- a token validator service (its reference implementation is bundled with MedDream for integration into PacsOne web interface) accepts object identifiers and returns a generated token. More combinations of object identifiers and storage names are supported;
- MedDream HIS integration links include only a single parameter called "token";
- when MedDream loads, it obtains object identifiers (study, accession number, . . .) from the token service.

For more information, including specification of the token service, see a separate document "MedDream integration MANUAL" available at https://www.meddream.com/documentation/installation-integration-guide/integration-via-url/.

Note: 8.8+ can additionally validate the user's identity in the token: if a header `md-user-id` is present when opening the HIS integration link, then it must match the token's .user.id element.

MedDream can also be integrated via its Communication API – a JavaScript-based wrapper over the legacy/token URL method. See the Communication API specification; you can obtain it from support@meddream.com.

There is a third method where the image-displaying part of the frontend is added to your application as a React component, and you implement all buttons by yourself. See the Viewport API specification, also available from support@meddream.com.

In MedDream 7.6.0+, the legacy "unsafe" integration (an object identifier in the URL is sufficient to display it) supports a few filters.

- `http://localhost/?study=1.2.392.200036.9107.500.110113,1.2.840.114350.2.171.2.798268.2.567817820.1&storage=MainArchive,RouterPacs`

  Opens the specified study (or studies) at once. Multiple values can be separated with commas. The `storage` parameter (optional) restricts the search to specified PACS plugins; the comma-separated list must contain values of the `.id` parameter in plugin configuration.

- `http://localhost/?accnum=0000000005`

  Opens **all studies** resolved by this single Accession Number **at once in the Viewer**. No multiple values.

- `http://localhost/?patient=TT054270`

  Opens **all studies** of this single patient either **in the Patient Studies window for subsequent choice** (default), or at once in the Viewer. No multiple values.

There is a risk that the number of studies is too large – for example, "0" is often assigned to Patient ID during anonymization and would therefore match a lot of studies. Such an operation will fail due to exceeded `authentication.his.maxObjects`. This limit applies both to the Patient Studies window and to opening multiple resulting studies immediately.

The setting `authentication.his.patient-integration-open-studies=false` disables the Patient Studies window. After that all resulting studies will open in Viewer immediately.

Currently there is no support for the legacy integration by Patient ID that used to result in a filtered Search window. The Patient Studies window is quite limited in comparison.

- `http://localhost/?patient=5702887&accnum=0000000005`

  Opens all studies resolved by this filter at once. No multiple values.

- `http://localhost/?file=DIR-WITH-MULTIPLE-STUDIES`
- `http://localhost/?file=STUDY-DIR`
- `http://localhost/?file=STUDY-DIR/102.dcm,295.dcm`

  Opens multiple studies, a single study or single image(s). Multiple values can be separated by commas; however, MedDream will treat them as separate studies and won't join images, even from adjacent files, into the same study.

  A single series is obviously possible if files are grouped into series subdirectories.

The parameters study, patient and accnum are supported by almost all PACS plugins. Rare exceptions are listed under a particular PACS integration in 3 Integration with PACS chapter.

Additional limitations: only HTTP GET method (old MedDream versions supported POST as a safer alternative).

To enable this integration mode, add `auth-his` to the `spring.profiles.include` parameter, specify URL parameters in `authentication.his.valid-his-params`, maximum number of resulting studies in `authentication.his.maxObjects` and list permissions in `authentication.his.defaultHisPermissions`. For example,

```
spring.profiles.include=auth-pacsone,auth-his
authentication.his.valid-his-params=study
authentication.his.maxObjects=2
#authentication.his.useSameSession=true
#authentication.his.patient-integration-open-studies=false
#authentication.his.allowOnePatientViewOnly=false
authorization.defaultHisPermissions=EXPORT_ARCH,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
#authentication.his.tokenServiceAuthUsername=basicUser
#authentication.his.tokenServiceAuthPassword=basicPassword
```

```
spring.profiles.include=auth-pacsone,auth-his
authentication.his.valid-his-params=study,accnum
authentication.his.maxObjects=2
#authentication.his.useSameSession=true
#authentication.his.patient-integration-open-studies=false
authorization.defaultHisPermissions=EXPORT_ARCH,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
```

```
spring.profiles.include=auth-pacsone,auth-his
authentication.his.valid-his-params[0]=patient,accnum
authentication.his.maxObjects=2
#authentication.his.useSameSession=true
#authentication.his.patient-integration-open-studies=false
authorization.defaultHisPermissions=EXPORT_ARCH,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
```

```
spring.profiles.include=auth-pacsone,auth-his
authentication.his.valid-his-params[0]=study
```

(continues on next page)

(continued from previous page)

```
authentication.his.valid-his-params[1]=patient,accnum
authentication.his.valid-his-params[2]=patient
authentication.his.valid-his-params[3]=storage
authentication.his.maxObjects=2
#authentication.his.useSameSession=true
#authentication.his.patient-integration-open-studies=false
authorization.defaultHisPermissions=EXPORT_ARCH,DOCUMENT_VIEW,BOUNDING_BOX_VIEW
```

(Formally the first example needs `...valid-his-params[0]=study`, too. Luckily "[0]" is optional in this situation.)

The second example allows either "study" or "accnum" in the same installation. The third one defines **a combination** of "patient" and "accnum" – not the same as **either** "patient" or "accnum". The fourth contains multiple parameters/combinations at once, and also allows specifying a particular PACS plugin via the "storage" parameter.

The "maxObjects" parameter is zero by default, which means no limit and no protection against excessive amount of studies. However, as of 8.8 there still is a certain hardcoded limit (100) at the frontend; therefore maxObjects should be smaller than that in order to help in all cases.

The "useSameSession" parameter allows you to have multiple viewer windows during "unsafe" HIS integration. By default, a newly opened viewer window also replaces the session, therefore a window from the earlier session may get "Access denied" errors.

When "allowOnePatientViewOnly" is true, MedDream won't accept tokens where .restrictions.patient contains more than 1 element.

The "tokenServiceAuthUsername" and "tokenServiceAuthPassword" parameters provide Basic Authentication credentials here, at the client side (when they are set up at the Token Service).

Warning: The "storage" parameter is treated as optional, and is silently ignored if not enabled in valid-his-params.

If missing or ignored, MedDream searches for objects via all available PACS plugins.

5. Deployment

This chapter is dedicated to PACS-independent part of the deployment. Normally you should start reading at support for a specific PACS under 3 Integration with PACS, as those parts refer to this chapter anyway.

In case of Linux, this chapter assumes that the user has root permissions and appropriate part of MedDream installation archive was extracted into /opt/meddream (the file /opt/meddream/MedDream-*.jar exists).

5.1. Configuration

5.1.1. Essential configuration and the first run

The main configuration file is `application.properties` in the directory of MedDream-*.jar.

Warning: The industry-standard Java Properties format doesn't support inline comments:

```
parameter = value # definitely not a comment
```

In this example, `parameter` is equal to `value # definitely not a comment`. The result will be unpredictable as not all parameters are validated in advance (you will not get a comprehensive error about a too long value, or invalid characters, or at least invalid numeric format).

The entire line must be dedicated to a comment:

```
# a proper comment that either disables something or just explains something
parameter = value
```

Another important caveat is that duplicate entries overwrite the previous ones. The following

```
parameter = value1
# lots of other configuration
parameter = value2
```

will result in "value2" without any warnings.

After any changes it is required to restart the Java application.

Note: During an upgrade it's recommended to use application.SAMPLE.properties (shipped with MedDream) as a template and update every setting based on contents of your old configuration file. This will prevent misunderstandings like parameters renamed or removed between versions.

1. Revise the following most important configuration parameters:
   - Application TCP/IP port

     ```
     server.port = APPLICATION_PORT
     ```

     Allows you to choose a different port if the default one is occupied. Example: `server.port=80`.
   - wkhtmltopdf (https://wkhtmltopdf.org/downloads.html) is required to include reports in exported studies. You can skip this if reports aren't used at all, or aren't included during export.

     Under Linux the binary is normally found in PATH. Under Windows the recommended install locations are C:\Program Files\wkhtmltopdf and C:\Program Files (x86)\wkhtmltopdf; you can also download the 7z Archive and just copy the EXE file to sys/report/wkhtmltopdf.exe. If your system must use a different location, then you can specify it in application.properties:

     ```
     com.softneta.export.htmlToPdf = D:/wkhtmltopdf
     com.softneta.export.htmlToPdf = /opt/wkhtmltopdf
     ```

     For Linux, also see https://sasablagojevic.com/setting-up-wkhtmltopdf-on-docker-alpine-linux. Particularly, when creating a Docker image based on adoptopenjdk/openjdk8:alpine-slim, one can use the command `apk add --update --no-cache wkhtmltopdf xvfb ttf-dejavu ttf-droid ttf-freefont ttf-liberation`. After the installation it is advised to check for missing fonts and other typical problems by running

     ```
     wkhtmltopdf --enable-local-file-access default.html test.pdf
     ```

     in the directory sys/report.

     If wkhtmltopdf is unavailable, or its dependencies like QtWebKit are missing, then a possible alternative is weasyprint (https://doc.courtbouillon.org/weasyprint/stable/first_steps.html or https://kozea.github.io/WeasyPerf/samples/doc/doc.html). Because it's a Python module, under Windows you should have Python binaries (one of "py", "python", "python3") reachable via PATH.

     You can even use a custom solution that accepts two command line parameters: a path to the HTML file and a path to the final PDF file. In that case the htmlToPdf setting must point

to an executable file with the extension “exe”, “bat” or “sh”.

- Remaining values

  During the first start these can usually remain unchanged. See 5.1.13 Reference for remaining configuration parameters.

2. Start the Java application manually with a console command `java -cp MedDream-8.9.1.jar --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED --add-opens=java.base/jdk.internal.misc=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/com.sun.jmx.mbeanserver=ALL-UNNAMED --add-opens=java.base/sun.reflect.generics.reflectiveObjects=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.time=ALL-UNNAMED --add-exports=java.desktop/sun.awt.image=ALL-UNNAMED org.springframework.boot.loader.launch.PropertiesLauncher`.

   The -DANTLR_USE_DIRECT_CLASS_LOADING option is needed because of a certain problem with Reporting function.

   Also see 5.1.9 Adjusting Java memory limits if your DICOM files are large and Java therefore needs more memory than JRE offers by default.

   Lastly, path to the subfolder `sys/plugins` must be given via -Dloader.path. This is still needed for using any of the included PACS integrations plugins because of a different loading mechanism of SPI-based plugins compared to OSGi.

   A full example: `java -cp MedDream-8.9.1.jar -Xmx1024m -Dloader.path=./sys/plugins -DANTLR_USE_DIRECT_CLASS_LOADING=true --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED --add-opens=java.base/jdk.internal.misc=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/com.sun.jmx.mbeanserver=ALL-UNNAMED --add-opens=java.base/sun.reflect.generics.reflectiveObjects=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.time=ALL-UNNAMED --add-exports=java.desktop/sun.awt.image=ALL-UNNAMED org.springframework.boot.loader.launch.PropertiesLauncher`

At this point it should just start up without errors, and the message “Application ... started” will eventually appear.

The login form at http://localhost:APPLICATION_PORT/ is disabled by default (results in a “403 Forbidden” message when application.properties is composed from scratch) unless 5.1.6 Built-in authentication and authorization is configured. On the other hand, application.SAMPLE.properties already configures the `auth-inmemory` authentication by default, with a predefined password. As the optimal method might be PACS-dependent, you can configure it a bit later, together with the plugin for a particular PACS.

5.1.1.1. Additional steps for Linux

If Java application will run under a non-root user, then adjust file system permissions:

```
chown LIMITED_USER:LIMITED_GROUP -R /opt/meddream
find /opt/meddream -type d -exec chmod 775 {} \;
find /opt/meddream -type f -exec chmod 664 {} \;
chmod 660 /opt/meddream/application.properties
```

Running Java application as root is OK for initial tests, however in production environment it should run under some limited user instead, ideally the one who can access all DICOM files in read only mode. A compromise would be to run the JVM under the same user as the PACS process.

This adjustment step makes sure that a process running under a limited user can access its files, especially the ones created earlier under name of root. In the simplest case you can examine the log file after starting the process under a different user, as even the log file will likely be non-writeable.

5.1.1.2. Additional steps for Windows

Make sure that any application.properties files are readable only by administrators and the account dedicated to the JVM process – not by the entire `Users` group.

When correcting this from the command line, you can use the following in the directory where the file resides:

```
icacls application.properties /inheritance:d /remove:g "Users"
```

5.1.2. System.json

The file system.json (sys/settings/system.json by default) is automatically created if missing when the Java application starts. Several key features require this file to be updated during installation; the application must be restarted afterwards.

Important Notice:

1. The default values in system.json are carefully selected based on medical device usage to ensure optimal workflow efficiency and user experience. It is recommended to review these settings before making any modifications to align with specific operational and security requirements.
2. If an endpoint or configuration setting is not required for your use case, it is strongly recommended to disable it or explicitly set its value to false. This helps ensure system security, optimize performance, and prevent unintended behavior. Please review all settings carefully before deployment.

An example of the entire file is shown below (only some values are default). When updating, take care to preserve the syntax (quotes, commas, brackets, etc.); a text editor with JSON syntax checking is recommended.

```
{
  "disableMultiFrameVideoAutoLoad": false,
  "preloadVtkNatives": false,
  "features": {
    "patientHistory": false,
    "keyObjectAndPresentationStateQuickSave": true,
    "keyObjects": true,
    "presentationState": true,
    "searchSettings": true,
    "viewerSettings": true,
    "reportSettings": true,
    "panelToolbarSettings": true,
    "hangingProtocolsSettings": true,
    "dicomPrintSettings": true,
    "export": true,
    "archive": true,
    "search": true,
    "reports": true,
    "remoteReports": true,
    "liveShare": false,
    "allowCopyMeasurementValue": true,
    "hangingProtocols": "V2",
    "openTabsTrackingMethod": "NONE",
    "prepareInstances": false,
    "batchImageRequests": true,
    "thumbnailsPerSeries": "FIRST",
    "summaryThumbnailsFor": ["CT", "MR", "PT", "NM"],
    "preloadSeries": "OFF",
    "preloadAllFrames": false,
    "boundingBoxAnnotations": true,
    "viewToClipboard": "IMAGE",
```

(continues on next page)

(continued from previous page)

```
    "pngPSToClipboard": "NONE",
    "viewToDICOM": true,
    "mistMpr3D": true,
    "mprExport": true,
    "clearCache":true,
    "dicomPrint": true,
    "measurementsPropagation": true,
    "callCachedMetadata": true,
    "callCachedMetadataForTransferSyntax": [
      "1.2.840.10008.1.2.4.201",
      "1.2.840.10008.1.2.4.202",
      "1.2.840.10008.1.2.4.203"
    ],
    "largeVolumeMprSplit": false,
    "overlay": true,
    "digitalSubtraction": "NONE",
    "returnMinMaxPerFrame": true,
    "cad": true,
    "checkKOPRPermissions": false,
    "playVideoInVideoTag": false,
    "browserExtensionIntegration": false,
    "hangingProtocolsQuickSave": true,
    "quadrantZoomEnabled": false,
    "workspace": {
        "allowWorkspaceSaving": true
    },
    "uploadDicomToPacs": false,
    "autoAdjustImageRotation": true,
    "allowForwardSeriesAndInstanceDicomToPacs": true
  },
  "blackListedSopClasses": [
    "1.2.840.10008.5.1.4.1.1.66",
    "1.2.840.10008.5.1.4.1.1.4.2"
  ],
  "forwardPacs" : [
    {
      "storeScpIp": "127.0.0.1",
      "storeScpPort": 104,
      "storeScpAet": "PACS",
      "description": "An example destination for Forward",
      "sender": "PacsOneRouter,MainArchive"
    },
    {
      "storeScpIp": "192.168.200.65",
      "storeScpPort": 5678,
      "storeScpAet": "OSIRIX",
      "description": "Osirix workstation",
      "sender": "MainArchive"
    }
  ],
  "languages" : ["lt","en","ru","de","fr","es","it"],
  "dicomLibraryConfiguration": {
      "dicomLibrarySender": "someone@mail.com",
      "dicomLibrarySubject": "My special study",
      "languages": ["en","de"]
      "language": "en"
```

(continues on next page)

(continued from previous page)

```
},
"formatIntegrationLinkInViewer": false,
"textureInterpolationType": 1,
"requestsConfig": {
  "maximumMetadataStreams": 5,
  "maximumPixelStreams": 15,
  "maximumMultiFrameStreams": 4,
  "batchSizeMb": 5,
  "multiFrameBatchSizeMb": 2
},
"workersConfig": {
  "maxWorkers": "50%",
  "maxSegmentationWorkers": "50%",
  "maxMeasurementWorkers": "50%",
  "maxJphWorkers": "0%"
},
"cacheSupervisorConfig": {
  "enabled": false,
  "runFrequencyInSec": 10
},
"structureListenerConfig": {
  "enabled": false,
  "runFrequencyInSec": 30
},
"scrollingConfig": {
  "modality":["CT", "MR", "PT", "OPT", "NM", "MG", "OCT"],
 "sopClass":[]
},
"notificationsConfiguration": {
  "autoCloseAfterSec":10
},
"personNameConfiguration": {
  "formatIn":"",
 "formatOut":null
},
"searchHistoryByPatientId": true,
"dontAutoOpenTheseAnnotations": ["Lunit"],
"fusionConfig": {
  "modality": ["CT", "MR", "PT"],
  "sopClass": [],
  "ignoreMissingSliceThickness": false
},
"segmentationConfig" {
  "smartPaintPreloadOptions": "wait-for-toolbar",
  "messagingLevel": "minimal",
  "modality": ["MR", "CT", "PT", "NM", "MG", "CR", "DX", "US"],
  "sopClass": []
}
"windowPosition": [{
    "name": "patient-history",
    "width": 1245
    "height": 800
  }
],
"instancesProgressivePreloadConfig": {
    "preloadCount": 5,
```

(continues on next page)

(continued from previous page)

```
        "supportedModalities": ["CT", "MR", "PT", "NM"],
        "supportedSOPClasses": [],
        "enableSeriesManualPreload": true
    },
    "mistMprConfig": {
        "modalities": ["CT", "MR", "PT", "NM"],
        "sopClasses": []
    },
    "dentalConfig": {
        "dentalAPIUrl": "lib/3rdparty/dental"
    },
    "multiframeCacheRetention": 20
}
```

- disableMultiFrameVideoAutoLoad: false

  (reserved) Disables multiframe and video autoload. Used for automated testing purposes, do not change this setting.

- preloadVtkNatives: false

  If `true`, VTK native libraries are preloaded early during Core startup. This can be beneficial when the installation is going to need VTK – for example, via regular use of the MPR Export functionality (see features.mprExport).

  By default, the libraries are loaded on demand (the first use of MPR Export after a restart will feel slower), or not loaded at all, therefore system resources are not needlessly wasted.

  Note: Under Linux, VTK is expected as a system package, with LD_LIBRARY_PATH updated accordingly (MedDream will use the latter when loading the package).

- patientHistory: false

  If `true`, the viewer window will load patient history and the corresponding button is enabled. Disabled by default.

- keyObjects: true

  The viewer window will load key objects.

- keyObjectAndPresentationStateQuickSave: true

  The viewer window will display the Key Object And Presentation State Quick Save button.

- presentationState: true

  The viewer window will load presentation state objects.

- searchSettings: true

  Enable search settings page in the settings window.

- viewerSettings: true

  Enable viewer settings page in the settings window.

- reportSettings: true

  Enable report settings page in the settings window. Always disabled if Lite/Enterprise license.

- panelToolbarSettings: true

  Enable panel toolbar settings page in the settings window.

- hangingProtocolsSettings: true

Enable hanging protocol settings page in the settings window.

- dicomPrintSettings: true

  Enable DICOM Print settings page in the settings window.

- export: true

  Globally enable the Export/ISO (Burn) function. If disabled, it won't be available even to users who have the EXPORT_ISO permission.

- archive: true

  Globally enable the Export/Archive (Save Active . . .) function. If disabled, it won't be available even to users who have the EXPORT_ARCH permission.

- search: true

  Globally enable or disable the Search window. If disabled, the window will be replaced with an error message even for users who have the SEARCH permission.

  The window is enabled by default to make the installation simpler. However currently it implements no per-user filters and allows access to every patient on the PACS. Do not forget to disable it if not needed.

  For example, if MedDream will open only via HIS integration, `true` is still recommended as an *initial* choice, as interactive login usually provides a faster way to verify whether images can be displayed. In this case do not forget to switch to `false` after testing.

- reports: true

  Globally enable or disable the Reporting window. If disabled, it won't be available even to users who have the REPORT_VIEW or REPORT_UPLOAD permission. A license of Lite/Enterprise type disables the functionality, too.

- remoteReports: true

  Globally enable or disable the "remote reports" functionality that replaces the Reporting window (remoteReports=true is incompatible with reports=true). If disabled, it won't be available even to users who have the REPORT_VIEW permission. Also see 5.1.8.4 Remote reports.

  A non-Premium license disables this, too.

- liveShare: false

  Globally disable or enable the LiveShare functionality.

  Note: With `true`, you must also enable the `liveshare` Spring profile and configure the legacy HIS integration with at least the parameter "study", in application.properties:

  ```
  spring.profiles.include = liveshare,auth-his
  authentication.his.valid-his-params = study
  ```

  A license of Lite/Enterprise type disables the functionality, too.

  Warning: Each LiveShare connection (from a guest to the host) occupies one concurrent connection. Make sure to have a license with enough connections.

- allowCopyMeasurementValue: true

  Globally enable or disable the "Copy measurement values" function. True by default, however can't be enabled under a Lite license.

- "hangingProtocols": "V2"

Globally enable or disable the Hanging Protocols functionality.

In MedDream 8.3+ the allowed values are `"V1"` (same as `true`), `"V2"`, `"NONE"` (same as `false`). "V2" is the default since 8.7.0. **Since 8.9.0, "V1" is officially deprecated** and will be removed in the future; customers are encouraged to migrate to V2.

- openTabsTrackingMethod: "NONE"

  Possibility to enable more strict tracking of open tabs. Ideally, when all tabs are closed by pressing the "X" button, MedDream should log off as if "Logoff" were chosen from the menu.

  The string "SOCKETS" enables websocket-based tracking. It is also needed to add `track-tabs` to spring.profiles.include (application.properties).

  Note: There is a grace period of 10 seconds. If you open MedDream once more during that time, then the session remains.

  Without this period, a single tab could get logged off even when reloading its contents (pressing the F5 button).

  The default value, "NONE", is forced in case of a Lite license.

  Note: The tab management logic stores IDs of open tabs in browser's local storage. If the latter isn't cleared properly due to some bug, a rare situation might arise where *any new tab will open and close immediately*. As a workaround, the end user can delete the browser cache (Ctrl+Shift+Del, then at least "Cookies and other site data", in Chrome).

- prepareInstances: false

  If true, the Preload Study function will make sure each image is prepared. This might increase performance up to 20%.

  The setting does not affect a similar endpoint, `POST /manage/prepareInstances/{STUDY}/{STORAGE}`. This one is to be called by HIS or other third-party systems, requires authentication incompatible with MedDream user interface, and is controlled via a different configuration (`management.endpoints.web.exposure.include = ...,prepareCache,...`).

- batchImageRequests: true

  Enables batch downloading of image pixels; this may result in multiple images per request and therefore reduce the overhead. Assign `false` for pre-7.7.0 behavior.

- thumbnailsPerSeries: "FIRST"

  By default, MedDream 7.7.0+ attempts to speed up loading of CT, PT and MR studies (defined by summaryThumbnailsFor and summaryThumbnailsForSopClasses) by displaying only the summary thumbnail in every series. Assign "ALL" for pre-7.7.0 behavior.

  Warning: As of 8.9.0+, "ALL" is not compatible with client-side image decompression (see com.softneta. preparation.leaveHtj2kPixelsExceptTheseSopClasses and settings with "prepareFromPacsMetadata" in the name). A study will many small images will require a lot of browser's memory and scrolling through thumbnails will be slow. Furthermore, when all 16 viewports in a 4x4 layout are occupied, an attempt to scroll through thumbnails that weren't loaded before, will result in a "context lost" error in some viewport.

  Another possible value is "MIDDLE". It requires com.softneta.preparation.thumbnailCreation = all. Unfortunately, as the preparation is performed per-instance and as soon as possible (without waiting until the entire study is sent), it is not feasible to determine which object is at the middle; all thumbnails must be available for thumbnailsPerSeries=MIDDLE.

- summaryThumbnailsFor: ["CT", "MR", "PT", "NM"]

List of Modality values to which the `thumbnailsPerSeries` setting applies. Before 7.9.0 the values CT", "PT" and "MR" were hardcoded in the frontend.

Note: When the backend is configured with com.softneta.preparation.thumbnailCreation = first (by default), then the set of values in summaryThumbnailsFor should include values of com.softneta.preparation.thumbnailCreationModalities (not all thumbnails are created for those modalities, in order to have faster processing). Generally, summaryThumbnailsFor can be a larger set, however it mustn't be smaller. Otherwise, the frontend will request thumbnails that aren't routinely produced by the backend; studies will open slower because of these unexpected requests.

- preloadSeries: "OFF"

  When that summary thumbnail is enabled, then it can also trigger preload of its series. "NONE" – no preload, "OFF" – the end user must click on the summary thumbnail to trigger the preload, "ON" – the preload starts automatically.

- preloadAllFrames: false

  True: during a study/series preload, load all frames of a multiframe image. **Incompatible with cacheSupervisorConfig.enabled=true.**

  False (default): load only the first frame (behavior before 8.7.0 and starting from 8.8.1). This might also be required if the studies are too large for the browser.

- boundingBoxAnnotations: true

  Globally enables the Segmentation toolbar button and the Bounding Box tool below it. The user must have the BOUNDING_BOX_EDIT permission to see the button. Second, the button is disabled by default in the Settings window (Viewer / Toolbar properties / Segmentation Tools / Bounding Box). Finally, the functionality is disabled by a Lite/Enterprise license, too.

- viewToClipboard: "IMAGE"

  Functionality of the button "Copy image to the clipboard". "IMAGE" – the button copies an image of the active viewport (with burned-in Info Labels, measurements, etc.); "JSON_IMAGE" – it copies a certain JSON structure with measurements and Base64-encoded image as in the "IMAGE" case; "NONE" – the button is not displayed.

- pngPSToClipboard: "NONE"

  Functionality of the button "Export original image to the clipboard". "IMAGE" – the button copies the active image of original size and without any burned-in information (as with "PNG" and "Active Image" in the Export window); "JSON_IMAGE" – it copies a certain JSON structure with measurements and Base64-encoded image as in the "IMAGE" case; "NONE" – the button is not displayed.

  Note: Additional requirements for displaying the button are: 1) modality DX, CR, RX, CT or MR; 2) not a video or multiframe file.

- viewToDICOM: true

  Globally enable the button "Save viewport content as secondary capture DICOM". The button also needs the COPY_TO_DICOM permission and a license type not Lite/Enterprise. If one of the conditions isn't satisfied, then the button is not visible and only secondary capture images saved earlier can be accessed.

  This feature is needed by the Montage functionality, too.

- mistMpr3D: true

  Globally enable or disable the 3D view (4th viewport) of the "Oblique MIST" functionality.

- mprExport: false

Globally disable or enable the MPR Export functionality.

- clearCache: true

  Globally enable or disable the Clear Cache functionality. When enabled, the user must also have the CLEAR_CACHE permission.

- dicomPrint: true

  Globally enabled or disable the DICOM Print functionality.

- measurementsPropagation: true

  Show copy/paste buttons when creating a measurement. Enabled by default. Can't be enabled under a Lite license.

- callCachedMetadata: true

  Just after calling the /structure endpoint, also call the /cachedMetadata endpoint: it will collect instance metadata resources already present in the cache, and therefore might reduce the number of requests to the backend. This is especially useful for hanging protocols.

- callCachedMetadataForTransferSyntax: . . .

  When callCachedMetadata above is false, it is still possible to get /cachedMetadata called if there is at least one object in the study with the Transfer Syntax listed here.

  The default is empty.

- largeVolumeMprSplit: false

  If true, MIST MPR viewports will divide the volume texture into up to 4 parts before rendering and blend them afterward. This effectively raises Chrome's v149+ 1 GB texture size limit to 4 GB.

- overlay: true

  Globally enable or disable the overlay functionality, including the related controls in the Viewer window.

- digitalSubtraction: "NONE"

  Enable the Digital Subtraction Angiography functionality. "NONE" (default): the functionality is disabled. "AUTO": the functionality is applied automatically. "MANUAL": a button is displayed if the images contain necessary information, and the end user can start DSA manually.

- returnMinMaxPerFrame: true

  Enable or disable retrieval of ranges of pixel values for each frame of multiframe images. Enabled by default.

- cad: true

  Globally enable or disable the CAD Marks functionality. Can't be enabled if the license type is Lite or Enterprise.

- checkKOPRPermissions: false

  `true` enables new permissions of 8.5.0+, KO_PR_EDIT and KO_PR_VIEW. By default (false), KO and PR functionality is unrestricted as before, however creation of new objects is not allowed by a Lite/Enterprise license.

- playVideoInVideoTag: false

  Set to `true` if video playback uses too much CPU on thin clients or virtual desktops. Then the playback is fully handled by the browser and becomes more efficient; on the other hand, MedDream image transformations like Zoom or Rotate do not work anymore.

- hangingProtocolsQuickSave: true

  `true` (default) globally enables the Quick Save HP functionality. Stays `false` if MedDream license is Lite or Enterprise. Also requires either the HANGING_PROTOCOLS_EDIT or the ADMIN permission.

- browserExtensionIntegration: false

  `true` enables the “Missing browser extension” warning message. Only a minority of scenarios can benefit from it (for example, when all end users are always supposed to use the extension), therefore the message is disabled by default.

- quadrantZoomEnabled: false

  `true` (default) globally enables the Quadrant Zoom functionality. However, this still has no effect (the corresponding buttons remain unavailable in the Settings window) if MedDream license is Lite or Enterprise.

- allowWorkspaceSaving: true

  `false` globally disables workspace saving. The data object is saved in the preparation cache near the “struct” file; the file name is `workspace_` plus login name.

- uploadDicomToPacs: false

  `true` globally enables the Upload function. If disabled (default), it won’t be available even to users who have the UPLOAD_DICOM_FILES permission.

- autoAdjustImageRotation: true

  `false` disables automatic rotation of some MG images by 180 degrees.

- allowForwardSeriesAndInstanceDicomToPacs: true

  When true (default), the Forward window opened from a Viewer window contains additional controls for selecting a particular series or image, and the backend forwards only the selected parts of the study.

- blackListedSopClasses: [“1.2.840.10008.5.1.4.1.1.66”]

  Thumbnails of images with these SOP Classes will not be shown in the viewer.

  This list is appended to a hardcoded list of non-image objects from DICOM Standard. (By the way, the above . . . 1.1.66 SOP Class is already there.) Main purpose of this setting is to hide any Private SOP Classes not known in advance.

- storeScpIp: “127.0.0.1”
- storeScpIp: “192.168.200.65”

  Destination IP address (or hostname) for the Forward function. Only for com.softneta.meddream.dcmsnd.forwardingMethod=native.

- storeScpPort: 4242
- storeScpPort: 5678

  Destination TCP port for the Forward function. Only for com.softneta.meddream.dcmsnd.forwardingMethod=native.

- storeScpAet: “PACS”
- storeScpAet: “OSIRIX”

  Destination AE Title for the Forward function. Will be visible in the Forward window.

- description: “An example destination for Forward”
- description: “Osirix workstation”

  Human-readable description of a destination device for the Forward window.

- sender: “PacsOneRouter,MainArchive”
- sender: “MainArchive”

Can be empty since 8.5 for com.softneta.meddream.dcmsnd.forwardingMethod=cmove, and is ignored otherwise.

Identifies PACSes that can send to this destination (because this destination is configured at the PACS). This comma-separated list, however, accepts not AE titles but PACS integration plugin storage identifiers. Empty means "all PACSes".

The Forward window will narrow the list of available destinations, depending on studies selected for forwarding, and will therefore prevent situations where the PACS does not know the destination AET. Suppose the study X, found via PACS integration plugin with a storage identifier Y, is selected. Then the window will list only destinations where the `sender` element contains this storage identifier, Y, or is empty completely.

In essence, the "cmove" method uses .storageScpIp, .storageScpPort and .storageScpAet of the storage from which the study has been opened. This PACS is ordered to forward the study/studies that exist on it, to forwardPacs[].storeScpAet selected by the end user.

- languages: ["lt","en","ru","de","fr","es","it"]

  List of localization languages that will be offered (in this order) in MedDream user interface. All major MedDream windows (Search, Viewer, Settings) have the "Languages" drop down near the upper right corner, unless there is a single language configured.

  When MedDream is opened on user's computer for the first time (or after clearing cookies), then the first entry from this list becomes the default.

- dicomLibraryConfiguration.dicomLibrarySender: "someone@mail.com"
- dicomLibraryConfiguration.dicomLibrarySubject: "An interesting case at DICOM Library"
- dicomLibraryConfiguration.languages: ["en", "de"]
- dicomLibraryConfiguration.language: "en"

  The Share function requires at least one non-null value here (can be just empty strings/arrays). Users will also need UPLOAD_DICOM_LIBRARY permission individually.

  The languages are offered in the Share window and refer to emails which will be sent. As of 7.9, values understood by dicomlibrary.com are "en" and "de". If `.languages` is empty, there is no language choice in the window.

  Note: DICOM Library is an external service. MedDream product includes the integration functionality only. **The actual service is provided under MedDream SLA, for educational or scientific purposes only, not for diagnostic use.**

- formatIntegrationLinkInViewer: false

  (experimental) After the viewer opens, the URL in browser's address bar is automatically adjusted so that it becomes a HIS integration link suitable for copying. Only the "study" parameter (and therefore a single PACS plugin) is supported.

  "true" is ignored under a Lite license.

- textureInterpolationType: 1

  Type of WebGL texture interpolation: 0 (disabled), 1 (linear, default), 2 (bicubic), 3 (bicubic spline), 4 (different implementation of "linear"), 5 (Lanczos).

  When the image is zoomed in to the extreme, pixels will become large squares if the setting is zero. With 3, boundaries of individual pixels are no more visible, however very small details, like a couple of pixels in total, might disappear completely.

- maximumMetadataStreams: 2

  Number of parallel streams to download study/series metadata.

Note: In MedDream before 7.8, this setting was called `seriesParallelStreams` and was applicable to batchImageRequests=true only.

- maximumPixelStreams: 15

  Number of parallel streams to download image pixels. Applies to all kinds of single-frame pixel requests: the legacy /pixels (for batchImageRequests=false) and the recently introduced /multi-pixels (for batchImageRequests=true).

  In MedDream before 7.8, this setting was named `parallelStreams` and was applicable to batchImageRequests=true only.

- maximumMultiFrameStreams: 4

  Number of parallel streams to download multiframe pixels. Applies to any value of batchImageRequests.

  Note: In MedDream before 7.8, this setting was named `multiFrameParallelStreams` and was applicable to batchImageRequests=true only.

- batchSizeMb: 5

  Image pixel downloads are grouped so that the combined estimate of download size doesn't exceed this setting (if batchImageRequests is not false).

- multiFrameBatchSizeMb: 2

  Equivalent of batchSizeMb for downloading frames of a multiframe image (if batchImageRequests is not false).

- workersConfig: . . .

  Number of browser threads allocated for various functions. "maxWorkers" are for decompression of pixel data, "maxSegmentationWorkers" – for the segmentation function, "maxJphWorkers" – for an experimental decompression of HTJ2K streams in the frontend (see also com.softneta.preparation.leaveHtj2kPixelsExceptTheseSopClasses), "maxMeasurementWorkers" – for speeding up calculation during certain measurement types. Values in percent are relative to actual number of CPU threads at a particular client machine.

- cacheSupervisorConfig: . . .

  Monitors caching of images at the frontend. Every `runFrequencyInSec` seconds images not displayed in any viewport are marked for removal, and those that were marked previously, are removed from memory.

  Specify true for `enabled` to enable this functionality.

- structureListenerConfig: . . .

  Monitors study structure changes by periodically comparing attributes NumberOfStudyRelatedSeries and NumberOfStudyRelatedInstances with values in the cached structure. When a mismatch is detected, the user is offered to refresh the study.

  Specify true for `enabled` to enable this (it's disabled by default). You can also adjust `runFrequencyInSec` (60 seconds by default), however too frequent polling increases the load at the PACS side and the possibility of detecting changes multiple times in a row when a large study is still being sent to the PACS. (Imagine pressing the "Apply structure changes" button and, after the study reloads, the button immediately becomes active again.)

  Additional requirements:

  – study structure caching must be enabled (com.softneta.cache.studyStructureSec > 0);

- CLEAR_CACHE permission is needed, as the structure will be automatically removed from the cache in order to fetch the changed version;
- as of MedDream 8.9.0+, polling of structure changes is supported only by plugins PacsOne, Dcm4chee2, Dcm4chee5, Orthanc, QR, Dicomweb, AmazonS3, AWSHealthImaging. A supporting plugin needs its own setting ... [PLUGIN_INDEX_NUMBER].studyStructureCountersEnabled=true;
- a "Premium" MedDream license.

Note: The plugins attempt to use dedicated/optimized queries in order to query and return only these two attributes, however not every integration makes this possible. With universal plugins like QR and Dicomweb, the outcome might still depend on the PACS.

The functionality will likely be incompatible with Amazon S3 integration, if one uses very short-lived URLs to the DICOM files and therefore structure caching needs to be disabled. At the moment the plugin doesn't support it, however the disabled caching would make adding the support pointless.

An alternative is com.softneta.cache.studyStructureExpiresOnChange=true: it makes sure that the study is automatically refreshed when opening, but doesn't detect a change while the study is still being viewed.

- scrollingConfig: ...

  The vertical slider is displayed on the image if its Modality or SOP Class attribute matches a value listed here. The default is a few modality values and an empty list of classes, and corresponds to behavior of earlier versions where those modalities were hardcoded.

- notificationsConfiguration.autoCloseAfterSec: 10

  A message is displayed on screen for this number of seconds, then hidden away.

- personNameConfiguration: ...

  Defines the format of person names (currently Patient Name and Referring Physician Name attributes). In 8.0.0, some plugins (QR, Dicomweb, PacsOne, DCM4CHEE 2.x and 5.x) were updated to return person names in DICOM standard format – with ^ separators, all 5 components, all 3 component groups.

  `formatIn` provides a pattern for identifying name components in the input string, based on separator positions. It applies to every component group. The default value (used instead of null or empty string) is `"{lastName}^{firstName}^{middleName}^{prefix}^{suffix}"` and complies to the Standard.

  In general, a component consumes the part of the input string up to the next separator. When there are not enough separators, then some component will consume all remaining input. Results will be unexpected if the components were entered in a non-compliant manner at the modality, like "John Doe^Dr." instead of "Doe^John^^Dr."; however, if all modalities behave that way, a corresponding `formatIn` could adapt to it.

  `formatOut` is a format string for every component group of the result. The default value (used instead of null or empty string) is `"{prefix} {lastName} {firstName} {middleName} {suffix}"`.

  The simplest use case is to change the component order via `formatOut` – for example, `"{firstName} {lastName}"`.

  After updating to 8.0.0, compliant plugins listed above will produce names with prefixes at the beginning, like "Dr. Doe John" instead of "Doe John Dr.". Furthermore, some plugins are now returning all components instead of just last and first name. Both issues can be worked around with `formatOut`. The behavior won't differ with non-compliant plugins like Orthanc, as they use the legacy space-delimited name format and `{lastName}` in formatIn gets everything.

- searchHistoryByPatientId: true

When true, MedDream uses the traditional patient history search key – a single Patient ID attribute. When false, the search key is a logical AND of attributes Patient Name, Patient's Sex and Patient's Birth Date.

> **Warning:** As of 8.1.0+, the value "false" is only for DCM4CHEE 2.x and 5.x PACS plugins. Other plugins might become confused and won't return any results; this also includes Multi-PACS setups where DCM4CHEE plugins are mixed with other ones.

- dontAutoOpenTheseAnnotations: ["Lunit"]

  Names of annotations that won't be opened automatically when "Auto open annotations" is enabled in the Settings window. The intended use case is filtering of annotations added by automated diagnosis systems, while annotations created manually would still open automatically.

  Empty by default.

- fusionConfig.modality: ["CT", "MR", "PT"]
- fusionConfig.sopclass: []

  In general Fusion is possible between any modalities in the same frame of reference but their values must be configured first. By default, they're CT, MR or PT. This can be changed by the "modality" setting.

  You can also configure a SOP Class UID whitelist in the "sopclass" setting. Both whitelists are combined with logical OR (if both are empty, then Fusion isn't allowed at all).

- fusionConfig.ignoreMissingSliceThickness: false

  If true, Fusion will not require Slice Thickness attribute present in all images.

- segmentationConfig.smartPaintPreloadOptions: "wait-for-toolbar"
- segmentationConfig.messagingLevel: "minimal"
- segmentationConfig.modality: ["MR", "CT", "PT", "NM", "MG", "CR", "DX", "US"]
- segmentationConfig.sopClass: []

  "smartPaintPreloadOptions" can be "wait-for-toolbar" (default) or "load-on-open". The latter will enable parsing of probability-based annotations as soon as possible.

  "messagingLevel" can be "minimal" (default) or "medium".

  "modality" and "sopClass" identify kinds of images for which the Segmentation functionality is enabled (the toolbar button appears).

- instancesProgressivePreloadConfig.preloadCount: 10
- instancesProgressivePreloadConfig.supportedModalities: ["CT", "MR", "PT", "NM"]
- instancesProgressivePreloadConfig.supportedSOPClasses: []
- instancesProgressivePreloadConfig.enableSeriesManualPreload: true

  When loading a particular image, also load up to preloadCount images before and after it. This behavior, however, applies only if Modality attribute matches an entry in supportedModalities, or SOP Class UID matches an entry in supportedSOPClasses.

  enableSeriesManualPreload=false disables the "cloud" icon in the series header, however only for series where supportedModalities or supportedSOPClasses match.

- mistMprConfig.modalities: ["CT", "MR", "PT", "NM"]
- mistMprConfig.sopClasses: []

  What kinds of images get the MIST MPR toolbar button. Previously not encountered IODs might also need to adjust com.softneta.preparation.sopClassesForFrameExtraction etc.

- dentalConfig.dentalAPIUrl: “lib/3rdparty/dental”

  Configures integration with a certain proprietary 3rd-party service for analyzing dental images. As of 8.8.0, the API is not public yet.

  .dentalAPIUrl is a URL fragment between MedDream’s own base web path (“/” in most cases) and service-specific endpoint (like {storageId}/studies/{studyUid}/. . .). A reverse proxy must route URLs of this kind to the external service.

- multiframeCacheRetention: 20

  In 8.9.0+, the frontend caches raw RGBA multiframe pixels in memory (downloaded via /multi-pixels and then decoded) for this amount of seconds.

  Due to this limited caching, memory requirements are significantly lower. Particularly,

  – multiple viewports that attempt to show the same multiframe image do not need to request their own copies from the backend; results of a single request will be reused by all of them.

  – if a viewport with a multiframe image is closed, and that image is reopened in another viewport, no new request to the backend is needed.

  – if the same study/series is closed and reopened within this time, the frontend will only fetch structure, thumbnails, instance metadata and single-frame pixels – but no multiframe pixels.

  There is no exact equivalent for pre-8.9.0 behavior.

- windowPosition: [{“name”: “name1”, “width”: width1, “height”: height1},{“name”: “name2”, . . . },. . .]

  Override default widths and/or heights of GUI windows. When configuring only the width, you can target a **group of windows**; for example, “modalSmallWidth” targets Register, About and many other windows, while “register” is for the Register window only.

  Dimensions are in pixels. Unspecified or zero means “default” and is ignored. **Minimum height is 300**, smaller values are also ignored. There is no minimum width, you can use the default below as a ballpark value; significantly smaller values were not tested and might result in incorrect appearance or behavior.

Group name	Def. width	Window names
`modalSmallWidth`	570 px	`realise-notes`, `color-picker`, `confirmation`, `demo-message`, `register`, `windowing-edit`, `share-dicom-library`, `slab-thickness`, `calibration`, `add-keyboard-shortcut`, `mark-key-object`, `save-key-objects`, `about`, `uploader-modal`
`modalMediumWidth`	750 px	`hanging-protocol-edit`, `eula`, `forward`, `fusion`, `patient-history`, `patient-studies`, `plugin-content`, `series-list`, `hanging-protocol-group`, `dicom-attributes`, `hanging-protocol-change`, `edit-shortcuts`. *It’s also the default width for any undocumented windows.*
`modalExtraMediumWidth`	850 px	`report-modal`, `template-modal`
`modalLargeWidth`	1200 px	`plugin-content`
`modalWideWidth`	90% of screen	`report`

5.1.3. EcgAnnotations.json

The file ecgAnnotations.json (sys/settings/ecgAnnotations.json by default) is automatically created if missing when the Java application starts. Like system.json, it can only be edited manually, there is no corresponding page in Settings window. When updating, take care to preserve the syntax (quotes, commas, brackets, etc.); a text editor with JSON syntax checking is recommended.

A minimalistic example of the file is shown below.

The “annotations” element is an array and provides definitions for each group of annotations (3 groups in this example). A group can contain one or more annotations (2, 1 and 1 in this example).

The “schemes” element provides validation schemas. An annotation where concept code meaning, concept code value or units code value is different, will be marked with a red asterisk (*) character.

```
{
    "annotations": [
        [
            {
                "label": "ecgAnnotations.vhr",
                "value": "(2:16016;5.10.2.5-1)",
                "diagnosis": false,
                "group": -1
            },
            {
                "label": "ecgAnnotations.prInterval",
                "value": "(2:15872;5.10.2.1-3)",
                "diagnosis": false,
                "group": -1
            }
        ],
        [
            {
                "label": "ecgAnnotations.qtQtc",
                "value": "(2:16160;5.13.5-11) /(2:16164;2:15876;5.10.2.5-5)",
                "diagnosis": false,
                "group": -1
            }
        ],
        [
            {
                "label": "",
                "value": "",
                "diagnosis": true,
                "group": -1
            }
        ]
    ],
    "schemes": {
        "MDC": [
            {
                "nameCodeValue": [
                    "2:16016"
                ],
                "nameCodeMeaning": [
                    "Ventricular Heart Rate"
                ],
                "unitCodeValue": "{H.B.}/min"
            },
```

(continues on next page)

(continued from previous page)

```
            {
                "nameCodeValue": [
                    "2:15872"
                ],
                "nameCodeMeaning": [
                    "PR Interval",
                    "PR Interval global"
                ],
                "unitCodeValue": "ms"
            }
        ],
        "SCPECG": [
            {
                "nameCodeValue": [
                    "5.10.2.5-1"
                ],
                "nameCodeMeaning": [
                    "Ventricular Heart Rate"
                ],
                "unitCodeValue": "{H.B.}/min"
            },
            {
                "nameCodeValue": [
                    "5.13.5-7"
                ],
                "nameCodeMeaning": [
                    "PR Interval"
                ],
                "unitCodeValue": "ms"
            }
        ]
    }
}
```

- annotations[i][j].label

  Annotation label (name of a concept). Can be a constant, like “Ventricular Heart Rate”, or a two-level identifier from the translations file, like “ecgAnnotations.vhr”.

- annotations[i][j].value

  A fragment in parentheses provides a semicolon-separated list of machine- or human-readable concept identifiers from (0008,0100) Code Value or (0008,0104) Code Meaning. When a match is found in (0040,B020)/[k]/(0040,A043)/[0], a nearby (0040,B020)/[k]/(0040,08EA) attribute is used to extract and format the annotation. The units are added from (0040,08EA) Measurement Units Code Sequence, this can’t be avoided.

  Note: For human readability, some unit codes are automatically converted before displaying:

  - `deg` – to a degree symbol,
  - `{H.B.}/min` – to bpm.

  As of 8.6.0+, these conversions are not configurable.

  Multiple fragments can be defined. The text that surrounds them is displayed as is.

- annotations[i][j].diagnosis

If true, .label and .value are ignored (should be empty); the annotations are taken from every (0040,B020)/[k]/(0070,0006). Default is false.

- annotations[i][j].group

  If -1 (default), the annotations are searched among every (0040,B020)/[k]. If zero and more, then (0040,B020)/[k]/(0040,A180) must match this value.

- schemes[NAME][i].nameCodeValue

  The expected contents of (0040,B020)/[k]/(0040,A043)/[0]/(0008,0100).

  NAME is for (0040,B020)/[k]/(0040,A043)/[0]/(0008,0102).

- schemes[NAME][i].nameCodeMeaning

  The expected contents of (0040,B020)/[k]/(0040,A043)/[0]/(0008,0104).

- schemes[NAME][i].unitCodeValue

  The expected contents of (0040,B020)/[k]/(0040,08EA)/[0]/(0008,0100).

5.1.4. The ECG print template

The ECG print template consists of two files, `ecg-print-template.html` and `ecg-print-template.css`, in the directory sys/ecg under configRoot. For security purposes, both files are sanitized to remove potentially unsafe parts; see the following sub-chapters for details.

MedDream is bundled with two example files that correspond to defaults hardcoded in the frontend: if some file is missing, or can't be read, or the backend reports an error while returning both files, then MedDream will still use the same default content. **These external resources can be edited for further customization.** Please keep a backup of the originals, as these examples aren't created automatically.

The HTML file contains {{placeholders}} that will be replaced with corresponding study-related data or findings. Supported *simple placeholders* are:

- {{patientName}} – DICOM tag (0010,0010)
- {{patientId}} – DICOM tag (0010,0020)
- {{studyDateTime}} – combines the DICOM tags (0008,0020) and (0008,0030)
- {{interpretationText}} – 1st entry from DICOM tags (0040,B020)/[k]/(0070,0006); see also entries of ecgAnnotations.json with .diagnosis=true
- {{frequency}} – DICOM tag (003A,001A)
- {{channelCount}} – DICOM tag (003A,0005)
- {{mmPerSecond}} – horizontal axis scale selected in the viewport (5, 10, 15, . . . mm/s) without units
- {{mmPerMillivolt}} – vertical axis scale selected in the viewport (5, 10, 15, . . . mm/mV) without units
- {{cutOffFrequency}} – the cut-off filter frequency selected in the viewport (the same text as seen by the user: "-", "20 Hz", etc) without units

Their values are HTML-escaped for security reasons, therefore it won't be possible to have additional HTML formatting inside the value, etc.

There are also a few *block placeholders* replaced with predefined pieces of HTML:

- {{measurementsHtml}} – a table with measurements
- {{interpretationHtml}} – {{interpretationText}} wrapped with , or an empty string
- {{diagnosisBlock}} – 2nd and following entries from DICOM tags (0040,B020)/[k]/(0070,0006)
- {{techSpecsHtml}} – constructed from {{mmPerSecond}}, {{mmPerMillivolt}}, {{cutOffFrequency}}, {{frequency}}, {{channelCount}}

Furthermore, a custom {{measurementsHtml}} could be built from the following:

- {{vhrLabel}} – a translation code “ecgAnnotations.vhr” that is automatically replaced with a corresponding text from translations.json, like “Ventricular Heart Rate”
- {{vhrValue}} – see .value in an ecgAnnotations.json entry where .label is “ecgAnnotations.vhr”
- {{prIntervalLabel}}, {{prIntervalValue}} – related to .label=”ecgAnnotations.prInterval”
- {{qrsDurationLabel}}, {{qrsDurationValue}} – related to .label=”ecgAnnotations.qrsDuration”
- {{qtQtcLabel}}, {{qtQtcValue}} – related to .label=”ecgAnnotations.qtQtc”
- {{prAxisLabel}}, {{prAxisValue}} – related to .label=”ecgAnnotations.prtAxis”

Warning: The above-mentioned {{vhrLabel}} ... {{prAxisValue}} will work correctly only if ecgAnnotations.json doesn’t specify custom values (free-form text) in the .label entries. This is a known limitation as of 8.9.0+.

An element with the attribute `class="ecg-patient-details"` receives the rendered ECG image. Therefore, its position determines where the ECG will be visible. If such an element is missing, the browser console will show an error and the hardcoded HTML+CSS will be used.

In order to preview the styled HTML file before using it with MedDream, you can insert the following first line and open with a browser directly:

```
<link rel="stylesheet" href="ecg-print-template.css">
```

This way you’ll at least see the overall layout and styling, with literal {{placeholders}} instead of substituted values.

The following example doesn’t use any block placeholders, only the simple ones:

```
<div class="ecg-page">
    <div class="ecg-header">
        <div class="ecg-patient-name">{{patientName}}</div>
        <div>ID: {{patientId}} - {{studyDateTime}}</div>
        <div style="display: flex; gap: 10mm; margin-top: 2mm;">
            <span>{{vhrLabel}}: {{vhrValue}}</span>
            <span>{{prIntervalLabel}}: {{prIntervalValue}}</span>
            <span>{{qrsDurationLabel}}: {{qrsDurationValue}}</span>
            <span>{{qtQtcLabel}}: {{qtQtcValue}}</span>
            <span>{{prAxisLabel}}: {{prAxisValue}}</span>
        </div>
    </div>
    <div class="ecg-image-container"></div>
    <div class="ecg-footer">
        <span>{{mmPerSecond}} mm/s</span>
        <span>{{mmPerMillivolt}} mm/mV</span>
        <span>{{cutOffFrequency}} Hz</span>
        <span>{{frequency}} Hz</span>
        <span>Channels: {{channelCount}}</span>
    </div>
</div>
```

5.1.4.1. Allowed HTML tags

Category	Tags
Layout / containers	div, span, p, br, hr
Headings	h1, h2, h3, h4, h5, h6
Tables	table, thead, tbody, tfoot, tr, td, th, caption, colgroup, col
Lists	ul, ol, li, dl, dt, dd
Inline formatting	strong, b, em, i, u, s, sub, sup, small
Media	img
Links	a
Text blocks	blockquote, pre, code
Semantic HTML5	header, footer, section, article, nav, aside, main, figure, figcaption

Only these tags survive sanitization. All other tags (especially <script, <style>, <iframe>, <svg>, <math>, <form>, <object>, <embed>, <link>, <meta>, <base>, <template>, etc.) are removed.

5.1.4.2. Allowed HTML attributes

Attribute	Notes
class	CSS class names
id	Element identifiers
style	Inline CSS (stripped if value contains javascript:)
colspan, rowspan, scope	Table cell attributes
src	Image source (stripped if value contains javascript:)
alt	Image alt text
width, height	Dimension attributes
href	Link target (stripped if value contains javascript:)
title	Tooltip text
lang, dir	Language and text direction
data-*	Any data attribute (matched by prefix)

Only these attributes survive sanitization. Event handler attributes (any attribute starting with on) and any unlisted attributes are removed.

Inline style attributes with normal CSS properties (e.g. style="display: flex") are preserved.

data: URIs in src attributes are preserved because they're needed for ECG image injection.

External image URLs in src attributes are preserved so that customers may embed logos.

5.1.4.3. Allowed CSS attributes

Known dangerous attributes are removed:

- expression() — IE-specific JavaScript execution in CSS
- javascript: in any url() value
- @import url("https://...") — external stylesheet imports
- behavior: and -moz-binding: — legacy browser-specific script execution vectors

Safe CSS features are fully preserved: @page, @media, class selectors, flexbox, grid, etc.

5.1.5. Dicomprint.json

The file dicomprint.json is edited interactively in the Settings window when features.dicomPrintSettings=true. (It is sys/settings/dicomprint.json by default, and USER_dicomprint.json or GROUP_g_dicomprint.json nearby have a higher priority.)

However, as of 8.9.0+, only a small subset of supported settings are editable in the Settings window: the rest is treated as "very rare" and must be edited manually. When updating, take care to preserve the syntax (quotes, commas, brackets, etc.); a text editor with JSON syntax checking is recommended.

The supported settings are listed below.

JSON field	Meaning (with defaults if applicable)
numberOfInstancesToPrintOut	Number of instances allowed to add to print. Default: 1.
devices[N].id	DICOM Print device id. Must be unique, or else MedDream attempts to modify the duplicate value.
devices[N].printScpIp	DICOM Print device Hostname or IP address
devices[N].printScpPort	DICOM Print device port
devices[N].printScpAet	DICOM Print device AET
devices[N].description	DICOM Print device description
devices[N].proxyHost	[Optional] DICOM Print proxy Hostname or IP address
devices[N].proxyPort	[Optional] DICOM Print proxy port
devices[N].connectTimeout	[Optional] Timeout in ms for creating a TCP connection to the device. Default: 10000.
devices[N].requestTimeout	[Optional] Timeout in ms for receiving A-ASSOCIATE-RQ. Default: 10000.
devices[N].acceptTimeout	[Optional] Timeout in ms for A-ASSOCIATE-AC. Default: 10000.
devices[N].releaseTimeout	[Optional] Timeout in ms for receiving A-RELEASE-RP. Default: 0.
devices[N].responseTimeout	[Optional] Timeout in ms for receiving other outstanding DIMSE RSPs than C-MOVE or C-GET RSPs. Default: 0.
devices[N].retrieveTimeout	[Optional] Timeout in ms for receiving outstanding C-MOVE or C-GET RSPs. Default: 0.
devices[N].idleTimeout	[Optional] Timeout in ms for aborting idle Associations. Default: 0.
devices[N].sendPDULength	[Optional] Set communication send PDU length. Default: 0, silently changed to 16378.
devices[N].receivePDULength	[Optional] Set communication receive PDU length. Default: 0, silently changed to 16378.
devices[N].connectionTransferSyntax	[Optional] TransferSyntax to use for DICOM print communication. Options: 1.2.840.10008.1.2 (default) or 1.2.840.10008.1.2.1.
devices[N].allowColorPrint	[Optional] Allow to send RGB image pixel data. Reserved and not used as of 8.9.0+ (no color printing support so far).
devices[N].supportedSessionPrinting	[Optional] Enable send N-ACTION with BasicFilmSession to print all sent pages (if multiple pages) or to make print with every BasicFilmBox page. Default: true.
devices[N].maxPrintSessions	[Optional] Print this number of print jobs, then wait (a printer limitation according to its DICOM Conformance Statement). Default: 6.
devices[N].sessionOptions	[Optional] Define print session options. Default: `{}`.
devices[N].sessionOptions.mediumType	[Optional] Medium Type (2000,0030). Not necessarily supported, and some printers only accept the same value as configured at their side. Options: PAPER, CLEAR FILM, BLUE FILM, MAMMO CLEAR FILM, MAMMO BLUE FILM.
devices[N].sessionOptions.priority	[Optional] Print Priority (2000,0020). Not necessarily supported, and some printers only accept the same value as configured at their side. Options: HIGH, MED, LOW.

continues on next page

Table 1 – continued from previous page

JSON field	Meaning (with defaults if applicable)
devices[N].sessionOptions .filmDestination	[Optional] Film Destination (2000,0040). Not necessarily supported, and some printers only accept the same value as configured at their side. Options: MAGAZINE, PROCESSOR.
devices[N].sessionOptions .numOfCopies	[Optional] Number of Copies (2000,0010). Not necessarily supported, and some printers only accept the same value as configured at their side. Default: 1.
devices[N].sessionOptions .memoryAllocation	[Optional] Memory Allocation (2000,0060). Amount of memory allocated for the film session, in KB. If supported by printer and needed.
devices[N].sessionOptions .maxMemoryAllocation	[Optional] Upper cap on the previous value, `memoryAllocation`. Larger values will be clamped to this one.
devices[N].sessionOptions.ownerId	[Optional] Owner ID (2100,0160). Identification of the owner of the film session. If supported by printer and needed.
devices[N].filmOptions	[Optional] Define printer film options. Default: `{}`.
devices[N].filmOptions .supportedImageDisplayFormats	[Optional] Define the list of Image Display Format (2010,0010) to select. Options: STANDARD (default), ROW, COL, SLIDE, SUPERSLIDE, CUSTOM.
devices[N].filmOptions .imageDisplayFormatMaxCol	[Optional] Define max columns(C) used with Image Display Format. E. g., STANDARD\C,R; COL\C1,C2,C3,... STANDARD\5,6 -> STANDARD\4,4; COL\5,8,6 -> COL\4,4,4. Default: 6.
devices[N].filmOptions .imageDisplayFormatMaxRow	[Optional] Define max rows(R) used with Image Display Format. E. g., STANDARD\C,R; ROW\R1,R2,R3,... STANDARD\2,8 -> STANDARD\2,4; ROW\5,8,6 -> ROW\4,4,4. Default: 6.
devices[N].filmOptions .imageDisplayFormatCols	[Optional] Default columns(C) to use with Image Display Format. Can be array of columns, e. g. [10, 11]. Default: 1.
devices[N].filmOptions .imageDisplayFormatRows	[Optional] Default rows(R) to use with Image Display Format. Can be array of rows, e. g. [10, 11]. Default: 1.
devices[N].filmOptions .supportedFilmSizeIds	[Optional] Define values for Film Size ID (2010,0050). Options to add: 8_5INX11IN (not included by default), 8INX10IN, 10INX12IN, 10INX14IN, 11INX14IN, 11INX17IN, 14INX14IN, 14INX17IN, 24CMX24CM, 24CMX30CM, A4, A3.
devices[N].filmOptions .imageDisplayFormat	[Optional] Default Image Display Format (2010,0010). Options: STANDARD, ROW, COL, SLIDE, SUPERSLIDE, CUSTOM.
devices[N].filmOptions.filmOrientation	[Optional] Default Film Orientation (2010,0040). Options: PORTRAIT, LANDSCAPE.
devices[N].filmOptions.filmSizeId	[Optional] Default Film Size ID (2010,0050). Options: 8INX10IN, 8_5INX11IN, 10INX12IN, 10INX14IN, 11INX14IN, 11INX17IN, 14INX14IN, 14INX17IN, 24CMX24CM, 24CMX30CM, A4, A3. If supported by printer and needed.
devices[N].filmOptions .magnificationType	[Optional] Magnification Type (2010,0060). Interpolation type by which the printer magnifies or decimates the image in order to fit the image in the image box on film. Options: REPLICATE, BILINEAR, CUBIC, NONE. If supported by printer and needed.
devices[N].filmOptions.smoothingType	[Optional] Smoothing Type (2010,0080). Further specifies the type of the interpolation function. Values are defined in Conformance Statement. Only valid for Magnification Type (2010,0060) = CUBIC. If supported by printer and needed.
devices[N].filmOptions.borderDensity	[Optional] Border Density (2010,0100). Density of the film areas surrounding and between images on the film. Options: BLACK, WHITE. If supported by printer and needed.
devices[N].filmOptions .emptyImageDensity	[Optional] Empty Image Density (2010,0110). Density of the image box area on the film that contains no image. Options: BLACK, WHITE. If supported by printer and needed.

continues on next page

Table 1 – continued from previous page

JSON field	Meaning (with defaults if applicable)
devices[N].filmOptions.minDensity	[Optional] Min Density (2010,0120). Minimum density of the images on the film, expressed in hundredths of OD. If supported by printer and needed.
devices[N].filmOptions.maxDensity	[Optional] Max Density (2010,0130). Maximum density of the images on the film, expressed in hundredths of OD. If supported by printer and needed.
devices[N].filmOptions.trim	[Optional] Trim (2010,0140). Specifies whether a trim box shall be printed surrounding each image on the film. Options: YES, NO. If supported by printer and needed.
devices[N].filmOptions .requestedResolutionID	[Optional] Requested Resolution ID (2020,0050). Specifies the resolution at which images in this Film Box are to be printed. Options: STANDARD, HIGH. If supported by printer and needed.

Below is an example JSON file. The last entry, “usable_sample”, corresponds to options that are edited in the Settings window.

```
{
  "numberOfInstancesToPrintOut": 10,
  "devices": [
    {
      "id": "min_sample",
      "printScpIp": "127.0.0.1",
      "printScpPort": 11114,
      "printScpAet": "DCM4CHEE11",
      "description": "test1"
    },
    {
      "id": "max_sample",
      "printScpIp": "127.0.0.1",
      "printScpPort": 11114,
      "printScpAet": "DCM4CHEE11",
      "description": "test1",
      "proxyHost": "proxy_host",
      "proxyPort": 12354,
      "connectTimeout": 10000,
      "requestTimeout": 10000,
      "acceptTimeout": 10000,
      "releaseTimeout": 100,
      "responseTimeout": 100,
      "retrieveTimeout": 100,
      "idleTimeout": 100,
      "sendPDULength": 16378,
      "receivePDULength": 16378,
      "connectionTransferSyntax": "1.2.840.10008.1.2",
      "allowColorPrint": true,
      "supportedSessionPrinting": true,
      "maxPrintSessions": 2,
      "sessionOptions": {
        "mediumType": "CLEAR FILM",
        "priority": "MED",
        "filmDestination": "PROCESSOR",
        "numOfCopies": 1,
        "maxMemoryAllocation": 1024,
        "ownerId": "owner"
      },
```

(continues on next page)

(continued from previous page)

```
      "filmOptions": {
        "supportedImageDisplayFormats": [
          "STANDARD",
          "ROW",
          "COL",
          "SLIDE",
          "SUPERSLIDE",
          "CUSTOM"
        ],
        "imageDisplayFormatMaxCol": 4,
        "imageDisplayFormatMaxRow": 5,
        "imageDisplayFormatCols": 2,
        "imageDisplayFormatRows": 2,
        "supportedFilmSizeIds": [
          "8INX10IN",
          "10INX12IN",
          "10INX14IN",
          "11INX14IN",
          "11INX17IN",
          "14INX14IN",
          "14INX17IN",
          "24CMX24CM",
          "24CMX30CM",
          "A4",
          "A3"
        ],
        "imageDisplayFormat": "STANDARD",
        "filmOrientation": "PORTRAIT",
        "filmSizeId": "8INX10IN",
        "magnificationType": "8INX10IN",
        "smoothingType": "CUBIC",
        "borderDensity": "WHITE",
        "emptyImageDensity": "WHITE",
        "minDensity": 10,
        "maxDensity": 10,
        "trim": "YES",
        "requestedResolutionID": "STANDARD"
      }
    },
    {
      "id": "usable_sample",
      "printScpIp": "127.0.0.1",
      "printScpPort": 11114,
      "printScpAet": "DCM4CHEE11",
      "description": "test1",
      "supportedSessionPrinting": true,
      "maxPrintSessions": 2,
      "sessionOptions": {
        "mediumType": "CLEAR FILM",
        "priority": "MED",
        "filmDestination": "PROCESSOR",
        "numOfCopies": 1
      },
      "filmOptions": {
        "imageDisplayFormatMaxCol": 4,
        "imageDisplayFormatMaxRow": 5,
```

(continues on next page)

(continued from previous page)

```
        "imageDisplayFormatCols": 2,
        "imageDisplayFormatRows": 2,
        "supportedFilmSizeIds": [
          "8INX10IN",
          "10INX12IN"
        ],
        "imageDisplayFormat": "STANDARD",
        "filmOrientation": "PORTRAIT",
        "filmSizeId": "8INX10IN"
      }
    }
  ]
}
```

5.1.6. Built-in authentication and authorization

MedDream supports a few universal authentication/authorization methods. They can be used when instructions for a particular PACS integration do not offer a dedicated (more proper) method.

Most methods are designed for the interactive login. As a result, the login form can be opened only when a particular method is chosen (for example, `spring.profiles.include = auth-pacsone`).

For authorization you can choose from the following list of permissions:

- `ADMIN` – Settings and Register;
- `SEARCH` – the Search window;
- `PATIENT_HISTORY` – Patient History;
- `UPLOAD_DICOM_LIBRARY` – Share (Share via DICOM Library window);
- `EXPORT_ISO` – the Export function related to “Download” and “Burn”;
- `EXPORT_ARCH` – the Export function related to “Save”;
- `FORWARD` – the Forward function;
- `UPLOAD_DICOM_FILES` – the Upload functionality (must also be enabled globally via features.uploadDicomToPacs in system.json);
- `REPORT_VIEW` – read-only access to Reporting function. Reporting still doesn’t work (corresponding buttons remain hidden) if the type of the license is Lite or Enterprise;
- `REPORT_UPLOAD` – full access to Reporting function (does not include “remote reports”). Also requires a Premium license;
- `KO_PR_VIEW` – allows viewing KO and PR objects present in study, given features.checkKOPRPermissions=true in system.json;
- `KO_PR_EDIT` – allows creating KO and PR objects, given features.checkKOPRPermissions=true in system.json and type of the license not Lite/Enterprise;
- `ANONYMOUS_VIEW` – on-the-fly anonymization of visible patient data (except PDFs, SRs and forwarded data);
- `DOCUMENT_VIEW` – ability to view Structured Report and PDF documents (however they are forwarded regardless of this permission);
- `BOUNDING_BOX_VIEW` – allows display of annotations saved by Segmentation > Bounding Box;
- `BOUNDING_BOX_EDIT` – full access to Segmentation > Bounding Box and visibility of this button. Also requires features.boundingBoxAnnotations=true in system.json, and type of the license must be not Lite/Enterprise;
- `FREE_DRAW_VIEW` – allows display of annotations saved by Segmentation > Free Draw;

- FREE_DRAW_EDIT – full access to Segmentation > Free Draw and visibility of this button (subject to features.boundingBoxAnnotations and non-Lite/Enterprise license type, too);
- SMART_DRAW_VIEW allows display of annotations saved by Segmentation > Smart Draw;
- SMART_DRAW_EDIT – full access to Segmentation > Smart Draw and visibility of this button (subject to features.boundingBoxAnnotations and non-Lite/Enterprise license type, too);
- COPY_TO_DICOM – display of the button "Save viewport content as secondary capture DICOM" (there is also a global setting, features.viewToDICOM=true, in system.json) and support of Montage functionality. The button is still not displayed for a Lite/Enterprise license;
- USER_SETTINGS – allows the user to save settings to a customized file (customizable files are global.json, hangingProtocols.json, hangingProtocolsV2.json, shortcuts.json, dicomprint.json),. Without this permission, a customized file can only be read;
- CLEAR_CACHE – allows the user to use the "Clear cache" function (there also is a global setting, features.clearCache, in system.json);
- SHORTCUTS_EDIT – allows the user to edit the shortcuts (this permission is required even for users that have ADMIN). Ignored in case of a Lite license;
- HANGING_PROTOCOLS_EDIT – allows editing and quick saving of hanging protocols for users that don't have ADMIN (there is also a global setting, features.hangingProtocols, in system.json);
- PACSONE_VIEW_ONLY_PUBLIC – an opposite of the legacy VIEW_PRIVATE if the message "Limited view access" is still required.

Warning: When upgrading to 8.2+, it is advised to remove obsolete permissions 3D_RENDERING and VIEW_PRIVATE from all possible places of configuration:

- authorization.defaultLoginPermissions setting
- authorization.defaultHisPermissions setting
- authorization.users[].role settings
- authorization.remapRoles.* settings
- embedded authentication database for the auth-meddream profile

For some time MedDream will gracefully ignore this permission if left in configuration. However, in the future the value will become really unknown and might crash the backend.

Warning: DOCUMENT_VIEW must be present for pre-7.5.1 functionality. Currently MedDream does not support permissions of "deny" type (existing ones are of "grant" type), therefore it is not possible to grant DOCUMENT_VIEW for all users and afterwards deny for a few ones.

Warning: Take caution with DOCUMENT_VIEW together with ANONYMOUS_VIEW. Anonymization of PDFs and SRs is not supported, therefore DOCUMENT_VIEW will leak sensitive patient data to users who do not see it normally.

Absence of DOCUMENT_VIEW only prevents interactive display and Export of this data. Forward still shares all DICOM objects in the study and so might present a workaround for users who don't have this permission.

Warning: REPORT_UPLOAD and REPORT_VIEW (also BOUNDING_BOX_EDIT and BOUNDING_BOX_VIEW, etc.) are mutually exclusive, although not all authentication methods enforce that. To avoid unexpected behavior, especially in future MedDream versions after the upgrade, do not assign both. This includes attempts to add . . . _VIEW to all users, then "upgrade" to read/write mode for selected users.

Until MedDream does not support deny-type permissions, it does not clean up conflicting or lesser-priority permissions.

Note: 8.2+ contains an inverted logic regarding the now obsolete `VIEW_PRIVATE` permission. It behaves as if that permission is always present; the message "Limited view access" is automatically displayed only when using PacsOne plugins with auth-pacsone and the account configured in PacsOne user administration doesn't have the "View private" privilege.

It is important to remember that in all other situations, like authentication other than auth-pacsone, or using a non-PacsOne plugin, that message was misleading (and still is): other plugins do not implement a corresponding filter, while other authentication types do not support First Name and Last Name of the currently logged in user, that are needed by the filter in PacsOne plugin.

For an unlikely event where the end users are expecting this message despite usage of a different authentication or a non-PacsOne plugin, you can get it back by adding the `PACSONE_VIEW_ONLY_PUBLIC` permission for corresponding users.

Starting from 8.5.0, due to default features.checkKOPRPermissions=false in system.json, KO_PR_VIEW and KO_PR_EDIT are initially present for all users. If there is a need to restrict KO/PR functionality, then change features.checkKOPRPermissions to true and assign KO_PR_VIEW or KO_PR_EDIT only to eligible users.

5.1.6.1. In-memory storage

Provides both authentication (username/password) and authorization (permissions). All this data must be entered into application.properties, in plain text form.

The `spring.profiles.include` setting must include `auth-inmemory`. 7.5.2+ does this by default and includes a single user named "demo", with a password known in advance that must be changed during the installation.

Elements of `authentication.inmemory.users[]` describe the credentials. Elements of `authorization.users[]` link to credentials and list the permissions; the per-user `accessibleStorages` setting can restrict the list of allowed Multi-PACS plugin instances (beware: values that don't match the .id parameter of a configured plugin will stop application loading). `authorization.defaultLoginPermissions` lists permissions common to everyone.

The parameter `accessibleStorages` can also be used at group level. The group is referenced by `authorization.users[].groups` and defined in `authorization.groups[]`.

Example with two users:

```
spring.profiles.include = auth-inmemory
authentication.inmemory.users[0].userName = user1
authentication.inmemory.users[0].password = pass1
authentication.inmemory.users[1].userName = user2
authentication.inmemory.users[1].password = pass2
authorization.users[0].userName = user1
authorization.users[0].role = ADMIN,PATIENT_HISTORY,REPORT_UPLOAD,\
    UPLOAD_DICOM_LIBRARY,DOCUMENT_VIEW
authorization.users[0].groups = administrators
authorization.users[1].userName = user2
authorization.users[1].role = ANONYMOUS_VIEW
authorization.users[1].accessibleStorages = PacsOneRouter,MainArchive
authorization.defaultLoginPermissions = SEARCH
authorization.groups[0].name = administrators
authorization.groups[0].accessibleStorages = PacsOneRouter,MainArchive,ResearchPacs
```

5.1.6.2. Embedded database storage

Keeps credentials and permissions in a built-in H2 database. Passwords are salted and hashed.

There is no GUI so far, management involves running SQL commands in the H2 Console.

After fresh installation of MedDream, make sure that application.properties contains

```
spring.profiles.include = auth-meddream
```

then do the following:

1. Try to log into MedDream. Make sure the login failed with "Access Denied" error and the log contains messages "Query returned no results for user . . . ", "User . . . not found".

   This first attempt will be unsuccessful as the database doesn't contain any accounts. However, the file sys/databases/authdb.mv.db with all relevant tables will be automatically created for blank H2 user and password values.

   If the logged error is different (for example, you've got a ClassCastException instead), then clear all cookies **and saved passwords** and try again.

2. Open H2 Console with the database sys/databases/authdb and **empty** username/password (see 5.1.14 Opening the H2 Console for details).

3. In the Console, make sure the database contains the tables "USERS" and "PERMISSIONS", then use the following SQL expressions to create new users:

   ```
   INSERT INTO users VALUES('LOGIN_NAME', 'ENCODED_PASSWORD', 1);
   INSERT INTO permissions VALUES('LOGIN_NAME', 'PERMISSION_NAME');
   ```

   ENCODED_PASSWORD consists of a prefix (like `{bcrypt}`) that identifies the algorithm, and the encoded value. For the bcrypt algorithm (recommended) you can use online services like https://www.javainuse.com/onlineBcrypt (do not forget to add the prefix afterwards). Example: `INSERT INTO users VALUES('admin', '{bcrypt}$2a$10$DqbOQDCzx.inz.anVIYUoOiw.33lUiOn.PIhLWZVlIKARQEJNsjzC', 1)`.

   The third column can either be `1` or `0`. Zero means a disabled account (login not possible).

   `PERMISSION_NAME` can be ADMIN, FORWARD, EXPORT_ISO, etc. Use multiple "INSERT INTO permissions" statements when adding more than one permission.

4. As shown in application.SAMPLE.properties, you can use authorization.defaultLoginPermissions to assign permissions that are needed for every user account.

Note: Forgetting Step 1 will have the following consequences:

- Step 2 will also create authdb.mv.db but without the relevant tables (unless you modify the JDBC URL in a particular cumbersome way). This state is unsuitable for continuing with Step 3;
- you can accidentally specify a non-blank username or password during Step 2, and not get any warning or error; however afterwards Core will be unable to access the database.

Provided that credentials were empty, you can retry Step 1 later which will populate the tables. Stopping Core and removing authdb.mv.db might also help.

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

Unlike in other mechanisms, the per-user accessibleStorages setting can't be stored in the database.

5.1.6.3. LDAP storage

Verifies credentials against an LDAP server (you can use third-party tools to manage credentials and groups). Mapping between user's LDAP groups and MedDream permissions is configured in application.properties. The latter can also append permissions to individual users.

The `spring.profiles.include` setting must include `auth-openldap` or `auth-activedirectory` when using the corresponding server. There also is a variant of AD for servers with mandatory GSSAPI/Kerberos, `auth-kerberosldap`.

The main setting is `authentication.ldap.url`: hostname, port and base DN. For SSL you need to use "ldaps://" for the protocol, and likely a different port (for example, 636). Other settings depend on the profile.

Example for OpenLDAP:

```
spring.profiles.include = auth-openldap
authentication.ldap.url = ldap://MY-SERVER:389/ou=DEPT-NAME,dc=COMPANY-NAME
authorization.defaultLoginPermissions = SEARCH,DOCUMENT_VIEW
authorization.remapRoles.DOCTORS-GROUP-NAME = CLEAR_CACHE,EXPORT_ARCH,\
    EXPORT_ISO,PATIENT_HISTORY,UPLOAD_DICOM_LIBRARY
authorization.remapRoles.ADMINS-GROUP-NAME = ADMIN
authorization.remapRoles.TECHNICIANS-GROUP-NAME = FORWARD
authorization.users[0].userName = USER-NAME-1
authorization.users[0].role = ADMIN
authorization.users[1].userName = USER-NAME-2
authorization.users[1].role = FORWARD,REPORT_UPLOAD,BOUNDING_BOX_EDIT
authorization.users[1].accessibleStorages = st3
```

To authenticate, MedDream binds to DN `uid=LOGIN-NAME,ou=People,ou=DEPT-NAME,dc=COMPANY-NAME` (the part `uid=...,ou=People` is hardcoded) using `LOGIN-NAME` and password from the login form.

Then it collects groups to which this user belongs, by searching for group objects exactly in `ou=groups,ou=DEPT-NAME,dc=COMPANY-NAME` with a filter `(memberUid=LOGIN-NAME)`; both `ou=groups` and the filter are hardcoded.

Finally, authorization.remapRoles.* and authorization.users[] are consulted for corresponding MedDream permission sets: the group `DOCTORS-GROUP-NAME` receives `CLEAR-CACHE` etc., and the user `USER-NAME-1` additionally receives `ADMIN`.

Example for Active Directory:

```
spring.profiles.include = auth-activedirectory
#spring.profiles.include = auth-kerberosldap
authentication.ldap.url = ldap://MY-SERVER:389/ou=DEPT-NAME,dc=COMPANY-NAME
authentication.ldap.bindUserDn = cn=MEDDREAM-APP,ou=SERVICE,ou=DEPT-NAME,dc=COMPANY-NAME
authentication.ldap.bindUserPassword = MEDDREAM-APP-PASS
authentication.ldap.userSearchBase = ou=USERS
authentication.ldap.groupSearchBase = ou=USERS
authorization.defaultLoginPermissions = SEARCH,DOCUMENT_VIEW
authorization.remapRoles.DOCTORS-GROUP-NAME = CLEAR_CACHE,EXPORT_ARCH,\
    EXPORT_ISO,PATIENT_HISTORY,UPLOAD_DICOM_LIBRARY
authorization.remapRoles.ADMINS-GROUP-NAME = ADMIN
authorization.remapRoles.TECHNICIANS-GROUP-NAME = FORWARD
authorization.users[0].userName = USER-NAME-1
authorization.users[0].role = ADMIN,REPORT_VIEW
authorization.users[1].userName = USER-NAME-2
authorization.users[1].role = FORWARD,REPORT_UPLOAD,BOUNDING_BOX_EDIT
authorization.users[1].accessibleStorages = st3
```

To authenticate, MedDream binds to bindUserDn using bindUserPassword, searches for a user object at or below `ou=USERS` (empty userSearchBase is not allowed) with a hardcoded filter `(sAMAccountName=LOGIN-NAME)`, then binds to the found DN using the password from the login form.

Then it extracts group names from this user object. Its attributes `memberOf` (hardcoded) are expected to contain group DNs which start with `cn=GROUP-NAME,...`. Due to technical limitations, the user object found earlier is not reused – a second search is performed at or below groupSearchBase (should obviously be equal to userSearchBase) with the same hardcoded filter `(sAMAccountName=LOGIN-NAME)`.

As with OpenLDAP, authorization.remapRoles.* provide permission sets for entire groups, while authorization.users[] can add more permissions to individual users.

If your server demands GSSAPI, just use the "auth-kerberosldap" profile instead.

Since 8.3 any kind of LDAP storage can authenticate only users that belong to specified groups:

```
# must be in "users" or "doctors" group
authentication.ldap.mandatoryGroups = users,doctors

# must be in "meddream" group, or both in "users" and "doctors" groups
authentication.ldap.mandatoryGroups[0] = meddream
authentication.ldap.mandatoryGroups[1] = users,doctors
```

Traditionally, groups in MedDream are used only to map permissions and do not participate in authentication. This parameter is empty by default and no group membership is enforced. Two-dimensional syntax similar to `validHisParams` is supported: the first level separates elements with logical OR, the second level – with logical AND.

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

Warning: Spaces embedded in role names are ignored. For example, "Super Users" and "SuperUsers" will be treated as the same role.

The setting authorization.ldap.passwordAttribute1 is not recommended since 8.6 and will be removed in the future. You should investigate how to configure your LDAP server in a more secure way. Please inform support@meddream.com if your installation depends on this setting.

5.1.6.4. SAML

Credentials entered in the login form will be validated by an external identity provider.

See https://developer.okta.com/code/java/spring_security_saml for details, however make sure to use `.../saml2/login/okta...` instead of `.../login/saml2/okta...`. Example configuration at MedDream side:

```
spring.profiles.include = auth-saml
spring.security.saml2.relyingparty.registration.okta.assertingparty.metadata-uri =\
    https://dev-0731725.okta.com/app/exk3k4dxuhko35exg5d7/sso/saml/metadata
spring.security.saml2.relyingparty.registration.meddream.acs.location =\
    {baseUrl}/saml2/login/sso/{registrationId}

authorization.defaultLoginPermissions = SEARCH,DOCUMENT_VIEW
authorization.remapRoles.doctor = CLEAR_CACHE,EXPORT_ARCH,EXPORT_ISO,\
    PATIENT_HISTORY,UPLOAD_DICOM_LIBRARY
authorization.remapRoles.administration = ADMIN
authorization.remapRoles.assistant = FORWARD
```

User roles are taken from the SAML attribute "memberOf". When using them, authorization.remapRoles.* is needed for mapping to MedDream permissions.

If HIS integration is needed in parallel to the login form, then you need to use spring.profiles.include = auth-saml-his. (It automatically enables functionality of auth-his, which can't be used explicitly together with auth-saml.)

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

Azure notably requires the "signing.credentials" element. A fragment of the above example would look like this:

```
spring.security.saml2.relyingparty.registration.65a6e97b-d94d-401f-a0f4-ab628cdcb822.acs.
↪location={baseUrl}/saml2/login/sso/{registrationId}
spring.security.saml2.relyingparty.registration.65a6e97b-d94d-401f-a0f4-ab628cdcb822.
↪assertingparty.metadata-uri=https://login.microsoftonline.com/c13d4087-0dce-45a1-9f3a-
↪079768c53769/federationmetadata/2007-06/federationmetadata.xml?appid=65a6e97b-d94d-401f-
↪a0f4-ab628cdcb822
spring.security.saml2.relyingparty.registration.65a6e97b-d94d-401f-a0f4-ab628cdcb822.signing.
↪credentials[0].private-key-location=file:////opt/meddream/private.pem
spring.security.saml2.relyingparty.registration.65a6e97b-d94d-401f-a0f4-ab628cdcb822.signing.
↪credentials[0].certificate-location=file:////opt/meddream/public.pem
```

5.1.6.5. OAuth

The behavior is very similar to the SAML case. Support was tested with Keycloak and Okta.

Example for Okta follows. The user must belong to a group "MD_DOCTOR".

```
spring.profiles.include = auth-oauth2
spring.security.oauth2.client.registration.okta.scope = openid,profile,email,groups
spring.security.oauth2.client.registration.okta.provider = meddream
spring.security.oauth2.client.registration.okta.client-id = VpiK045loHkd5gd5ma7K
spring.security.oauth2.client.registration.okta.client-secret = \
KEfh5XyuWGfInP7CO-zPBsVNMBeCX5cG3DtBrBkkO3fFoXy3-1ePByU-Salr3x9b
spring.security.oauth2.client.registration.okta.client-authentication-method = \
client_secret_basic
spring.security.oauth2.client.registration.okta.authorization-grant-type = authorization_code
spring.security.oauth2.client.registration.okta.roles-claim = groups
spring.security.oauth2.client.provider.meddream.issuer-uri = https://dev-65487606.okta.com
spring.security.oauth2.client.provider.meddream.user-name-attribute = sub
authorization.remapRoles.MD_DOCTOR = SEARCH,EXPORT_ISO,EXPORT_ARCH,FORWARD,REPORT_VIEW,\
REPORT_UPLOAD,UPLOAD_DICOM_LIBRARY,ADMIN
```

The example above allows validation of credentials entered in the login form but isn't compatible with auth-his. If you need to use HIS integration together with the login form, then change "auth-oauth2" to "auth-oauth2-his" (auth-his will be included automatically).

Note: When using auth-oauth2-his with the Amazon Cognito service, add the "cognito" profile, too:

```
spring.profiles.include = auth-oauth2-his,cognito
```

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

5.1.6.6. Database connection

Generic authentication against a DBMS server (you can use third-party DBMS tools to manage credentials). Internal MedDream permissions are assigned to users in application.properties.

Example with two MariaDB users:

```
spring.profiles.include = auth-db-connection
authentication.database.driverClassName = org.mariadb.jdbc.Driver
authentication.database.jdbcUrl = jdbc:mariadb://127.0.0.1:3306/pacs2
authorization.defaultLoginPermissions = SEARCH,DOCUMENT_VIEW
authorization.users[0].userName = user1
authorization.users[0].role = ADMIN,EXPORT_ARCH,REPORT_VIEW,BOUNDING_BOX_VIEW
authorization.users[1].userName = user2
authorization.users[1].role = REPORT_UPLOAD,BOUNDING_BOX_EDIT
authorization.users[1].accessibleStorages = st3
```

`authorization.defaultLoginPermissions` lists permissions common to everyone.

Supported values for `authentication.database.driverClassName`:

- `org.mariadb.jdbc.Driver`,
- `org.postgresql.Driver`,
- `com.microsoft.sqlserver.jdbc.SQLServerDriver`.

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

5.1.6.7. SSO: Kerberos over SPNEGO

When the end users use Active Directory logins under Windows, it's possible to automatically authenticate them in MedDream, too. The browser on the client machine will use the SPNEGO mechanism to get an AD authentication token, which allows MedDream on the server machine to extract user's groups from AD. The mandatoryGroups setting is important unless you want to allow all valid AD users (not only physicians) into MedDream.

Add to application.properties:

```
spring.profiles.include=auth-spnego-krb5
authentication.krb5.url=ldap://test.local:389/DC=test,DC=local
authentication.krb5.servicePrincipal=HTTP/service.test.local@TEST.LOCAL
authentication.krb5.keytabLocation=c:\\meddream\\bbbb.keytab
authentication.krb5.bindUserDn=bbbb@TEST.LOCAL
authentication.krb5.bindUserPassword=bbbb
authentication.krb5.userSearchBase=CN=users
authentication.krb5.groupSearchBase=CN=users
authentication.krb5.userSearchFilter=(userPrincipalName={0})
authentication.krb5.groupSearchFilter=(member={0})
authentication.krb5.mandatoryGroups=doctor
```

Add to the registry in the client machine:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome]
"AuthNegotiateDelegateAllowlist"="*.test.local"
"AuthServerAllowlist"="*.test.local"
```

(continues on next page)

(continued from previous page)

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Edge]
"AuthNegotiateDelegateAllowlist"="*.test.local"
"AuthServerAllowlist"="*.test.local"
```

Create C:\Windows\krb5.ini in the client machine:

```
[libdefaults]
   default_realm = TEST.LOCAL
   dns_lookup_realm = false
   dns_lookup_kdc = false
   ticket_lifetime = 24h
   renew_lifetime = 7d
   forwardable = true
    default_keytab_name = file:/c:/meddream/bbbb.keytab
    allow_weak_crypto = true

    default_tkt_enctypes = aes256-cts-hmac-sha1-96
    default_tgs_enctypes = raes256-cts-hmac-sha1-96
    permitted_enctypes   = aes256-cts-hmac-sha1-96

[realms]
   TEST.LOCAL = {
       kdc = TEST.LOCAL
       admin_server = TEST.LOCAL
   }

[domain_realm]
   .test.local = TEST.LOCAL
   test.local = TEST.LOCAL
```

Here TEST.LOCAL is the AD server.

Add to the hosts file in the client machine:

```
127.0.0.1 service.test.local SERVICE.TEST.LOCAL
service.test.local 127.0.0.1
```

Run at the Active Directory server:

```
setspn -S HTTP/service.test.local test.local\bbbb

ktpass -princ HTTP/service.test.local@test.local -mapuser bbbb@test.local -pass bbbb -crypto␣
↪RC4-HMAC-NT -ptype KRB5_NT_PRINCIPAL -out bbbb.keytab
```

This will create the file “bbbb.keytab” that needs to be copied to c:\meddream\bbbb.keytab in the MedDream machine.

In the browser, MedDream needs to be accessed as http://service.test.local/sso. (This example is dedicated to opening MedDream from its own machine.)

Now the /sso endpoint will result in the Search window, bypassing the login screen. Meanwhile the /?study=... endpoint will open the Viewer window with studies like in usual HIS integration.

For troubleshooting, the `klist` command (one of Kerberos utilities) on the client computer shows active Kerberos tickets. If there are none, then this kind of MedDream SSO won’t work, too.

Note: Do not use auth-his, as it conflicts with auth-spnego-krb5. The alternative, auth-spnego-krb5-his, automatically enables the HIS integration functionality. Furthermore, for /?study=... etc one still needs to

update authentication.his.valid-his-params.

Note: By default, application.SAMPLE.properties enables `auth-inmemory` and this is done on another line. There is no error message if you assign to spring.profiles.include multiple times, and the outcome might be counter-intuitive. Make sure to comment the unneeded line out.

5.1.7. Predefined login credentials

In demo installations it is sometimes desirable to have the login form automatically filled with some working credentials.

Starting from 8.8.0, it is no more required to patch md5/index.html. Use the following in application.properties instead:

```
com.softneta.meddream.login.autoLogin.username=ACCOUNT_USERNAME
com.softneta.meddream.login.autoLogin.password=ACCOUNT_PASSWORD
```

This behavior is enabled by a non-empty `username`. The password can remain empty.

5.1.8. The Reporting function

Reports and attachments are saved into the same H2 database, sys/database/reportdb.*. It is also possible to use MySQL/MariaDB instead.

- The option of MedDream's former PHP backend `$attachment_upload_dir` (config.php) is not implemented yet. Attachment contents are always stored into the database.
- The structure is slightly different from the legacy "studynotes" and "attachment" tables. Access to legacy tables, especially the ones updated by PacsOne GUI, is not possible even after configuring MySQL/MariaDB storage instead of H2.

There are two attachment size limits that you might need to configure. MedDream will inform you which specific limit was exceeded. The size of the largest file, and the total size of all files during the same upload operation, are configured separately:

```
spring.servlet.multipart.max-file-size=1MB
spring.servlet.multipart.max-request-size=10MB
```

8.6+ brings another limit – size of the entire report including embedded pictures:

```
report.maxContentSize=25MB
```

(this is the default). In order to be helpful, it must be smaller than server.tomcat.max-http-form-post-size; and, when reports are stored in MySQL/MariaDB instead of H2, the "max_allowed_packet" database server setting needs to be increased accordingly, too.

Print templates are supported like in 7.1 MedDream and existing ones might work without modification. The file sys/report/default.html is an example with all supported value placeholders. Adjacent files CR.html, XC.html etc. (named after the first modality value in the study), if present, will be used for studies with particular modalities. MedDream 8.6+ can use per-user and per-group print templates, too; the file names are tried in the following order:

- USERNAME_FIRSTMOD.html
- USERNAME_default.html
- GROUPNAME_g_FIRSTMOD.html
- GROUPNAME_g_default.html

- FIRSTMOD.html
- default.html

(USERNAME, GROUPNAME, FIRSTMOD above are placeholders).

Stamps in print templates are expected in form of files sys/report/stamp/USERNAME.jpg.

When Reporting is used in unsafe HIS integration mode (or the token does not include the "user" field), MedDream assigns the same user identity in all saved reports. It can be configured by

```
authentication.his.username=his-user-name
```

5.1.8.1. Migrating the legacy MySQL tables to the 7.5.2+ database

Please contact support@meddream.com for the migration script. It will copy the MySQL database records (and content of externally stored attachment files) to a .sql file, which can afterwards be imported via H2 Console.

5.1.8.2. Copying templates from one user to another (H2 storage)

This task was sometimes needed in PHP-based MedDream and is now possible in 7.5.2+, too.

Open the H2 Console in the browser (for details, see 5.1.14 Opening the H2 Console). Open sys/database/reportdb there (example URL: `jdbc:h2:file:/C:/PROGRA~1/PacsOne/php/meddream/sys/database/reportdb`) with username "sa" and no password.

View contents of the TEMPLATE table. Add the `WHERE "USER"='SOURCE_LOGIN_NAME'` clause (with real username) to the autogenerated SQL statement and confirm that the needed records exist.

Replace the SQL statement with the following, then change DESTINATION_LOGIN_NAME and SOURCE_LOGIN_NAME to real values and execute:

```
insert into template (id, node, name, "USER", content)
select
    nextval('TEMPLATE_SEQ'),
    t.node,
    t.name,
    'DESTINATION_LOGIN_NAME',
    t.content
from
    template t
where
    t."USER" = 'SOURCE_LOGIN_NAME'
```

In a similar fashion, view the TEMPLATE table again and confirm that DESTINATION_LOGIN_NAME now has the needed records, too. At this point a refreshed MedDream Reporting window should also display the corresponding templates.

In general, the settings spring.h2.console.* are also needed for editing the auth-meddream database, while authentication.manager.* are also needed for editing auth-meddream and for various investigations (or additional functionality) via /manage endpoint. If these tasks are not planned immediately, then all four settings should be commented out for security after you finish with templates.

5.1.8.3. Using MariaDB/MySQL/PostgreSQL instead of H2

H2-based report storage works out of the box and is adequate for small institutions. However heavy use of reports (especially with attachments), or the need to access them from third-party software, might call for a different database engine. MedDream 7.6.0+ supports MySQL/MariaDB as an alternative; 8.6.0+ must use the MariaDB connector to access MySQL databases; 8.7.0 brings PostgreSQL support.

We strongly recommend to create a new database dedicated to reports, in order to avoid name conflicts with existing tables.

Relevant settings in application.properties for MariaDB/MySQL:

```
report.datasource.url=jdbc:mariadb://DB_HOST:DB_PORT/DB_NAME
report.datasource.driverClassName=org.mariadb.jdbc.Driver
report.datasource.username=USER_LOGIN
report.datasource.password=USER_PASSWORD
```

Use the following for PostgreSQL:

```
report.datasource.url=jdbc:postgresql://DB_HOST:DB_PORT/DB_NAME
report.datasource.driverClassName=org.postgresql.Driver
report.datasource.username=USER_LOGIN
report.datasource.password=USER_PASSWORD
```

Missing tables are created automatically when MedDream starts. Table structure might be automatically updated during the first run of a newer MedDream version. *The login name USER_LOGIN must initially refer to a privileged user* – with `ALL PRIVILEGES` (or at least REFERENCES, INDEX, ALTER, CREATE, SELECT, INSERT, UPDATE, DELETE), or similarly capable of manipulating the schema. Only after making sure that the entire functionality of reports is working, you can downgrade to a less-privileged user (SELECT, INSERT, UPDATE, DELETE are sufficient).

If you absolutely need to use an existing database, then plan carefully:

1. MAKE A DATABASE BACKUP before the first run with non-H2-based report storage, and from now on – before the first run of a newer MedDream version;
2. names of MedDream-related tables are “report”, “template”, “attachment”, “report_attachment” and “flyway_schema_history”. Make sure your schema does not already contain tables with these names. For example, PacsOne uses the table “attachment” for its own purposes;
3. it is possible to assign new names to most tables *except “flyway_schema_history”*:

   ```
   report.tableNames.report=NEW_REPORT_NAME
   report.tableNames.attachment=NEW_ATTACHMENT_NAME
   report.tableNames.template=NEW_TEMPLATE_NAME
   report.tableNames.report_attachment=NEW_REPORT_ATTACHMENT_NAME
   ```

4. those custom table names are a critical piece of configuration. If you accidentally comment them out, or specify the name of an existing third-party table, then your data is at danger! In the best case nothing serious will happen to a foreign table but MedDream won’t be able to use it due to still incompatible structure.

5.1.8.4. Remote reports

The "Remote reports" functionality requires custom works by the customer and has the following limitations:

- MedDream Premium license is required;
- only the Query/Retrieve integration;
- report status available only after opening the study or in the Patient History window (not in the Search window);
- operation is read-only (no editing, just display);
- supported content types are application/pdf, application/json, application/xml, application/xhtml+xml, image/gif, image/jpeg, image/png, text/html, text/plain, text/xml.

It must be enabled in application configuration:

- specify features.remoteReports=true in system.json (incompatible with features.reports=true);
- add REPORT_VIEW permission for the user (REPORT_EDIT, however, will be ignored).

In the configuration of the QR plugin, a custom DICOM tag number needs to be specified:

```
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.report.tag=CUSTOM_
↪DICOM_TAG_IN_DECIMAL
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.report.type=string
```

The PACS will receive this attribute as a return key, and should return a study-level attribute with this tag. If a non-empty value is found in the response, then the plugin replaces it with the `reportURL` setting after filling the placeholders `{studyUid}` and `{storageId}` (at least the first one should be present):

```
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].reportURL=http://127.0.0.
↪1:8088/report?study={studyUid}&storage={storageId}
#com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].reportURL=/mnt/attached-
↪reports/{studyUid}.pdf
```

reportURL can specify a custom endpoint that returns content of the report attached to the study. The URL can be generic or RESTful. A GET request without authentication will be performed. If the URL doesn't begin with "http", it is treated as a file system path and must be readable by the JVM process on the machine where MedDream is installed.

Additional configuration (optional):

```
report.remoteReports.contentType=text/plain
report.remoteReports.proxyHost=172.30.55.1
report.remoteReports.proxyPort=8088
report.remoteReports.cacheMaxAgeSec=300
report.remoteReports.httpTimeout=5000
report.remoteReports.maxSize=1000000
```

The first step in getting the content type is to use the value returned by the server, or configured as file name extension in `reportURL` when it specifies a file system path. If the server returned some type, or it was successfully determined from the path, then it is validated against the list of supported types. Otherwise (the server didn't return the type, or detection has failed) the `contentType` setting is used (defaults to "application/pdf"). The final value is added to the response from MedDream backend and the browser will react correspondingly.

`proxyHost` and `proxyPort` are only used if both are provided and the port number is larger than zero.

If `httpTimeout` is larger than zero, it is used both as connection timeout and read timeout (in milliseconds). By default, there is no timeout.

A larger than zero `maxSize` imposes a size limit on downloadable or file system resource: there will be no attempts to display a larger file.

If `cacheMaxAgeSec` is larger than zero, then a `Cache-control: max-age=...` header with this value is added to the corresponding MedDream response (from backend to frontend). Might be required when MedDream itself is being accessed via a caching proxy.

5.1.8.5. The default template

When creating a new report, initially it opens in an empty state; if one wants to start from a template, then some manual steps are needed.

This can be changed for all users by storing the chosen template under the username `*defaultForNewReport*`. In case multiple records are present, the most recent one is used (according to the "updated" column).

Suggested steps for H2 are as follows:

1. View contents of the TEMPLATE table for a particular user and write down "ID" of the chosen template (substitute USER_LOGIN_NAME with relevant value):

```
SELECT * FROM TEMPLATE where "USER"='USER_LOGIN_NAME';
```

2. Make a copy for this reserved username:

```
insert into template (id, node, name, "USER", content)
select
    nextval('TEMPLATE_SEQ'),
    t.node,
    t.name,
    '*defaultForNewReport*',
    t.content
from template t
where t.id=ID_OF_CHOSEN_RECORD;
```

   This automatically updates the "updated" column, making the record most recent. After testing the behavior, you can remove the older records. They could be left alone, too, in case there is a need to quickly switch between different default templates by updating the "updated" column of one record.

5.1.9. Adjusting Java memory limits

Some example configurations included in MedDream installation archive specify 1 GB, which is sufficient for typical files.

In general, the application requires **an amount of memory no less than the size of the largest DICOM file**. Use this as a ballpark figure, and add a generous overhead for simultaneous connections. Furthermore, up to `com.softneta.preparation.maxThreadCount` files of the same study will be processed in the background simultaneously, so this should be taken into account, too, when your studies contain multiple moderately-sized files.

Experiments with a single connection show that a single 1.3 GB multiframe file opens rather slowly under a 2 GB memory limit and without problems under a 3 GB limit; consequently, 4 GB would be recommended in this case.

The memory limit is configured by the Java option `-XmxSIZE`, where SIZE can use suffixes "M", "G", etc. (for example, `-Xmx1024M`).

If the same value is also specified with `-XmsSIZE`, then performance might be better as no resources are wasted while growing the heap – it's always of this size. At least 2 GiB is recommended.

Furthermore, even with adequate -Xmx and -Xms values Java might still crash due to "OutOfMemoryError: Direct buffer memory". The solution is to configure a small limit (approx. 1 MiB) via `jdk.nio.maxCachedBufferSize`.

- When you are starting the backend from the console (during 5.1.1 Essential configuration and the first run, or for a quick experiment with a stopped service), then just add `-Xmx4G` etc. as the first option. For example,

```
java -cp MedDream-8.9.1.jar -Xmx4G -Dloader.path=./sys/plugins -DANTLR_USE_
↪DIRECT_CLASS_LOADING=true --add-opens=jdk.management/com.sun.management.
↪internal=ALL-UNNAMED --add-opens=java.base/jdk.internal.misc=ALL-UNNAMED --
↪add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/com.
↪sun.jmx.mbeanserver=ALL-UNNAMED --add-opens=java.base/sun.reflect.generics.
↪reflectiveObjects=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-
↪opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-
↪UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/
↪java.time=ALL-UNNAMED --add-exports=java.desktop/sun.awt.image=ALL-UNNAMED␣
↪org.springframework.boot.loader.launch.PropertiesLauncher
```

```
java -cp MedDream-8.9.1.jar -XX:MaxRAMPercentage=50.0 -Dloader.path=./sys/
↪plugins -DANTLR_USE_DIRECT_CLASS_LOADING=true --add-opens=jdk.management/com.
↪sun.management.internal=ALL-UNNAMED --add-opens=java.base/jdk.internal.
↪misc=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-
↪opens=java.management/com.sun.jmx.mbeanserver=ALL-UNNAMED --add-opens=java.
↪base/sun.reflect.generics.reflectiveObjects=ALL-UNNAMED --add-opens=java.base/
↪java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-
↪opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-
↪UNNAMED --add-opens=java.base/java.time=ALL-UNNAMED --add-exports=java.
↪desktop/sun.awt.image=ALL-UNNAMED org.springframework.boot.loader.launch.
↪PropertiesLauncher
```

(The last version allows Java to utilize 50% of available physical memory.)

- When the service is already set up, then you need to adjust the application command line in the corresponding part of the installation. See the chapter 5.2 Running as a service for possible locations.

Cloud-based installations (especially high-availability installations) will also benefit from the option `-XX:+ExitOnOutOfMemoryError`.

- With this option, if there is an out of memory error, Core will restart automatically and the /manage/health endpoint will duly indicate a temporary downtime. The latter is mandatory for infrastructure like Kubernetes that heavily depends on /manage/health.
- Without this option (default), you might get an error when opening some huge study, but Core will keep running and a subsequent attempt to open a small study will likely succeed – without a need to wait while Core fully starts again. (If it fails anyway, a manual restart will be required as usual.)

Depending on number of concurrent connections, the following options are recommended:

Concurrent connections	JVM parameters related to memory consumption	core-Thread-Count*
1	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx3072m -Xms2048m	20
2	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx3072m -Xms2048m	20
5	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx6144m -Xms2048m	40
10	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx8192m -Xms2048m	40
20	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx12288m -Xms2048m	40
30	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx16384m -Xms2048m	40
60	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xmx20480m -Xms2048m	40
60+	-Djdk.nio.maxCachedBufferSize=1048576 -XX:NativeMemoryTracking=detail - Xms20480m	40

* Specified as `com.softneta.preparation.coreThreadCount` in application.properties.

5.1.10. Enabling SSL

The single Tomcat web server port, `server.port`, can be configured for HTTPS instead of HTTP.

(Option 1) To create a self-signed certificate, you can use `keytool` from JRE:

```
keytool -genkeypair -alias tomcat -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore path/
↪to/keystore.p12 -validity 3650 -storepass PASSWORD
```

It will ask for a few identifying attributes of the certificate; it is possible to just press Enter every time. When asked if the information is correct, type `yes`. At the last prompt (regarding the key password), press Enter, too.

```
What is your first and last name?
    [Unknown]:
What is the name of your organizational unit?
    [Unknown]:
What is the name of your organization?
    [Unknown]:
What is the name of your City or Locality?
    [Unknown]:
What is the name of your State or Province?
    [Unknown]:
What is the two-letter country code for this unit?
    [Unknown]:
Is CN=localhost, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
[no]: yes

Enter key password for <tomcat>
    (RETURN if same as keystore password):
```

For verification, use keytool again:

```
keytool -list -v -storetype pkcs12 -keystore path/to/keystore.p12
```

(Option 2) If you already have a certificate, then conversion to a Java keystore is often still needed. The certificate must be in DER (not PEM) format.

```
keytool -import -alias tomcat -file myCertificate.crt -keystore keystore.p12 -storepass␣
↪PASSWORD
```

To specify the keystore for the JAR, copy its file to configRoot and modify application.properties:

```
server.ssl.key-store-type=PKCS12
server.ssl.key-store=${com.softneta.meddream.configRoot}/keystore.p12
server.ssl.key-store-password=PASSWORD
server.ssl.key-alias=tomcat
security.require-ssl=true
server.port=8443
```

`server.port` is here to minimize confusion: the current default port number, 8080, is normally used with HTTP, not HTTPS.

When not using the well-known HTTPS port 443, you might need to always add `https://` to MedDream URLs from now on.

5.1.10.1. SSL for the bundled Token Service

The Token Service can be configured by the same Spring Boot properties as above in its own application.properties file.

In the simplest scenario, it is sufficient to reuse the same PKCS#12 file with `server.ssl.key-store` and `server.ssl.key-store-password`. The example .properties file has those settings commented out.

Mutual SSL authentication between Token Service and MedDream Core service is possible, too. An example configuration follows; it assumes the certificate was created for "localhost:8443" with passphrase "changeit", and is stored in files clientBob.crt / clientBob.key. This Token Service will accept HTTP, traditional HTTPS and HTTPS with mutual SSL.

application.properties of the Token Service:

```
server.port=8443
server.ssl.enabled=true

server.ssl.key-store=keystore.jks
server.ssl.key-store-password=changeit
server.ssl.key-alias=localhost
server.ssl.key-password=changeit

server.ssl.trust-store=truststore.jks
server.ssl.trust-store-password=changeit
server.ssl.client-auth=need
```

MedDream Core keystore file is created as:

```
keytool -import -file rootCA.crt -alias rootCA -keystore client-truststore.jks -storepass␣
↪changeit
```

MedDream Core truststore file is created as:

```
keytool -importkeystore -deststorepass changeit -destkeypass changeit -destkeystore client-
↪keystore.jks -srckeystore clientBob.p12 -srcstoretype PKCS12 -srcstorepass changeit -alias␣
↪clientBob
```

MedDream Core process is started with JRE arguments

```
-Djavax.net.ssl.keyStoreType=JKS -Djavax.net.ssl.keyStore=client-keystore.jks -Djavax.net.
↪ssl.keyStorePassword=changeit -Djavax.net.debug=all -Djavax.net.ssl.trustStore=client-
↪truststore.jks -Djavax.net.ssl.trustStorePassword=changeit
```

A third party HTTP client that creates tokens must know about rootCA.crt, too.

5.1.11. Setting up ATNA event logging

Implementing ATNA (Audit Trail and Node Authentication) logging is a critical step in ensuring the security, compliance, and traceability of healthcare data within your system. Here's why setting up ATNA logs is essential:

1. **Data Security and Privacy Protection**: ATNA logging helps track access to sensitive patient data, ensuring that only authorized users can access protected health information (PHI). It supports compliance with standards like HIPAA and GDPR by providing a record of user activities.
2. **Audit Trails for Accountability**: the logs capture detailed events, including user actions, system interactions, and data modifications. This audit trail enhances accountability and helps in detecting unauthorized activities or potential breaches.
3. **Regulatory Compliance**: many regulations in the healthcare industry require systems to log and monitor access to medical data. ATNA logging ensures your system complies with standards such as DICOM PS 3.15, ISO 27789, and ISO 27001 for information security management.
4. **Incident Response and Forensics**: in the event of a security incident, detailed logs are essential for investigating what happened, identifying vulnerabilities, and preventing future occurrences. ATNA provides a comprehensive log of interactions for post-incident analysis.
5. **Interoperability with Healthcare Systems**: ATNA logging is part of the IHE (Integrating the Healthcare Enterprise) framework, which promotes interoperability between medical devices and systems. Proper logging ensures your system is compatible with other ATNA-compliant systems in a healthcare environment.

By setting up ATNA logs, you safeguard your data, enhance security, and demonstrate a proactive approach to risk management and compliance with healthcare standards. This small step plays a significant role in protecting sensitive patient information and maintaining trust in your system.

The relevant configuration parameters are:

1. ATNA log collector server host name

   ```
   com.softneta.audit.serverHostname = 192.168.1.123
   ```

   Empty by default, which disables the ATNA logging. ...serverPort is another mandatory parameter.

2. ATNA log collector server port

   ```
   com.softneta.audit.serverPort = 6514
   ```

   Empty by default, which disables the ATNA logging. ...serverHostname is another mandatory parameter.

3. Use TLS transfers (mandated by ATNA profile) or not

   ```
   com.softneta.audit.useTls = true
   ```

   False by default (suitable for testing only).

4. Trust store location for TLS.

   ```
   com.softneta.audit.trustStore = file:///c:/tmp/nxlog/client/cacerts.jks
   ```

   If empty (default), will result in errors when ...useTls=true.

5. Trust store password for TLS.

```
com.softneta.audit.trustStorePassword = {password}
```

   If empty (default), will result in errors when . . . useTls=true.

6. Key store location for TLS with mutual authentication.

```
com.softneta.audit.keyStore = file:///c:/tmp/nxlog/client/cacerts.jks
```

   If empty (default), will likely result in errors when . . . useTls=true.

   You should always set up the key store, too. The ATNA profile mandates mutual authentication. One-way TLS authentication (common in World Wide Web clients that only need to prove identity of the server) is not officially supported by MedDream and there is no guarantee that it will work, even for testing purposes.

7. Key store password for TLS with mutual authentication.

```
com.softneta.audit.keyStorePassword = {password}
```

   If empty (default), will likely result in errors when . . . useTls=true.

   You should always set up the key store, too. The ATNA profile mandates mutual authentication. One-way TLS authentication (common in World Wide Web clients that only need to prove identity of the server) is not officially supported by MedDream and there is no guarantee that it will work, even for testing purposes.

8. Require user identity in ATNA messages

```
com.softneta.audit.requireUserIdentity = true
```

   Historically ATNA messages from MedDream have a fake user identity (”login@domain” etc). Some scenarios, mainly those based on token integration, carry a true identity.

   Set this parameter to true in order to crash (a stack trace will be visible in the application log) when attempting to send a fake identity via ATNA. On the one hand, messages like these won't reach the ARR at all; on the other hand, this will help to identify scenarios that still need an update, provided that you are monitoring the logs.

   8.9.0 has fixed a bug related to Login window scenarios, therefore the value “true” should now be safer to use.

Although the *previous chapter* offers more detailed configuration of generic SSL configuration, you are welcome to read it, too, for additional insights in the ATNA context.

5.1.12. The external links plugin

This plugin adds custom text-only buttons to MedDream toolbar. They execute various URLs which can contain metadata of an active image (like Study Instance UID) and are therefore suitable for additional integration with third-party systems.

The custom buttons appear below the toolbar button “Plugins” that is hidden by default. Go to Settings and search for “Plugins” in Viewer / Toolbar Settings, then choose at least “Show in desktop mode”.

To configure the plugin, modify the file medcadPlugins/links.json. By default, it contains a single entry that implements a link to English version of the MedDream user manual:

```
[
  {
    "name": "Example: A link to User Manual",
    "openType": "ACTIVE_VIEWPORT_IFRAME",
```

(continues on next page)

(continued from previous page)

```
    "url": "../languages/en/help"
  }
]
```

The top-level array can contain multiple entries instead of this single one. Elements of an entry:

Element	Allowed values	Explanation
name	Any string	Text on the button
openType	NEW_WINDOW, SAME_WINDOW, ACTIVE_VIEWPORT_IFRAME, MODAL_IFRAME, HIDDEN_IFRAME, CONFIGURED_VIEWPORT_IFR	See the next table
url	Any URL, including relative paths	Can also include `{INSTANCE}` and other placeholders from the list below
searchParams	INSTANCE, SERIES, STUDY, ACCESSION_NO, PATIENT_ID, EXAM_DATE, STORAGE	(optional) An array consisting of `"urlParamName":"INSTANCE"` etc, will be automatically added to `url` in form of `?urlParamName=dynamicValue&...`
windowWidth	Any number	(optional) Window width in pixels. To be used with MODAL_IFRAME.
windowHeight	Any number	(optional) Window height in pixels. To be used with MODAL_IFRAME.
layout	Layout parameters for CONFIGURED_VIEWPORT_IFR	See the next 3 rows
layout.columns	Number, 1 . . . 8	Viewport columns. If present, then .layout.rows is also mandatory.
layout.rows	Number, 1 . . . 3	Viewport rows. If present, then .layout.columns is also mandatory.
layout.openIn	Number, 0 . . . 24	Which viewport, from a matrix of .columns-by-.rows, is to be used. Zero indicates the first unused viewport.
close	Parameters for additional steps before the plugin is closed	Used with ACTIVE_VIEWPORT_IFRAME, MODAL_IFRAME, CONFIGURED_VIEWPORT_IFRAME. See the next 3 rows.
close.confirm	true, false	Enables a confirmation dialog where the user agrees to close the plugin. Not compatible with .close.url.
close.url	Any URL, including relative paths on the same domain	Endpoint of a third-party system that is informed about plugin closure. Response with a status code 200 or 204 approves the closure. Not compatible with .close.confirm.
close.urlResult	Any string	If not empty, then responses from .url with status code 200 must contain this text or else the response is treated as “do not close”

Values of openType control where the URL is opened:

NEW_WINDOW	Opens in a new window
SAME_WINDOW	Replaces the existing window (and entire MedDream)
ACTIVE_VIEWPORT_IFRAME	Replaces the active viewport
MODAL_IFRAME	Uses a modal IFRAME
HIDDEN_IFRAME	Uses a hidden IFRAME (not visible to end user; for triggering some event on a third-party system)
CONFIGURED_VIEWPORT_IF	Uses an IFRAME divided into viewports (up to 3 x 8)

The URL is constructed from `url` and `searchParams` by the frontend. The following placeholders that refer to metadata from the active image can be used:

- STORAGE – identifies the current PACS plugin (its `.id` in application.properties)

- INSTANCE – (0008,0018) SOP Instance UID
- SERIES – (0020,000E) Series Instance UID
- STUDY – (0020,000D) Study Instance UID
- ACCESSION_NO – (0008,0050) Accession Number
- PATIENT_ID – (0010,0020) Patient ID
- EXAM_DATE – (0008,0020) Study Date

Button examples

- Opens http://demo.example.com in a new window:

```
{
  "name": "External website demo",
  "openType": "NEW_WINDOW",
  "url": "http://demo.example.com/"
}
```

- Opens a specific study in the same window (for example, http://demo.example.com/?study=1.2.826.0.1.3680043.2.4852.20180403.113343677.435858665&storage=PacsOne&newParam=otherValue):

```
{
  "name": "Metadata in URL query",
  "openType": "SAME_WINDOW",
  "url": "http://demo.example.com/",
  "searchParams": {
    "study": "STUDY",
    "storage": "STORAGE",
    "newParam": "otherValue"
  }
}
```

- Opens a study in the active viewport, the UID is specified in the path and other parameters – in the query (for example, http://demo.example.com/PacsOne/1.2.392.200036.9125.2.37100184241200.64618798380.28802?image=1.2.826.0.1.3680043.6.6054.19688.20100603165016.704.123):

```
{
  "name": "Metadata in URL query and path",
  "openType": "ACTIVE_VIEWPORT_IFRAME",
  "url": "http://demo.example.com/{STORAGE}/{STUDY}",
  "searchParams": {
    "image": "IMAGE"
  }
}
```

- Opens a webpage in the last viewport of a 8x3 layout:

```
{
  "name": "Layout 8x3 and open page in last viewport",
  "openType": "CONFIGURED_VIEWPORT_IFRAME",
  "url": "http://meddream.com/",
  "layout": {
    "columns": 8,
    "rows": 3,
    "openIn": 24
  }
}
```

- Opens a webpage in a modal IFRAME, calls a custom URL on close, expects a certain text in the response:

```
{
  "name": "Open a modal iframe, on close call url",
  "openType": "MODAL_IFRAME",
  "url": "https://meddream.com/",
  "close": {
    "url": "https://meddream.com/products/meddream-dicom-viewer/",
    "urlResult": "test_word"
  }
}
```

- Executes a custom URL handler on the client computer:

```
{
  "name": "Run a 3rd party application on the client computer",
  "openType": "SAME_WINDOW",
  "url": "applauncher:{STUDY}"
}
```

  Custom URL handlers is a Windows feature that starts a registered application with the entire URL on its command line. This example uses a generic protocol name "applauncher". There also are success stories for Linux (using `gconftool-2`) and OSX (using a specifically crafted Application Bundle).

  MedDream includes a couple of helper files for Windows:

  – `runAppFromBrowser/runAppFromBrowser.bat` – a wrapper for 3rd-party applications. It attempts to remove the "applauncher:" prefix from the command line and passes the remainder to the application in form of `studyuid=VALUE`. You will need to adjust the path to your application, and its command line;

  – `runAppFromBrowser/runAppFromBrowserReg.bat` – registers the former file as a handler of the `applauncher` protocol. Must be run in an elevated Command Prompt window. You will need to adjust the path to runAppFromBrowser.bat and probably invent a better protocol name instead of this "applauncher".

5.1.12.1. Creating a Microsoft Teams meeting

The following example opens the "New Meeting" window of Microsoft Teams, with Patient ID and Study Date values already filled in:

```
{
  "name": "Create Teams Meeting",
  "openType": "NEW_WINDOW",
  "url": "https://teams.microsoft.com/l/meeting/new?subject=Meeting%20regarding%20patient
↪%20id:%20{PATIENT_ID}&content=Study%20date:%20{EXAM_DATE}"
}
```

5.1.12.2. Parameter override in token-based HIS integration

It is possible to override most parameters, and also add some more parameters, when MedDream is opened via token. The latter uses the pluginConfigurations element for this:

```
{
    "items": [ ],
    "pluginConfigurations": [{
      "pluginName": "Metadata in URL query",
      "parameters": [{
              "name": "url",
              "value": "http://example.com/viewer/"
        },{
              "name": "windowWidth",
              "value": "300"
        },{
              "name": "windowHeight",
              "value": "100"
        },{
              "name": "openType",
              "value": "CONFIGURED_VIEWPORT_IFRAME"
        },{
              "name": "searchParams.storage",
              "value": "STORAGE"
        },{
              "name": "layout.columns",
              "value": "2"
        },{
              "name": "autoStartOnStudy",
              "value": true
      }]
    }]
}
```

`pluginName` here must match `name` of an entry in links.json.

Like in the configuration file, searchParams supports any URL parameter name passed in form `searchParams.<parameter name>`, and the same set of metadata placeholders for dynamic values.

Parameters unique to HIS integration sessions:

autoStartOnInstance	Plugin is started every time an image is opened
autoStartOnStudy	Plugin is started every time a study is opened
autoStartOnInstanceOne	Plugin starts once when any image from the study is opened
autoStartOnStudyOneTi	Plugin starts once for any study in the token. It starts again if the viewer window is closed and opened anew.
closeIframeAfterSec	The IFRAME closes after this many seconds (MODAL_IFRAME and HIDDEN_IFRAME only)

However, a plugin configured as ACTIVE_VIEWPORT_IFRAME doesn't start in case autoStartOnStudy=true or autoStartOnStudyOneTime=true. At that moment no viewport is usually available so far.

5.1.12.3. Use case: additional image filtering

MedDream 8.8.1+ contains an experimental image filtering scenario that can be called via the Links plugin.

1. Update md5/js/filter/consts.js with the same MedDream URL that you're opening in the browser:

```
const MD_URL = 'http://MY_SERVER:MY_PORT';
```

2. Add the following entry to medcadPlugins/links.json:

```
{
  "name": "Image filters",
  "position": "VIEWER",
  "openType": "MODAL_IFRAME",
  "url": "http://MY_SERVER:MY_PORT/filters.html",
  "windowWidth": 600,
  "windowHeight": 800
}
```

   The "url" element must contain the same MedDream URL that you're opening in the browser, plus "/filters.html".

   The tested values for "openType" are MODAL_IFRAME, NEW_WINDOW and ACTIVE_VIEWPORT_IFRAME.

   "windowWidth" and "windowHeight" are important for MODAL_IFRAME.

3. Add to application.properties (or update an existing entry there) and restart Core:

```
security.postMessageWhitelist = *
```

Afterwards you can use this newly configured button, Plugins > Image filters (according to the "name" field).

Note: ACTIVE_VIEWPORT_IFRAME currently requires some counter-intuitive actions that might seem redundant:

- in order to activate the Plugins toolbar button, you need to select some non-empty viewport with a displayed image, then choose Plugins > Image filters. The viewport will then be replaced with a plugin window which is initially empty;
- in order to tell the plugin which image is to be processed, open that image in any other viewport (for example, side-by-side), or select any other non-empty viewport;
- press the "Re-load" button in the plugin window. This button always loads the image from an active viewport.

5.1.13. Reference for remaining configuration parameters

The parameters listed below can be left alone in most situations. The rarest of them aren't included in application.SAMPLE.properties for the sake of its simplicity.

1. MedDream root directory

```
com.softneta.meddream.configRoot = PATH_TO_MEDDREAM_DIRECTORY
```

   All other directories are based on this value by default, therefore a correct specification means less parameters are needed in application.properties.

   If left unconfigured, the current directory of the JVM process is substituted automatically. The substitution is based on the "user.dir" Java property, which in turn must be initialized by the JVM and not via the -D option.

If the outcome is still incorrect, you can try to specify configRoot explicitly. It is expected in the URI format, where the "file://" prefix is mandatory and relative paths are not supported. An absolute Windows path with forward slashes still works.

Examples:

`com.softneta.meddream.configRoot = file:///opt/meddream`

`com.softneta.meddream.configRoot = C:/MedDreamPACS/MedDream`

`com.softneta.meddream.configRoot = file:///C:/MedDreamPACS/MedDream`

`com.softneta.meddream.configRoot = file:///C:/PROGRA~1/PacsOne/php/meddream`

Warning: The third and fourth examples, where both the "file:///" prefix and an absolute Windows path "C:/..." are present, are not compatible with the default H2 databases. The workaround is to define paths to H2 files explicitly, according to 5.10 File-based H2 storage and non-default configRoot under Windows. The second example doesn't have this issue.

2. Path to MedDream temporary files

```
com.softneta.meddream.tempDir = PATH_TO_MEDDREAM_TEMP_FILES
```

Allows you to specify the path to a location where MedDream temporary files are stored; the default "temp" below configRoot might not always be suitable.

Examples:

`com.softneta.meddream.tempDir = ./temp` (non-URI format might not work in all cases)

`com.softneta.meddream.tempDir = file:///mnt/mdtmp`

`com.softneta.meddream.tempDir = file:///E:/mdtmp`

Note: Since 7.0 some cached files might need paths longer than 260 characters which are not allowed under Windows. The only solution is to have com.softneta.meddream.tempDir as short as possible.

3. Session duration in seconds

```
server.servlet.session.timeout = 60
```

Some PACS plugins are slower and might take several minutes to load even the structure of a large study (afterwards error icons are displayed instead of thumbnails). In these situations, the timeout should be increased.

Please remember that due to larger timeouts, client connections will persist longer in case of inactivity or incorrectly closed MedDream window (using the "X" icon on the corner instead of "Logoff"). With a license that limits the number of concurrent connections, this might exhaust the free connections more often.

4. Session cookie name

```
server.servlet.session.cookie.name = MEDDREAMSESSID
```

Value in the example is a hidden default and won't likely need correction.

For cases where the current value is used by HIS or different software on the same domain; in these situations, MedDream unpredictably loses its session.

5. Browser's security policy

This is the default and allows embedding MedDream only on the server that hosts it:

```
security.frameOptionsPolicy = SAMEORIGIN
#security.contentSecurityPolicy = frame-ancestors 'self'; default-src 'self'; base-
↪uri 'self'; object-src 'none'; style-src 'self' 'unsafe-inline'; script-src 'self'
↪'unsafe-eval'; img-src 'self' data: blob:; connect-src 'self'; frame-src 'self'␣
↪blob:; worker-src 'self' blob:; form-action 'self';
```

(Results in headers `X-Frame-Options: none`, `Content-Security-Policy: frame-ancestors 'self'; default-src 'self'; base-uri 'self'; object-src 'none'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-eval'; img-src 'self' data: blob:; connect-src 'self'; frame-src 'self' blob:; worker-src 'self' blob:; form-action 'self';`.

This disables both headers altogether, therefore MedDream can be opened in an IFRAME without any restrictions:

```
security.frameOptionsPolicy = NONE
#security.contentSecurityPolicy =
```

This disallows IFRAME embedding completely:

```
security.frameOptionsPolicy = DENY
#security.contentSecurityPolicy = frame-ancestors 'none'; default-src 'self'; base-
↪uri 'self'; object-src 'none'; style-src 'self' 'unsafe-inline'; script-src 'self'
↪'unsafe-eval'; img-src 'self' data: blob:; connect-src 'self'; frame-src 'self';␣
↪worker-src 'self' blob:; form-action 'self';
```

(Results in headers `X-Frame-Options: deny`, `Content-Security-Policy: frame-ancestors 'none'; default-src 'self'; base-uri 'self'; object-src 'none'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-eval'; img-src 'self' data: blob:; connect-src 'self'; frame-src 'self'; worker-src 'self' blob:; form-action 'self';`. Please note that **DICOM PDFs won't open any more**, as MedDream uses an internal IFRAME for the system-default PDF viewer.)

This allows embedding on servers listed in the second parameter:

```
security.frameOptionsPolicy = NONE
security.contentSecurityPolicy = frame-ancestors 'self' \
    93.184.216.34 example.com
```

(Results in a header `Content-Security-Policy: frame-ancestors 'self' 93.184.216.34 example.com`. We strongly encourage to use the entire long list of directives, with default-src, script-src and so on.)

In the examples above, .contentSecurityPolicy is commented out as the same default value comes from .frameOptionsPolicy. Exotic cases might require overrides. By the way, if .contentSecurityPolicy is explicitly specified and contains `frame-ancestors 'self'`, then .frameOptionsPolicy must be `NONE`.

Most importantly,

- if workersConfig.maxJphWorkers > 0 in system.json, then the script-src directive must include `'unsafe-eval'`. When not using the client-side decompression (look for "prepareFromPacsMetadata" in this Manual), this unsafe-eval is needlessly permissive and should not be used;
- when not using any kind of IFRAME embedding (and not viewing PDFs), frame-src is safer without `blob:`;
- the Login window needs `'unsafe-inline'` in the script-src detective, in order to detect too old browser versions, or else there will be an error in the developer's console;

- worker-src is also safer without `blob:`, although this hasn't been thoroughly tested;
- the Links plugin is obviously affected by frame-src if openType in plugin configuration involves an IFRAME and `url` contains a different domain. However, once the content downloaded via `url` takes control, its success depends on a CSP directive sent **by that server**. In theory it might be possible to host something under MedDream's `md5` subdirectory (subject to heavy limitations) and only then security.contentSecurityPolicy would be relevant.

Note: 8.9.0+ has tightened some default directives that are in effect when security.contentSecurityPolicy is not configured:

8.8.1	8.9.0
	base-uri 'self'
	object-src 'none'
script-src 'self' 'unsafe-eval' 'unsafe-inline'	script-src 'self' 'unsafe-eval'
img-src 'self' data:	img-src 'self' data: blob:
child-src 'self' blob:	
connect-src 'self' *	connect-src 'self'
	frame-src 'self' blob:
	form-action 'self';

If you didn't have it configured and something isn't working any more after the update, we first suggest configuring one of detailed security.contentSecurityPolicy examples above, then looking at browser's console for the actual error message, which might provide a hint what directive should be adjusted.

As a last resort, you can try the full previous value. The following corresponds to the SAMEORIGIN case above:

```
security.contentSecurityPolicy = frame-ancestors 'self'; default-src 'self';␣
↪base-uri 'self'; object-src 'none'; style-src 'self' 'unsafe-inline'; script-
↪src 'self' 'unsafe-eval'; img-src 'self' data: blob:; connect-src 'self';␣
↪frame-src 'self' blob:; worker-src 'self' blob:; form-action 'self';
```

6. Which causes of a login failure are revealed to the end user

```
security.allowedAuthErrors = noStudyFound,wrongUsernamePassword,\
unsupportedConfiguration,connectionLimitReached,\
reachedMaximumAttemptsMin,reachedMaximumAttemptsSec
```

The messages in the Login window (and in the corresponding part of the GUI when using HIS integration) by default include "No study found", "Wrong username or password", "Unsupported configuration", "Connection limit reached", "Reached maximum login attempts". This set corresponds to the example above.

If this setting is present and doesn't contain one or more of those predefined entries, then the corresponding message is replaced with a more obscure "Access denied". Note that as of 8.7.0 it might not be possible to retrieve the original cause from the application log, therefore this choice hinders both abuse and troubleshooting.

7. "Secure" flag of the session cookie

```
server.servlet.session.cookie.secure = true
```

After configuring `true`, MedDream must be accessed over HTTPS or else the browser won't send session cookies back to the server after a successful login. Recommended in HTTPS-only installations.

8. "SameSite" flag of the session cookie

```
server.servlet.session.cookie.sameSite = strict
```

The default value is the string "unset", which means the flag is not added to the cookie. Other possible values:

- `none` – no restrictions on the cookie, however the "Secure" flag is still mandatory. **Not supported by older browsers**;
- `lax` and `strict` will further prevent embedding MedDream in an IFRAME, and might cause problems with viewing DICOM PDFs.

Note: Since February 2020, Chrome changed the handling of cookies without this flag. Other browsers will follow, too.

Now their default is "lax"; to get the older equivalent of "none", you must specifically use `sameSite = none` in MedDream settings.

With "lax", MedDream should be hosted on a subdomain – for example, https://hospital.com will be able to display an IFRAME containing https://viewer.hospital.com.

In application.SAMPLE.properties the "lax" version is commented out by default. You can uncomment it if all your clients have compatible up-to-date browsers, as even support for "lax" is not widely present.

---

9. Disable the X-XSS-Protection header if needed

```
security.xssProtectionDisabled = false
```

The header X-XSS-Protection is sent by default; `true` here will turn it off for troubleshooting etc.

10. Disable the X-Content-Type-Options header if needed

```
security.contentTypeOptionsDisable = false
```

The header X-Content-Type-Options is sent by default; `true` here will turn it off for troubleshooting etc.

11. Address of the reverse proxy through which requests from Internet reach MedDream

```
security.meddreamProxyIp = 192.168.111.100
```

Needed to correctly extract remote IP addresses from the X-Forwarded-For header. This way MedDream logs will contain true remote IPs instead of the proxy address (unless the proxy does not provide X-Forwarded-For). Typically used in installations with both remote clients (they reach MedDream via a proxy server) and local clients (direct connections from the LAN).

Empty by default. **Must not be present if there is no proxy**, or else MedDream will believe spoofed values of X-Forwarded-For.

Since 8.3.1 hostnames beside IPs are supported, too.

12. Detect the change of remote IP address

```
security.detectRemoteIpChange = WARN
```

`LOGOFF` invalidates the session, `WARN` just logs a warning, `OFF` disables the feature (default).

Might also require the following settings:

```
server.tomcat.remote-ip-header = x-forwarded-for
server.tomcat.protocol-header = x-forwarded-proto
server.forward-headers-strategy = native
server.use-forward-headers = true
```

13. Restrict client-side addresses that can open MedDream via HIS integration

```
security.authAllowedOrigins = 192.168.111.20,192.168.111.24
```

If not empty, IPs not in the list will be denied access to the /auth endpoint, which is a crucial part of HIS integration.

14. Restrict client-side addresses that can use the Login window

```
security.meddreamLoginAllowHostIp = 192.168.111.0/24,dep.infra.acme.com
```

If not empty, IPs not in the list will get the same "403" page instead of the Login window, as in case of meddream.loginEnabled=false.

CIDR-style subnets and hostnames are supported, too.

15. Allow non-compliant UIDs

```
security.uidValidationPattern = [0-9\\.]+
```

The regular expression in the example is the default and allows standard DICOM UIDs (dot-delimited numbers). Non-matching UIDs will result in an error "UID is of invalid format".

As MedDream uses the Java's Matcher.match() method, the pattern already applies to the entire string and start/end anchors are not needed.

Example 1. Your installation already contains a few files with UIDs like "d2fd4606790bd..." (some hashed data). These can be accepted by `[0-9a-f\\.]+`.

Example 2. You're using MS Azure DICOM Service that allows UIDs of alphanumeric, minus and dot characters, and a few files have corresponding UIDs. This could be covered by `[0-9A-Za-z\\.-]+`.

Warning: It is too easy to write a regular expression that accepts too much or anything (we won't give ideas to you) instead of protecting from malicious data. You should know what you're doing.

Some kinds of regular expressions are themselves vulnerable to denial of service attacks. Our default is safe but your version might not, especially if the hacker somehow knows what you've configured (although the regexp is not quoted in the error message, it's only logged). Generally, it is safer to just use a simple character range instead of trying to enforce some combinations, fragments and repetition count.

Note: The characters `:` and `!` are reserved by MedDream for suffix separators in some scenarios and UIDs containing them won't be accepted despite this setting.

Basically, the validation algorithm expects at most a single separator, instead of just using the last fragment after this separator.

16. Limit the number of failed login attempts

```
com.softneta.meddream.login.maxAttemptsToConnect = 5
com.softneta.meddream.login.timeoutSeconds = 60
```

(continues on next page)

(continued from previous page)

```
com.softneta.meddream.login.doubleTimeOnRepeatedFailedAttempts = true
com.softneta.meddream.login.failedAttemptsMaxWaitSeconds = 900
```

After maxAttemptsToConnect or more failures from the same IP address, any further attempts will immediately fail (without validating the provided credentials) during next timeoutSeconds seconds **and** double the remaining wait time (but not above failedAttemptsMaxWaitSeconds). The usual validation of credentials resumes after this wait time ends. The attempts counter is reset by a successful login.

doubleTimeOnRepeatedFailedAttempts = false corresponds to pre-8.5 behavior.

Default values are shown in the example. The limiting is disabled if maxAttemptsToConnect or timeoutSeconds is zero.

While the wait time is enforced, a malicious network client that attempts to log in repeatedly can result in a denial of service for everyone behind the same IP address. Decreasing failedAttemptsMaxWaitSeconds, and even disabling doubleTimeOnRepeatedFailedAttempts, will be of little help. The setting `security.meddreamProxyIp` is important for correct IP addresses; the administrators must ensure that local proxies preserve them in corresponding headers, in order to minimize the impact.

17. Disable the message about an unsuitable password

```
com.softneta.meddream.login.showWarningMessagesAboutPassword = false
```

By default, 8.7.0+ checks passwords against a hardcoded list of publicly known values and displays a warning on a match. This check can be disabled, which makes sense for demo installations with a few anonymized studies and a deliberately simple password.

18. Path to writeable settings files (system.json / global.json / shortcuts.json / hangingProtocols*.json / columns.json / dicomprint.json)

```
com.softneta.settings.fileLocation = ${com.softneta.meddream.configRoot}/\
    sys/settings
```

The value in the example is the default.

Since 7.8.0, to facilitate installations where most of directory tree is read-only, this parameter applies only to:

- system.json / columns.json (created automatically if missing),
- global.json / shortcuts.json / dicomprint.json / hangingProtocols.json / hangingProtocolsV2.json (created interactively by the Settings window etc),
- hangingProtocolsConfig.json (might need manual editing together with hangingProtocolsV2.json),
- ecgAnnotations.json (created automatically if missing).

Location of rebranding configuration (including `rebranding_configuration/favicon.ico`) and `robots.txt` is now hardcoded as “sys/settings” below configRoot. Since 8.3.1, in a similar fashion the translation files are in “sys/locales” (not “sys/settings/locales”) below configRoot.

Therefore, in a Docker image one can specify a dedicated directory for fileLocation, bindmount it and keep all modifiable files outside the container.

19. Store settings JSON files on AWS S3

```
com.softneta.settings.awsBucketName = meddream-settings-per-region
com.softneta.settings.awsRegionName = us-east1
```

The intended use covers the files sys/settings/*.json (subject to com.softneta.settings .fileLocation):

- system.json
- global.json (also USER_global.json and GROUP_g_global.json)
- hangingProtocols.json (also USER_hangingProtocols.json and GROUP_g_hangingProtocols.json)
- hangingProtocolsV2.json (also USER_hangingProtocolsV2.json and GROUP_g_hangingProtocolsV2.json)
- hangingProtocolsConfig.json
- shortcuts.json (also USER_shortcuts.json and GROUP_g_gshortcuts.json)
- dicomprint.json (also USER_dicomprint.json and GROUP_g_dicomprint.json)
- ecgAnnotations.json
- columns.json

Notably, **sys/rebranding/rebranding_configuration.json and related files are not included** as of MedDream 8.8+.

The new behavior is enabled by a non-empty awsBucketName and needs non-empty awsRegionName for proper function. awsRegionName alone is ignored.

In this scenario MedDream authenticates at S3 via the DefaultCredentialsProvider of S3 SDK. Particularly, environment variables AWS_SECRET_ACCESS_KEY / AWS_ACCESS_KEY_ID and instance profile credentials (ECS task permissions like AmazonS3FullAccess) have been tested.

Note: As of 8.8.0+, when these two settings are active, the application log doesn't indicate that. It still shows an absolute path like "/opt/meddream/sys/settings/global.json"; however, the path is discarded when accessing the S3 bucket, and the bucket will contain just "/global.json".

20. Disable usage statistics collection

```
com.softneta.statistics.enabled = true
```

By default, MedDream logs number of sessions and similar things to files statistics/YYYY-MM.stat below configRoot.

When using the "pay per view" licensing model, these offline data provide proof for a certain amount of monthly payment. In other cases, statistics can help to find the optimal number of concurrent connections or provide insights when troubleshooting performance problems.

21. Path to usage statistics files

```
com.softneta.statistics.fileLocation = ${com.softneta.meddream.configRoot}/\
    statistics
```

The default value is shown above. In dockerized installations this option might be more convenient than symlinks etc.

22. Local AE Title for saving annotations and screenshots

```
com.softneta.meddream.dcmsnd.bind = MEDDREAM
```

Created DICOM objects will be sent to the PACS over DICOM C-STORE. This Local AET identifies MedDream as a sender. For testing purposes, one might use any AET already accepted by the PACS.

Not used in case of the "QR" plugin with storageApiEnabled=true because the plugin has a separate parameter for Local AET.

23. Wait for annotation and screenshot object availability at the PACS

```
com.softneta.meddream.secToWaitAfterCreatedDICOMSend = 10
```

If this setting is nonzero, then a fresh PR/KO/RTSTRUCT/SC object must appear in study structure not later than after this number of seconds. Default is zero, it disables checking the presence of this object in the structure and assumes that the PACS is able to import an object received via C-STORE quick enough.

You might also see an error message that suggests to increase this parameter, in the case when annotations are sent to a wrong receiver. For example, the storeScp* parameters refer to a different Storage SCP which receives the new object without any errors; meanwhile the original PACS that provides the study structure still doesn't know about this object. A scenario worth mentioning is the Save Key Objects function with the "Include annotations" checkbox turned on.

The third reason behind this message could be the series restriction in the token: some annotations from MedDream do not reuse an existing series, while those that do, will still create a new series during the first time. This has been fixed in 8.8.0+ due to default com.softneta.meddream.blacklist.tokenSeriesRestrictionPreservesAnnotations = true; however, if you need (and still use) `false` instead, then remember that a series restriction is in essence incompatible with new annotations.

24. Source of certain attributes in annotation and screenshot objects

```
com.softneta.meddream.newObjectsUseMetadataFromPacs = false
```

The default value is `true` and Patient ID is taken from the PACS by a dedicated request for study metadata (normally the result isn't cached and the request is executed during every save operation). This might be beneficial if there is a chance that the PACS will change the Patient ID without updating the DICOM files.

When changed to `false`, Patient ID is taken from attributes of the image being annotated (these are normally cached). This is important for integrations where the Patient ID isn't present in study metadata at all.

25. Patient History scope for MultiPACS configurations

```
com.softneta.meddream.searchPatientHistoryInAllStorages = true
```

The default value is `false`: the PACS from where the study was opened, also provides the patient history for this study. In contrast, `true` forces searching for related studies in all PACSes available to the current user.

26. Maximum depth of Patient History, in days

```
com.softneta.meddream.patientHistoryDaysLimit = 365
```

The default is zero (no limit).

27. Place a measurement on the current frame or on all frames

```
com.softneta.meddream.separateFrameMeasurements = false
```

8.8.0+ places a measurement on a single frame of a multiframe image by default. Change to `false` to place them on all frames (legacy behavior).

28. Credentials for management endpoints like /manage, /h2-console, etc.

```
authentication.manager.username = LOGIN_NAME
authentication.manager.password = LOGIN_PASSWORD
```

Use a strong password, and do not leave it configured longer than necessary. (An empty username or password makes the login impossible.)

Note: Both values can also be **file names, or paths to files**, from where the final content will be loaded. A successful load operation is not indicated in the logs. File contents are used as is, without removing empty lines, whitespace or newline characters (please make sure that there is no line ending in the file).

Due to this "hidden" functionality, a quite rare bug is possible: when you have a too simple password that matches the name of an existing file, then content of that file is used instead, without any warning. Strong passwords will prevent that. With usernames the chance is obviously higher.

29. Enable functionality of the /manage endpoint

```
management.endpoints.enabled-by-default = true
management.endpoints.web.exposure.include = *
```

This is diagnostic functionality for qualified personnel. The endpoint can be used to examine MedDream logs etc. over HTTP, therefore it presents a security risk.

Do not leave it configured longer than necessary, and use a strong password in authentication.manager.password.

By default, .enabled-by-default is `false` and yields an empty directory anonymously; .include is just `health`.

It is also possible to trim down the amount of exposed data as it might contain information sought by hackers. You should enable only individual subsets relevant to your case. The example below exposes only /manage/env and /manage/health:

```
management.endpoint.env.enabled = true
management.endpoint.health.enabled = true
management.endpoints.web.exposure.include = env,health
```

By default, /manage is available on the same port and address(es) like other parts of MedDream. It is strongly recommended to at least use a different port, or even bind to 127.0.0.1 to prevent access from other machines:

```
management.server.port = 8081
management.server.address = 127.0.0.1
```

Note: Spring Framework requires to have management.server.port different from server.port when management.server.address is used.

30. Enable the /h2-console endpoint

```
spring.h2.console.enabled = true
spring.h2.console.settings.web-allow-others = true
```

The /h2-console endpoint allows someone to manipulate contents of internal databases (Reporting function, auth-meddream credentials storage, job engine and so on). This is an obvious security risk.

Other places of this Manual explain the rare cases when this endpoint is needed. Generally, do not leave it configured longer than necessary, and use a strong password in authentication.manager.password.

31. Log level

The log level can be adjusted for every class or a particular part of class hierarchy. The example below provides separate parameters (not alternatives of a single parameter).

```
logging.level.ROOT = ERROR
logging.level.com.softneta = INFO
logging.level.com.softneta.meddream.dicomStoreServer = DEBUG
```

32. Log file name without extension

```
logging.file.name = meddream
#logging.file.name = /opt/md/log/meddream
```

Commenting this line out will disable logging to file. The console, if visible, will still receive the messages.

Do not include the extension; it is added automatically together with additional suffixes.

A path can be included, too. All directories must exist, the last one must be writeable by the user under which the JRE process is running. A related Spring Boot setting, logging.file.path, only allows you to change the path while keeping default file names (logging doesn't work if you add both settings).

33. Limit size of old log files

```
logging.logback.rollingpolicy.max-file-size = 10MB
```

The log file is automatically rotated (renamed to *.log.1, *.log.2, ...) when its size reaches this value.

Before 8.1.0, `logging.file.max-size` was suggested instead, however it didn't work.

34. Limit number of old log files

```
logging.logback.rollingpolicy.max-history = 10
```

During log rotation, the oldest ones are removed unless this parameter is set to 0.

Before 8.1.0, `logging.file.max-history` was suggested instead, however it didn't work.

35. Base directory for caching thumbnails, pixel data, DICOM attributes and study structure objects

```
com.softneta.preparation.cacheDir = ./temp/cache
```

By default, it's the subdirectory "cache" under tempDir.

36. Directory where converted videos and conversion process logs are stored

```
com.softneta.video.convertedDir = ./temp/converted-videos
```

37. MPEG-4 AVC profiles assumed to be playable on any client device

```
com.softneta.video.forceAcceptMp4Profiles = 100, 88, 77, 66
```

If the DICOM video file contains an MPEG-4 AVC stream and its "profile" codec parameter is found in this list, then the video is assumed already compatible and won't be converted. Otherwise, the frontend will use the MediaSource API to detect support for this video in the browser, and most likely the video will get converted.

The value in the example is the default and was hardcoded before 8.5.0. Now it is possible to add some higher profile (for example, 244) if all client devices support it, or remove some profile not supported by the lowest-end device. In the latter case the video is converted for all devices; in order to make it compatible with the lowest-end device, please remember to adjust com.softneta.video.commandLineTemplates.generic.

There is a certain support for files cached by older versions of MedDream. In those versions the backend was producing the "videoNeedsConversion" flag for the frontend, and it was cached among the instance metadata. If this cached flag is encountered by 8.5.0+, then:

- its meaning "video is good enough" forces immediate play of the video in the browser like in older versions (the forceAcceptMp4Profiles logic is skipped). If you have adjusted the list of accepted profiles but the behavior didn't change, then you might need to use Clear Cache for at least a particular study;
- its meaning "video needs conversion" forces immediate conversion according to com.softneta.video.commandLineTemplates.generic (the ...codec_PPCCLL logic described further is skipped). The new more flexible behavior will need the Clear Cache functionality, too.

38. Generic FFmpeg command line for video conversion

```
com.softneta.video.commandLineTemplates.generic = -i {INPUT} -y -vcodec libx264␣
↪\
-preset medium -movflags faststart {OUTPUT}
```

`{INPUT}` and `{OUTPUT}` are automatically replaced with corresponding file names. There is **no support for options with embedded spaces**, and quoting will not help; the space character always separates adjacent options.

The time-proven value for most official builds of FFmpeg is shown in the example above. It utilizes a software H.264 encoder and produces a `High@3.0` video stream playable on most devices. For lower-end devices, you can try `-profile:v main` instead of `-preset medium`, etc.

If you have a hardware encoder, it might be possible to speed up the conversion without noticeable loss of quality. For an Nvidia video adapter under Windows, one can try the command line `-hwaccel cuda -i {INPUT} -y -vcodec h264_nvenc -preset slow -movflags faststart {OUTPUT}` after installing recent drivers (the version in Windows Update might still be too old). Experiments under Linux will need a recent FFmpeg build, at least configure'd with `--enable-nvenc`. Do not forget to examine the files *.out under convertedDir if the conversion is failing.

The thumbnail creation command line is hardcoded.

The default value in 8.6+ is adapted to MedDream's own limited build of FFmpeg:

```
com.softneta.video.commandLineTemplates.generic = -i {INPUT} -y -c:v␣
↪libopenh264 \
-b:v 7M -f mp4 -movflags faststart {OUTPUT}
```

39. Codec-specific FFmpeg command lines for video conversion

```
com.softneta.video.commandLineTemplates.codec_PPCCLL = ...
com.softneta.video.commandLineTemplates.codec_64001E = -i {INPUT} -y -vcodec \
libx264 -preset medium -pix_fmt yuv420p -movflags faststart {OUTPUT}
com.softneta.video.commandLineTemplates.codec_F4001F =-i {INPUT} -y -vcodec \
libx264 -profile:v high444 -pix_fmt yuv444p -movflags faststart {OUTPUT}
```

Every name "codec_PPCCLL" must contain hexadecimal values of profile, constraints and level bytes that will be present in the resulting video. No default values (like above) are present: application.properties only allows you to overwrite their values, however some scenarios might need their complete removal – which is not possible using the standard Java property file syntax.

The frontend uses the MediaSource API to verify whether a video, identified by those PPCCLL parameter triplets, is playable in the browser. This is done both for original video in the DICOM file, and for all PPCCLL triplets collected from configuration. The original video needs conversion, too, if it has a different container (not MPEG-4) or a different codec (not AVC/H.264).

- If the detection is negative (video can't be played as is), then the highest quality promised by the configuration and still supported by the browser is used during conversion. To determine the highest quality, the PPCCLL parameter triplets are simply sorted; therefore, higher profile values (and, for the same profile, higher level values) are treated as "better".

  - In case there are no "codec_PPCCLL" options configured, or all parameter triplets are reported as unsupported, then the legacy command line (com.softneta.video.commandLineTemplates.generic) is used as a fallback; consequently, it should be adjusted for compatibility with the lowest-performing client device.

- If the detection fails (for example, MediaSource API is missing), then the conversion is performed using the first configured triplet, instead of choosing the highest quality.

Note: The build of FFmpeg bundled with MedDream 8.6+ might be too limited for the mechanism documented above. The examples are for official builds.

---

40. Directory with video conversion scripts flv.bat / flv.sh

```
com.softneta.video.shellScriptsDir = \
${com.softneta.meddream.configRoot}/sys/ffmpeg
```

The value above is the default in versions before 8.6. Now the default is empty, which is automatically replaced with "./sys/ffmpeg/windows_x86-64", "./sys/ffmpeg/linux_x86-64" or "./sys/ffmpeg/linux_arm64", depending on detected platform; a non-empty value is used as is (without platform detection).

It configures only the directory. The script name is still chosen automatically, depending on the detected platform.

The former scripts in "./sys/ffmpeg" have been moved into corresponding platform subdirectories. Changes inside them are minor, related only to the location of the FFmpeg binary.

flv.sh was calling a system-wide "ffmpeg" before 8.6, now it expects the binary in the same directory. The simplest way to use a different binary is a full path in this line:

```
#EXEDIR=$(dirname "$0")
EXEDIR=/directory/for/ffmpeg/binary
```

flv.bat in 8.6+ still expects ffmpeg.exe in the same directory but uses a different syntax for that. To use a different binary, a full path is recommended, too (do not forget the trailing backslash):

```
rem set FLV_CWD=%~dp0
set FLV_CWD=c:\ffmpeg-7.1.57\
```

Note that this different binary will likely need different `com.softneta.video.commandLineTemplates.*` and `com.softneta.thumbnails.multiframeConverter.*`.

41. Show a message if study structure loading takes too long

```
com.softneta.thumbnails.loadingMessageTime = 3
```

When the study structure is loading longer than this number of seconds, a message "STUDY LOADING..." is displayed. Zero by default, which means behavior of versions before 8.5 (no message).

42. Use a different build of FFmpeg for video thumbnails, conversion of unsupported video formats to MPEG-4, video alternative to multiframe

```
com.softneta.thumbnails.ffMpegExecutable = PATH_TO_FFMPEG
```

Starting from 8.6, an empty value (default) is automatically replaced with "./sys/ffmpeg/windows_x86-64/ffmpeg.exe", "./sys/ffmpeg/linux_x86-64/ffmpeg" or "./sys/ffmpeg/linux_arm64/ffmpeg", depending on detected platform. MedDream now includes its own special build of FFmpeg for all supported platforms.

Before 8.6, the value "ffmpeg" (default) was used as is under Linux and usually referred to a system-wide binary; under Windows it was replaced with "./sys/ffmpeg/ffmpeg.exe" where the "essentials" build from gyan.dev was located.

You can switch to a previous/different build as follows:

- under Linux, change ffmpegExecutable to just "ffmpeg" for a system-wide binary found via PATH environment variable, or to a full path;
- same under Windows, although full paths are more popular there;
- adjust the related settings `com.softneta.video.shellScriptsDir` (or at least the path to binary in the corresponding script), `com.softneta.video.commandLineTemplates.*`, `com.softneta.thumbnails.multiframeConverter.*`.

43. Options for producing a video alternative to multiframe

    (For the scenario when the alternative is used, see `com.softneta.preparation.sopClassesForConversionToVideo`.)

```
com.softneta.thumbnails.multiframeConverter.inputName = frame%d.png
com.softneta.thumbnails.multiframeConverter.commandLineTemplate = -framerate \
{FRAME_RATE} -i {INPUT} -c:v libx264 -pix_fmt yuv420p -vf \
pad=ceil(iw/2)*2:ceil(ih/2)*2 {OUTPUT}
```

    The example above corresponds to hardcoded behavior before 8.6. You can use these values when switching to a "non-free" FFmpeg build that your Linux distribution provides (or to the "essentials" build from https://www.gyan.dev/ffmpeg/builds/ under Windows).

    The defaults since 8.6 are adapted to the bundled FFmpeg build which has a different encoder and no support for PNG input:

```
com.softneta.thumbnails.multiframeConverter.inputName = frame%d.jpg
com.softneta.thumbnails.multiframeConverter.commandLineTemplate = -framerate \
{FRAME_RATE} -i {INPUT} -c:v libopenh264 -pix_fmt yuv420p -vf \
pad=ceil(iw/2)*2:ceil(ih/2)*2 {OUTPUT}
```

44. Directory to store temporary anonymized images before uploading to DICOM Library

```
com.softneta.dicom.library.upload.tempDir = ./temp/upload-to-dicom-library
```

45. Settings for Export to ISO/Burn

```
com.softneta.export.tempDir = ${com.softneta.meddream.tempDir}
com.softneta.export.tmpCleanAfterMilliSec = 86400000
com.softneta.export.fileAliveTimeInMillis = 10800000
com.softneta.export.additionalSoftwarePath=
#com.softneta.export.isoSizes[0].id = MiniCD
#com.softneta.export.isoSizes[0].name = Mini CD (210 MB)
#com.softneta.export.isoSizes[0].size = 220200960
com.softneta.export.appendIsoSizes = false
com.softneta.export.ihePdiEnabled = true
```

    Every `tmpCleanAfterMilliSec` milliseconds after application start, MedDream attempts to remove outdated (older than `fileAliveTimeInMillis`) export jobs and their temporary files under `tempDir`. During normal operation, this data is removed after the Export window closes, provided that there are no related background downloads.

    If `additionalSoftwarePath` is specified, it shall be the path to a directory with files of a DICOMDIR viewer (autorun.inf and others). They will be automatically included in each disc of exported studies. Since 7.9, Export can create a ZIP file like the Archive function does (always of unlimited size, however), and the DICOMDIR viewer is included there as well.

The `isoSizes[]` array can be used to specify custom media sizes. `id` is a simple alphanumeric identifier (Latin letters, no spaces). `size` is in bytes, zero means no limit.

When `appendIsoSizes` is true, custom media sizes are appended to the default list of unlimited (0), CD (681574400), DVD (5046586572), DL-DVD (9126805504). Otherwise the default list is lost and, if some of its entries are still needed, they need to be added to `isoSizes[]` manually.

ihePdiEnabled=true (default is false) enables the "Web content option" as per the IHE PDI profile:

- A static rendering of image files will be added in the "IHE_PDI" subdirectory on the exported volume. In other words, the output of the Export function will also include some equivalent of the Archive function with JPEG format.
- This does not work properly when splitting into volumes is requested; to avoid misunderstandings, you can disable the splitting altogether, by using appendIsoSizes=false and a single entry . . . isoSizes[0].size=0.
- The file README.TXT is generated from the template sys/settings/ihe_pdi_readme.txt. One example is bundled with MedDream. ${INSTITUTION_INFO} is auto-filled from the HIS integration token, "institution" field; ${PRODUCT_INFO} comes from the license and rebranding; in simple installations you can enter your own text instead of these placeholders.

46. Resource limits for MPR Export

```
com.softneta.meddream.mpr.maxMemoryBytes = 8589934592
com.softneta.meddream.mpr.maxConcurrentJobs = ${executor.export-task-executor.
↪core-pool-size}
```

The MPR Export functionality may require a lot of memory. In that case, multiple exports (even by the same user) become risky as the Java backend sometimes doesn't properly recover from an "out of memory" situation without a restart. These two settings can limit:

- the total amount of memory in all MPR Export jobs. A new job is not started if it would exhaust maxMemoryBytes, therefore the setting should provide enough space for at least a single uncompressed series;
- the total number of parallel MPR Export jobs. A new job is not started if maxConcurrentJobs are already running. However, values larger than executor.export-task-executor.core-pool-size will be ignored. (By the way, the latter is shared between MPR Export, Export/ISO and Export/Archive.)

Unlimited by default.

47. System settings related to DICOM Print

```
com.softneta.meddream.dicomPrint.printScuAet = $(com.softneta.meddream.dcmsnd.
↪bind)
com.softneta.meddream.dicomPrint.taskThreadCount = 5
```

Default values are shown in the example.

When communicating with a DICOM printer, you can use a different Calling AE Title than for other DICOM operations.

The maximum number of threads dedicated to printing (and, eventually, number of parallel print sessions) is configurable, too. This sets both core and max counts of a dedicated thread pool.

48. Possibility to control anonymization amount in the Share via DICOM Library function (VET license only)

```
com.softneta.dicom.library.upload.anonymizeLevel = false
```

For non-VET license this parameter is ignored and remains set to 1.

49. Possibility to disable the study count request for Patient History window

```
com.softneta.meddream.patientStudyCountEnabled = false
```

This is not needed under normal circumstances (value should remain `true`).

50. Open a custom viewer from Search window

```
com.softneta.meddream.alternativeViewerUrl = http://SERVER/viewer?study={study}
```

If not empty, the Search window will open this URL instead of the Viewer window.

Can contain placeholders "{study}" and "{storage}" that will be replaced with Study Instance UID and storage ID. If "{study}" is missing, then a hardcoded query "?studyUid=<Study Instance UID>&storageId=<storage ID>" is appended. The other one, "{storage}", is optional.

51. Don't allow adding studies of multiple patients from the Search window

```
com.softneta.meddream.onePatientViewEnabled = true
```

When a study is being opened from the Search window, the Viewer window will automatically close already open studies where Patient ID differs from the new one.

52. Bind address of the HTTP server

```
server.address = 192.168.200.100
```

This parameter can be used to expose MedDream on a single IP address that belongs to the machine. The default value is 0.0.0.0, meaning "all IP addresses".

53. HTTP context path

```
server.servlet.context-path = /meddream
```

By default, the MedDream backend is available at `/`. Some reverse proxying scenarios might benefit from different values.

54. Enable/disable compression of HTTP responses

```
server.compression.enabled = true
server.compression.mime-types = application/octet-stream
```

If the first parameter is set to true (by default it's false), then the second one lists comma-separated MIME types to which the compression will be applied.

55. Maximum size of HTTP headers

```
server.max-http-request-header-size = 24576
```

MedDream configures 24 KB by default (a big difference compared to 8 KB in Tomcat). Some integrations might need even more and fail with HTTP 400.

Value is in bytes.

56. Age of the /branding resource

```
com.softneta.meddream.http.brandingCacheMaxAgeSec = 0
```

The /branding resource will expire in proxy/browser caches after this many seconds. Zero (default) turns the caching off, that is, the browser will always perform the request anew. Before 8.0.0, the value was hardcoded as 3600.

57. Rename some settings files

```
com.softneta.settings.globalFileName = global.json
com.softneta.settings.systemFileName = system.json
com.softneta.settings.shortcutsFileName = shortcuts.json
com.softneta.settings.columnsFileName = columns.json
com.softneta.settings.hangingProtocolsFileName = hangingProtocols.json
com.softneta.settings.hangingProtocolsV2FileName = hangingProtocolsV2.json
com.softneta.settings.hangingProtocolsConfigFileName = hangingProtocolsConfig.
↪json
com.softneta.settings.dicomPrintFileName = dicomprint.json
com.softneta.settings.robotsFileName = robots.txt
```

There is no protection against using the same name in multiple settings. Default values are shown in the example.

MedDream is shipped with an example file sys/settings/robots.txt. If you are experimenting with multiple files, it might be more convenient to point to a different one instead of rewriting contents of the same file.

58. Adjust what color palettes are offered, and in what order

```
#com.softneta.meddream.enabledColorPalettes = HOT_IRON,PET,HOT_METAL_BLUE,\
#   PET_20_STEP,SPRING,SUMMER,FALL,WINTER,SPECTRUM2
com.softneta.meddream.enabledColorPalettes = HOT_IRON,PET,HOT_METAL_BLUE,\
    PET_20_STEP,SPRING,SUMMER,FALL,WINTER,RAINBOW,SPECTRUM2,SOKOLOFF,\
    SPECTRUM,RAINBOW2
```

Affects both Fusion functionality and the Color Palette Selection widget.

The default value (effective when application.properties doesn't contain the setting, or it's commented out) is shown in the commented-out example:

- HOT_IRON, PET, HOT_METAL_BLUE, PET_20_STEP, SPRING, SUMMER, FALL, WINTER are exact definitions from DICOM Standard;
- SPECTRUM2 is a mathematical definition from a public source: a sweep in HSV color space from H=240 to H=0.

The second version is offered in the sample configuration files, and contains all possible palettes. The additions are:

- RAINBOW – a legacy example of a custom palette, very old and likely from a public source;
- SOKOLOFF, SPECTRUM, RAINBOW2 – added upon a customer's request, contain inexact/dull colors due to a lossy-compressed (JPEG) source image, likely still useful.

Custom palettes are stored as files `sys/clut/*.json`, file name must be listed in the setting exactly (case sensitive). The accompanying icon is expected as a file `sys/clut/icons/*.png` and is declared in form of a relative URL `palette/*.png` via the "thumbnailUrl" parameter in the JSON file.

59. Display of CAD Marks: importance of findings that are to be displayed

```
com.softneta.meddream.cad.renderingIntentFilter = all
```

Allowed values are: * `required` – selects only findings with rendering intent (111150, DCM, "Presentation Required"). This is the default; * `optional` – intents (111150, DCM, "Presentation Required") and (111151, DCM, "Presentation Optional"); * `all` – intents (111150, DCM, "Presentation Required"), (111151, DCM, "Presentation Optional") and (111152, DCM, "Not For Presentation").

Note that MedDream only knows about the intent codes listed above. `all` doesn't include any other or proprietary codes.

60. "Share via DICOM Library": upload chunk size in bytes

```
com.softneta.dicom.library.upload.partSize = 100000
```

The default is 1 MB. If the Internet connection to DICOM Library is slow, then smaller values might yield faster recovery from connection drops.

61. “Share via DICOM Library”: number of retries and related pause

```
com.softneta.dicom.library.upload.repeatTimes = 5
com.softneta.dicom.library.upload.repeatAfterSeconds = 5
```

If the upload process fails, then the related job pauses for `repeatAfterSeconds` seconds and retries up to `repeatTimes` times.

62. “Share via DICOM Library”: thread pool management

```
com.softneta.dicom.library.upload.job.corePoolSize = 10
com.softneta.dicom.library.upload.job.maxPoolSize = 100
com.softneta.dicom.library.upload.job.keepIdleForSeconds = 30
```

See https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/ThreadPoolExecutor.html. Initially, `corePoolSize` threads are ready for work; up to `maxPoolSize` threads can be created on demand and they will be terminated after `keepIdleForSeconds` seconds of inactivity.

63. Local Storage SCP: AE Title

```
com.softneta.dicomStoreService.localAETitle = MEDDREAM
```

PACSes need to know this value, the corresponding IP and port for sending images to MedDream. “MEDDREAM” is the default.

`localAETitle` also accepts multiple values separated by commas. Even more, an empty parameter will result in any Called AE Title being accepted (some PACSes, too, don’t care about the Called AE Title).

64. Local Storage SCP: TCP port

```
com.softneta.dicomStoreService.port = 11116
```

The SCP listens on this port (11116 by default).

65. Local Storage SCP: bind address

```
com.softneta.dicomStoreService.address = 192.168.111.70
```

By default, the SCP binds to 127.0.0.1. This parameter can specify a single address, including “0.0.0.0” which means all available addresses (that was the default before MedDream 8.0.0).

A symbolic name will automatically resolve to a numeric address, but this isn’t recommended unless a single address is known to the resolver.

66. Local Storage SCP: base directory for received DICOM files

```
com.softneta.dicomStoreService.saveDirectory = ${com.softneta.meddream.tempDir}/
↪STORE
```

If left empty, the SCP will not start. The default value is “STORE” under tempDir.

For better compatibility with the temporary files cleaner, empty value is recommended here instead of commenting the entire setting out.

Under this directory the SCP creates received files: CALLING_AET/ STUDY_INSTANCE_UID /SERIES_INSTANCE_UID/ SOP_INSTANCE_UID.dcm. Avoid long saveDirectory under

Windows, because together with UID values from the files the path might become longer than supported by some third-party applications.

67. Local Storage SCP: accepted Calling AE Titles

```
com.softneta.dicomStoreService.acceptAETitles = DCMSND,SENDTOPACS
```

A comma-separated list of accepted titles. It is not possible to accept "any" title.

If left unconfigured or empty, the SCP will not start. This is the default.

68. Local Storage SCP: accepted IP addresses

```
com.softneta.dicom-store-service.allowedIps = 192.168.111.87,192.168.111.88,\
    192.168.111.91
```

If not empty, then DICOM file uploads and C-ECHO connections will be allowed only from the listed IP addresses. Clients from wrong addresses will see an immediate socket closure (a connection reset).

Totally hiding the listener from some clients, as if nobody listens at the port, is possible only at a lower network layer and requires a firewall.

69. Local Storage SCP: number of simultaneous connections

```
com.softneta.dicomStoreService.multiThreadCount = 5
```

Connections beyond this number will be ignored instead of rejecting associations.

70. Local Storage SCP: try preserving compressed transfer syntaxes

```
com.softneta.dicomStoreService.preferSingleComprSyntaxInCtx = true
```

Some DICOM senders are proposing presentation contexts with multiple transfer syntaxes. The Standard allows the receiver to accept only one syntax. By default, the first one is accepted. As it's often Implicit VR LE, the sender becomes obliged to uncompress the object, which wastes time and resources.

If true is configured here, MedDream verifies whether the syntaxes are uncompressed except a single one, and accepts only that syntax. (Looks like it's a popular scenario with Sectra software.) If, however, the list contains more than one compressed syntax, then the first from the list is accepted again.

MedDream can't possibly know which syntax is the original one and therefore most efficient for sending. The sender can avoid this ambiguity by sending the original syntax first, or by packing every syntax:class pair in a separate context.

71. Local Storage SCP: create preparation jobs

```
com.softneta.dicomStoreService.prepareReceivedFile = true
```

`true`: after receiving an IAN message with availability "ONLINE", or a file over C-STORE, a preparation job will be created. During its execution the related DICOM object will be cleared from the plugin cache in order to refresh it, too. Correspondingly an IAN message with availability "UNAVAILABLE" will refresh the cached study structure from the PACS.

72. Local Storage SCP: create preparation jobs for certain Called AE Title values only

```
com.softneta.dicomStoreService.prepareForTheseLocalTitles = MEDDREAM1,MEDDREAM2
```

When prepareReceivedFile is true, this parameter can restrict values of Called AE Title for which preparation jobs are created. Applies to C-STORE only, not IAN. No restrictions if empty (default).

Unless localAETitle is empty (by default it isn't), the allowed values must be from localAETitle.

73. Local Storage SCP: different name of subdirectory for AE Title

```
com.softneta.dicomStoreService.aetToFolderMapping.TITLE1 = DIRNAME1
com.softneta.dicomStoreService.aetToFolderMapping.TITLE2 = DIRNAME2
# ...
```

Usually, when a subdirectory is created for files received from a particular AE Title, this Title is used as subdirectory name.

These settings allow you to change the subdirectory name to a different one (separate line is needed for each change). Only the subdirectory is affected; other settings like acceptAETitles will still be using the original title.

The intended usage is to deliberately "merge" two or more AEs (...aetToFolderMapping.TITLE1=TITLE2) when the PACS doesn't support forwarding but the originating device does, and there is a need to fill the plugin cache in advance when using Query/Retrieve. Therefore, the DICOM files sent by a different device can go to the same subdirectory that would be used when C-MOVE'ing those files from the PACS.

74. Local Storage SCP: allow remapping AE Titles for preparation jobs

```
com.softneta.dicomStoreService.aetMappingAppliesToPrepare = true
#com.softneta.dicomStoreService.aetToFolderMapping.TITLE3 = TITLE1
#com.softneta.dicomStoreService.prepareReceivedFile = true
```

The setting ...aetToFolderMapping above, as its name implies, only ensures a different subdirectory when a received file is saved to the disk. Starting the preparation (...prepareReceivedFile = true) might still not be possible if there is no plugin with a matching ...storeScpAet.

aetMappingAppliesToPrepare = true reuses aetToFolderMapping when searching for a suitable storage among plugins' storeScpIp values. This affects preparation jobs started by reception of not only DICOM files (C-STORE) but also IAN messages (N-CREATE).

In other contexts, like ...acceptedAETitles, the original Calling AET is still used.

A counter-intuitive example with the QR plugin. ...remoteAET=SERVER is a typical Q/R SCP (C-FIND + C-MOVE) and even forwarded files will come from SERVER. However, the annotations must be sent to a different AE, ...storeScpAet=TRIGGER, which also provides the IAN messages for preparation. If you use ...aetToFolderMapping.SERVER=TRIGGER and ...aetMappingAppliesToPrepare=true, the received files go to subdirectory "TRIGGER" and IAN messages successfully start the preparation. Now the only missing thing is that the plugin must expect the received files in that subdirectory "TRIGGER"; its cacheDir setting must point there and not to "SERVER" implied by remoteAET (in this case, a subdirectory "SERVER" is never created).

75. The Upload function

```
#com.softneta.meddream.uploader.skipFileExt = CMD
#com.softneta.meddream.uploader.skipFileNames = install,deleteTemp
#com.softneta.meddream.uploader.supportArchiveFileExt = 7z
#spring.servlet.multipart.max-file-size = 1MB
#spring.servlet.multipart.max-request-size = 10MB
#com.softneta.meddream.uploader.tempDir = \
#   ${com.softneta.meddream.tempDir}/uploader
#com.softneta.meddream.uploader.fileAliveTimeInMillis = 18000000
#com.softneta.meddream.uploader.tmpCleanAfterMilliSec = 18000000
#com.softneta.meddream.uploader.storeScuAet = \
#   ${com.softneta.meddream.dcmsnd.bind}
#com.softneta.meddream.uploader.proxyHost = 172.16.32.100
#com.softneta.meddream.uploader.proxyPort = 104
```

(continues on next page)

(continued from previous page)

```
#com.softneta.meddream.uploader.taskThreadCount = 5
#com.softneta.meddream.uploader.sendThreadCount = 1
#com.softneta.meddream.uploader.uploadThreadCount = 10

com.softneta.meddream.uploader.devices[0].id = 1
com.softneta.meddream.uploader.devices[0].storeScpIp = 127.0.0.1
com.softneta.meddream.uploader.devices[0].storeScpPort = 11114
com.softneta.meddream.uploader.devices[0].storeScpAet = DCM4CHEE
#com.softneta.meddream.uploader.devices[0].sendFilesInOneThread = 1000
```

skipFileExt and skipFileNames configure additional file names/extensions that must not be uploaded. The extensions are converted to upper case automatically, the names aren't. Afterwards the lists get augmented by hardcoded defaults, like "README" (10+ entries) and "EXE" (100+ entries). It is not possible to avoid this, or remove some value from the defaults; you can only make the blacklists more strict.

supportArchiveFileExt configure additional archive file extensions that request treating those files as containers (descending deeper into them). Values are uppercased automatically, then "ZIP" and "ISO" are added, and entries from skipFileExt (if any) are discarded.

max-file-size and max-request-size not only configure Spring but are also passed to the frontend, which can inform the user about too large files before Spring crashes on them. The values in the example are the defaults (obviously too small). MedDream has been tested with fairly large values like `2048MB`.

. . . uploader.tempDir is cleaned in the background every tmpCleanAfterMilliSec milliseconds, by removing any files and directories older than fileAliveTimeInMillis milliseconds. (Removal updates the timestamp of a parent directory, therefore empty directories might linger up to twice as long.) Negative fileAliveTimeInMillis is automatically replaced by 3600000 (one hour). In a cloud environment with multiple MedDream instances, the directory must be shared between all of them.

The files specified by the end user are uploaded into a subdirectory `<random>_upload` under . . . uploader.tempDir.

The subdirectory is then scanned for unsupported files, and those are removed.

- If the subdirectory contains a DICOMDIR file, then its references to DICOM files are used instead of collecting file and subdirectory names recursively. This recursive process continues from every found subdirectory unless a DICOMDIR file exists there, too.
- Files identified by extensions in supportArchiveFileExt are decompressed into temporary subdirectories with random names, then scanned further.
- Unsupported files, even those found via DICOMDIR, are removed. These include file extensions from skipFileExt, file names from skipFileNames, non-DICOM format, or files with SOP Class of Media Storage Directory (renamed DICOMDIR files).

The remaining files are sent over C-STORE to the user-selected device that has been chosen from com.softneta.meddream.uploader.devices[].

- "id" is just a unique alphanumeric identifier for the upload API.
- storeScpIp, storeScpPort and storeScpAet are typical parameters of a C-STORE destination; values must not be empty, port must be larger than 0.
- sendFilesInOneThread limits how many files will be sent in one uploader thread (one C-STORE session). The uploader just sends this many files in a single DICOM association; there is no attempt to make separate associations for file groups with the same SOP Class, etc. (Until 8.9.0, there was a bug where values larger than ~300 might had resulted in a too large PDU length of the A-ASSOCIATE-RQ packet.)
- The default source AE Title is . . . dcmsnd.bind, as in the Forward functionality.

- The proxy parameters specify a dedicated SOCKS5 proxy, their effect is identical to com.softneta.meddream.dcmsnd.forwardProxy*.

(It is not currently possible to reuse StorageAPI from some plugins, like the Forward function does.)

After finishing the network operations, the temporary subdirectory under . . . uploader.tempDir is removed. In essence, automatic cleaning every tmpCleanAfterMilliSec is normally not needed – only for files that somehow were not removed because Core has crashed, or was restarted, during an unfortunate time.

taskThreadCount and sendThreadCount configure thread pools (core and maximum sizes are equal) for directory scanning and network operations, respectively. uploadThreadCount is passed to the frontend and configures the number of parallel upload operations there. The default values are shown in the example; values smaller than 1 are corrected to 1.

76. Default value of (0008,0005) Specific Character Set when this attribute is missing in the DICOM file

```
com.softneta.dicomParser.defaultCharset =
```

Empty value, commented out line and `US-ASCII` are equivalent.

77. Force using `defaultCharset` above even if (0008,0005) is present in the DICOM file

```
com.softneta.dicomParser.defaultCharsetOverride = false
```

`true` will ignore any declarations in DICOM files. This might be adequate in a (small) clinic where all modalities specify wrong Specific Character Set.

78. Disable rendering of old-fashioned overlays

```
com.softneta.dicomParser.builtInOverlayEnabled = false
```

MedDream 8.2 began to render overlays that are stored in unused bits of Pixel Data. `false` here will disable that.

Overlays specified via the (60xx,3000) attribute are supported since initial versions, and cannot be disabled. Since 8.4 their display is controlled interactively.

79. Do not use the built-in DICOM Dictionary

```
com.softneta.dicomParser.useJdtLookupDDict = false
```

Since 8.3 the declared Value Representation of every DICOM attribute is ignored and is taken from a built-in dictionary instead (this setting is `true` by default).

Change to `false` for previous behavior. Some files that violate the standard might need different values of this setting. Hopefully a particular installation doesn't have both kinds of files.

80. Do not assume that a sequence with "UN" VR is encoded as ILE

```
com.softneta.dicomParser.parseUNasILE = false
```

The Standard suggests Implicit VR Little Endian for sequences with undefined length and "UN" VR. Starting from 8.7.0, this is implemented in MedDream by default.

The value `false` will switch to previous behavior where Explicit VR Little Endian was assumed. This is indeed required for a small number of incorrectly encoded DICOM files we have encountered. Unfortunately, a third-party DICOM parser in MedDream isn't smart enough to automatically adapt to both variants.

81. Allow parsing of listed private tags

By default, private tags are ignored. Consequently, standard tags in private sequences are inaccessible, too.

```
com.softneta.dicomParser.additionalPrivateTags = 00430010,2001105F/2005107B
#com.softneta.dicomParser.additionalPrivateTags[0].tagTree=2001105F/2005107B
#com.softneta.dicomParser.additionalPrivateTags[0].inEveryLevel=true
```

Simple syntax is the tag numbers (8 hexadecimal characters) separated by commas. In this syntax tags are taken only from the root level. If the number is prefixed by slash-separated container number(s), like SEQ1/SEQ2/TAG1 instead of just TAG1, then the tag is taken from specified sequence(s) starting at root level.

Extended syntax, which is commented out in the example, allows you to specify a tag number or path, and enable searching for it everywhere – not only from the root level.

Tags needed by certain functionalities, namely 00211106, 00291010 and 001910D7 (all with inEveryLevel=true), are added to this parameter automatically.

82. Register non-standard UIDs in the third-party DICOM parser

MedDream uses a third-party library for DICOM network communication and file parsing. Due to the library's internal design, an unknown UID won't allow receiving a file by the Local Storage SCP or forwarding it with com.softneta.meddream.dcmsnd.forwardingMethod=native.

Out of the box the library supports only those values of Transfer Syntax UID and SOP Class UID that are listed in the standard. However, this part of the library, and the library itself in MedDream, is not updated very often so the definitions might be simply out of date.

It is possible to work around that:

```
com.softneta.dicomParser.customSopClasses[0].uid = 1.2.840.113619.4.5.249
#com.softneta.dicomParser.customSopClasses[0].shortName = GEPrivateRTPlan
#com.softneta.dicomParser.customSopClasses[0].fullName = GE Private DICOM RT
# Plan Storage
com.softneta.dicomParser.customTransferSyntaxes[0].uid = 1.2.276.0.50.1.2
#com.softneta.dicomParser.customTransferSyntaxes[0].shortName = VisusILE
#com.softneta.dicomParser.customTransferSyntaxes[0].fullName = VISUS Implicit
# VR Little Endian
```

The .shortName and .fullName are optional and will just make MedDream logs prettier in some situations. Only .uid is important in order to recognize a new UID.

By default customTransferSyntaxes already contains three private syntaxes for VISUS:

- "1.2.276.0.50.1.2", "VisusILE", "VISUS Implicit VR Little Endian"
- "1.2.276.0.50.1.2.1", "VisusELE", "VISUS Explicit VR Little Endian"
- "1.2.276.0.50.1.2.2", "VisusEBE", "VISUS Explicit VR Big Endian"

If you define customTransferSyntaxes[0], then these three will be lost. In an installation that communicates not only with VISUS equipment but also with some other equipment that insists on private UIDs, you should manually add [0]. . . [2] with the entries above.

Note: This alone does not guarantee that a particular DICOM object will be displayed or otherwise handled. Some related logic in MedDream is totally hardcoded, and some (like detection of SRs or ECGs by a particular SOP Class UID) can essentially be configured during build time, but not in production. A new transfer syntax usually also means some new compression algorithm; that is, a corresponding codec needs to be added during build time.

The most important use case is to allow Local Storage SCP to receive the entire study, especially in installations based on Q/R.

---

83. MedCAD: path to backend plugins directory

```
com.softneta.meddream.medcad.pluginsDir = ./medcadPlugins
```

The value in this example is the default.

84. MedCAD: path to frontend plugins directory

```
com.softneta.meddream.medcad.frontendPluginsDir = ./md5/plugins
```

The value in this example is the default, and currently it's hardcoded in the frontend.

85. HIS integration: JWT security layer

```
authentication.his.jwtServiceAddress = http://192.168.111.30:90/jwt/
authentication.his.jwtServerKey = ATDwsT...DFHTQ==
authentication.his.jwtMedDreamKey = DjSWFt...psVzg==
authentication.his.responseDecryptPassword = g9RxMR...an67Z4=
authentication.his.responseDecryptSalt = /9kzgA...pYdnw==
security.cookieSecureFlag = false
```

A custom token validator can use JSON Web Tokens instead of the "token" URL parameter. (Not supported by the validator shipped with MedDream.)

This mode is enabled by adding a header `Authorization: Bearer JWT_STRING` to the initial (at least) request to MedDream. In this case jwtServiceAddress must be configured because MedDream will send the JWT in a header of a POST request to this URL and expect the same JSON object as from the usual HIS integration. The service can use claims in the token to identify what needs to be returned.

The request to jwtServiceAddress will contain the original JWT – or, if jwtMedDreamKey and jwtServerKey are configured, then MedDream verifies original signature with jwtServerKey and re-signs the token with jwtMedDreamKey (both operations support only HMAC-SHA of length 256, 384 and 512).

If responseDecryptPassword and responseDecryptSalt are configured, then MedDream will decrypt the response (AES in GCM mode, no padding) before parsing it as JSON.

The password, salt and keys must be specified in Base64 format.

If cookieSecureFlag is true, the "md-jwt" cookie is sent over HTTPS only.

86. Common encryption parameters for stateless mode

```
com.softneta.meddream.encryptionPassword = E6kLQb ... Eu/hI=
com.softneta.meddream.encryptionSalt = Om0UyI ... p2Bg==
```

Initializes a common symmetric cipher (AES with GCM, preferably 256 bits) for certain functionalities when running multiple instances of MedDream in the stateless mode (spring.profiles.include contains "stateless"). This way Export, Forward and LiveShare can begin in one instance of MedDream and continue in another one.

If encryptionPassword is empty, then the cipher is initialized with random 256-bit and 128-bit values when MedDream starts up. Otherwise, the value must contain 32 bytes encoded with Base64.

encryptionSalt must contain 16 bytes encoded with Base64 when encryptionPassword is not empty.

Both are empty by default.

87. LiveShare

```
#liveshare.guestRoles=DOCUMENT_VIEW
#liveshare.linkStrategy=SIMPLE_HIS
```

(continues on next page)

(continued from previous page)

```
#liveshare.liveShareTokenValiditySeconds=3600
#liveshare.oneLinkPerVisitor=true
#liveshare.tokensGeneratorService=http://192.168.111.30:111/v3/generate
#liveshare.tokensInvalidationService=http://192.168.111.30:111/v3/invalidate
```

LiveShare is enabled when spring.profiles.include contains "liveshare". Additionally, system.json must contain features.liveshare=true. Please remember that every guest will occupy one concurrent connection, and the host won't be notified that guests are attempting to connect but no more free connections are available; you need to plan in advance and check whether your license has enough concurrent connections.

`guestRoles` contains permissions for guest sessions. You should include only permissions for operations that you plan to demonstrate; for example,

- if EXPORT_ISO is not included but you accidentally begin exporting a disc image, then guests won't see this activity;
- if ANONYMOUS_VIEW is included then guests will not see original patient information seen by you.

If you change `linkStrategy` to TOKENIZED_HIS (default is SIMPLE_HIS), then a token service (for example, the one shipped with MedDream) is used:

1. to convert Study UIDs and permissions to a second-level token via a request to `tokensGeneratorService`,
2. for an opposite operation via `authorization.his.token-service-address`,
3. to discard second-level tokens on end of a LiveShare session via `tokensInvalidationService`.

`liveShareTokenValiditySeconds` is used during validation (first-level tokens themselves contain the creation timestamp only). Lifetime of second-level tokens is configured separately at the token service, see its parameter `com.softneta.token.cache.time-to-idle-seconds` in a different application.properties file.

`oneLinkPerVisitor` (false by default) can request that only one invitation link is valid for a guest session, and creating a new one invalidates the old one.

Furthermore, spring.profiles.include at MedDream side must contain "auth-his". For SIMPLE_HIS, `authentication.his.valid-his-params` must allow the "study" parameter even if you don't use it for integration. For TOKENIZED_HIS, `authentication.his.token-service-address` must be configured.

88. Background DICOM preparation: thread management etc.

```
com.softneta.preparation.coreThreadCount = 1
com.softneta.preparation.maxThreadCount = 40
com.softneta.preparation.workQueueSize = 8000
com.softneta.preparation.threadTimeoutSeconds = 120
com.softneta.preparation.multiFrameThreadCount = 5
com.softneta.preparation.thumbnailCreation = all
com.softneta.preparation.thumbnailCreationModalities = CT,PT,MR,NM
com.softneta.preparation.thumbnailCreationSopClasses = 1.2.840.10008.5.1.4.1.1.
↪2.2
com.softneta.preparation.prepareStudyInstancesThreadCount = 5
com.softneta.preparation.prepareStudyInstancesMaxThreadCount = 40
com.softneta.preparation.prepareStudyInstancesWorkQueueSize = 8000
com.softneta.preparation.downloadDicomCoreThreadCount = 5
com.softneta.preparation.downloadDicomMaxThreadCount = 40
com.softneta.preparation.downloadDicomQueueCapacity = 5000
com.softneta.preparation.cachePreparationFiles = false
```

(continues on next page)

(continued from previous page)

```
com.softneta.preparation.compressPixelsBeforeSave = gzip
com.softneta.preparation.compressPixelsInStepsBeforeSave = false
com.softneta.preparation.cacheTimeToIdleSeconds = 86400
com.softneta.preparation.trackDicomModifications = false
com.softneta.preparation.modalitiesForFrameExtraction = PT,NM,MG,OPT
com.softneta.preparation.sopClassesForFrameExtraction = 1.2.840.10008.5.1.4.1.1.
↪4.1,1.2.840.10008.5.1.4.1.1.2.1,1.2.840.10008.5.1.4.1.1.2.2
#com.softneta.preparation.expandEnhancedDicomThreadCount = 8
com.softneta.preparation.sopClassesForConversionToVideo = 1.2.840.10008.5.1.4.1.
↪1.3.1
com.softneta.preparation.sopClassesForMoveToNewSeries = 1.2.840.10008.5.1.4.1.1.
↪88.67,1.2.840.10008.5.1.4.1.1.88.22:SR
#com.softneta.preparation.sopClassesForScoutImageMoveToNewSeries = 1.2.840.
↪10008.5.1.4.1.1.2
com.softneta.preparation.fullScanForScoutImageMoveToNewSeries = true
com.softneta.preparation.skipExpandedFrameThumbnailPreparation = false
com.softneta.preparation.skipStructurePreparation = false
com.softneta.preparation.skipInvalidFrames = false
com.softneta.preparation.updateTagsFromPixelContent = true
com.softneta.preparation.adjustBitsStoredTagInUncompressedStream = true
com.softneta.preparation.useSessionCache = false
com.softneta.preparation.removeHtmlTagsFromSr = false
com.softneta.preparation.replicateTo = /mnt/nas/mdcache1,/mnt/nas/mdcache2
com.softneta.preparation.replicateFileThreadCount = 1
com.softneta.preparation.checkCrc = log
com.softneta.preparation.leaveHtj2kPixelsExceptTheseSopClasses = 1.2.840.10008.
↪5.1.4.1.1.4
com.softneta.preparation.prepareFromPacsMetadataExceptTheseSopClasses = 1.2.840.
↪10008.5.1.4.1.1.4
com.softneta.preparation.prepareFromPacsMetadataForTheseTransferSyntaxes = 1.2.
↪840.10008.1.2.4.202,1.2.840.10008.1.2.4.50
com.softneta.preparation.ignoreStepsFor512Resolution = true
com.softneta.preparation.skipTagPreparationOnStudyStructureFetch = false
com.softneta.preparation.nfsSynchronizationTimeoutInSeconds = 10
com.softneta.preparation.nfsSynchronizationSingleStepSleepTimeoutInSeconds = 1
com.softneta.preparation.partitions.* = SUBDIRECTORY_NAME
```

See https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/ThreadPoolExecutor.html. Initially, `coreThreadCount` threads are ready for work and up to `maxThreadCount` threads can be created on demand. The same is valid for other similar sets of parameters, prepareStudyInstances* and downloadDicom*.

Note: In some scenarios, like prefetching a series or an entire study, increasing coreThreadCount and maxThreadCount to 100-200 might not help to improve performance – CPU utilization and network traffic will still be too low. In that case, add `-Djava.util.concurrent.ForkJoinPool.common.parallelism=200` (same value as for number of threads) to the JVM command line.

Threads are terminated after `threadTimeoutSeconds` even if busy. You might need more time if some files are very large.

For every multiframe file, up to `multiFrameThreadCount` threads will be created for processing frames of the same file in parallel.

MedDream 7.7.0+ exhibits a different thumbnail display and an optimized study loading. By default, only a single "summary" thumbnail is displayed for series of modalities listed in thumbnailCreationModalities and, due to thumbnailCreation = all, remaining thumbnails are still

created in advance. thumbnailCreation = first might cause the study to be displayed quicker, at a cost of slower opening of a series after clicking on its summary thumbnail. Note, however, that `first` applies only to series with modalities in thumbnailCreationModalities; for other kinds of series, all thumbnails are always displayed and created. thumbnailCreationSopClasses is an alternative to thumbnailCreationModalities, based on a different DICOM Attribute.

The downloadDicom* parameter set is used in the "Preload series" and "Preload study" scenarios to make sure the DICOM files are downloaded and cached before processing their contents. It might make a big difference with non-direct integrations (QR, Orthanc, Dicomweb, etc.) that have the caching enabled.

`cachePreparationFiles` controls an in-memory index of cached files. In 7.8.0+ it is false by default; this saves a significant amount of memory; however outdated files are only removed by temporary files cleaner (see below). The legacy value, `false`, might improve performance and usually results in automatic removal of outdated files – even without the temporary files cleaner.

`compressPixelsBeforeSave` turns on compression during caching. Possible values are `gzip` (default), `lz4`, `false`. When enabled, **batchImageRequests is ignored and remains false**; furthermore, the legacy **"progressive loading" ("steps") for large images is disabled**, too. This might be better than server.compression.enabled as the latter is performed on every request. Both options have little sense if the network connection is fast enough. After changing compressPixelsBeforeSave, you must clear all cached data (see `com.softneta.preparation.cacheDir`). **Note:** 8.5+ doesn't support the value "true" any more, it must be changed to "gzip" – or the setting should be removed altogether as "gzip" is the default.

`compressPixelsInStepsBeforeSave` is similar to compressPixelsBeforeSave but additionally reorganizes the image lines (for "progressive download" / "steps") so that large images like CR which are downloaded this way, can be sent from cache as is – instead of decompressing a cached file, extracting lines for a particular step, compressing this data again. In this scenario, however, only "gzip" and "false" are supported. You can enable compression via this setting or via compressPixelsBeforeSave, but not both. The legacy "progressive download" requires compressPixelsBeforeSave=false, however this setting, compressPixelsInStepsBeforeSave, can have both values. After changing this setting, you must clear all cached data (see `com.softneta.preparation.cacheDir`).

`cacheTimeToIdleSeconds` is the main means to indirectly control the cache size. The preparation cache removes outdated files according to it. The cache will not work properly if files are removed by external means, like manually or by the preparation-related entry in configuration of temporary files cleaner. The latter must therefore be used only as a last resort (to clean files that became orphaned due to an application crash, etc), and its time limit must be significantly higher than cacheTimeToIdleSeconds – at least a few times higher.

If `trackDicomModifications` is true, MedDream makes sure that a cached file is created anew if the original DICOM file has a newer Last Modified timestamp. This lowers the risk of displaying outdated patient information, however if MedDream and PACS are on different machines, then their system clocks must be synchronized for better protection and to avoid repeated re-caching.

Warning: `trackDicomModifications` is intended to use only with *direct-integration plugins*. Plugins that must cache the DICOM files, like Dicomweb, have the following fundamental obstacles:

- If the temporary files cleaner is set up to keep those original DICOM files as short as possible (or caching is disabled in the plugin completely), then their absence will invalidate the prepared files, which in turn will require to re-download the originals again. The cleaning will be useless and performance will be poor.
- Even in less extreme examples, the timestamp simply corresponds to the download time and there is no mechanism to get an updated file from the PACS. As the cached files don't change, the timestamp check is not useful.

In theory, when the PACS deliberately forwards DICOM files to the MedDream listener, and

can therefore change the files in the background, then this timestamp check could still prevent display of outdated information. However, so far the QR plugin can not be configured this way.

`modalitiesForFrameExtraction` and `sopClassesForFrameExtraction` specify which objects are treated as spatial multiframe images (displayed as separate slices) instead of the traditional temporal multiframe images (for example, the "movies" in ultrasound studies). A related option, `expandEnhancedDicomThreadCount`, specifies how many threads are allocated for processing of these objects; the default value is equal to autodetected number of processors.

`skipExpandedFrameThumbnailPreparation` makes sure only the first thumbnail for spatial multiframe images is generated. The `thumbnailCreation` setting behaves similarly but for other kinds of images.

`sopClassesForConversionToVideo` lists SOP classes of temporal multiframe objects (for example, Ultrasound Multiframe Image Storage) for an experimental feature: during preparation started by DICOM Listener (local Store SCP), an equivalent video in MPEG4 format is created, and the frontend attempts to use a video player instead of a cine player. Empty by default. Also requires com.softneta.dicomStoreService.prepareReceivedFile=true, the files need to be forwarded to MedDream, sender's AE Title must be configured as .storeScpAet at the corresponding PACS plugin (see 5.1.18 Caching studies via DICOM Listener).

`sopClassesForMoveToNewSeries` (empty by default) lists SOP classes that must not be combined with other classes in the same series. If an object of this class is found in a series where other classes are also present, then it's moved to a separate virtual series dedicated to this class. For example, SR objects together with CT, MR, etc. will interfere with the MPR functionality, and even the DICOM Standard requires to have them in separate series. **Note:** entries of this list can be literal SOP classes or class:modality pairs; the latter means that a new series for objects of this class will get this dedicated Modality value, instead of reusing the value from the original series.

`sopClassesForScoutImageMoveToNewSeries` (empty by default) lists SOP Classes of scout images that need to be moved to a separate series. Some imagers place them at the beginning of an ordinary series (for example, together with CT slices). If this setting is not empty, SOP Class of the first object in series matches one of configured values and the series contains at least 5 images, then the Image Type attribute is extracted from first 5 objects, the middle and the last object. An equal value from 4th, 5th, middle and last objects is treated as the "main" one; objects with a different value are moved to a new series. Due to attribute extraction and early parsing of DICOM files, the initial Viewer window (without thumbnails) will need more time to be displayed.

`fullScanForScoutImageMoveToNewSeries` requests detection of scout images in the entire series. If false (default), only the first image is checked.

`skipStructurePreparation` (false by default) disables the call to findStudy during preparation when studyStructureSec > 0, and to findStudies when pgwStudyResponseSec > 0. Otherwise, both responses are re-cached every time when preparation is triggered due to com.softneta.dicomStoreService.prepareReceivedFile = true. A typical use case for `true` is an installation that must query the PACS as rarely as possible.

`skipInvalidFrames` ignores frames that fail to decompress; the frontend attempts to handle this gracefully, for example, by displaying the previous frame with a warning. By default, a decompression failure is fatal and MedDream stops processing/loading the image.

`updateTagsFromPixelContent` analyzes data in the pixels tag in order to detect any kind of JPEG compression and its parameters. Might help to open images with a misleading attributes like Transfer Syntax, Samples Per Pixel or Bits Stored. False by default, as this analysis costs additional CPU time.

`adjustBitsStoredTagInUncompressedStream` is similar to the previous one but for non-compressed syntaxes and incorrect Bits Stored. When the latter is smaller than required for the actual range of values (likely some DICOM actor has used a different lossy decompressor),

then the darkest or brightest image areas will contain overflow-like artifacts. Might interfere with old-fashioned overlays that are stored in higher bits of PixelData, therefore is disabled by default.

`useSessionCache` enables session subdirectories in the paths to cached files. When the session ends, those subdirectories will be removed; furthermore, the PACS integration plugins will get a corresponding event, and therefore will remove their own session subdirectories if those are supported and configured. This setup ensures that the cached files stay for a time as short as possible, however the benefit due to pre-caching of particular studies in other sessions is lost. It is also not compatible with in-advance preparation.

`removeHtmlTagsFromSr` enables conversion of HTML markup in DICOM SR objects to plain text. Such files are not standards compliant, and a security risk to viewers. MedDream displays them in plain text mode where any HTML tags are visible and make the content difficult to read; `true` here will improve the look.

`replicateTo` copies all preparation cache data related to a particular SOP Instance, to every location given by this comma-separated list. `replicateFileThreadCount` (by default 1) adjusts the number of threads used for that.

`checkCrc` (by default "off") is an experimental pixels CRC value checking for AWS HealthImaging when returning the pixels to the frontend. Compatible only with plugins of this type. A non-default value "log" just logs mismatching values, "fail" crashes the endpoint with an exception, "paused" still calculates and caches the CRCs but doesn't compare (the cache is still updated, therefore values will be present for subsequent switching to "log" or "fail").

`leaveHtj2kPixelsExceptTheseSopClasses` (experimental, related to AWS HealthImaging) lists SOP Class values for HTJ2K-compressed images that must reach the frontend in decompressed form. As a result, empty means that HTJ2K compression is preserved in all images. The value `*` (default) matches any SOP Class and therefore any occurrence of HTJ2K is decompressed.

`prepareFromPacsMetadataExceptTheseSopClasses` (experimental, work in progress) lists SOP Class values for images whose metadata must be prepared according to the traditional scenario. The value `*` means all images are handled like before 8.7.0, while an empty string (the default) forces the new scenario for all. In the new scenario, Core attempts to prepare instance metadata early – from a lot of attributes fetched from the PACS, instead of downloading the DICOM file and parsing it. It is expected that MedDream takes more time before the thumbnail placeholders are displayed, but appears faster afterwards. Currently this is supported only by the "AWSHealthImaging" plugin, while others won't do anything. The plugin must also be configured with com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER] .prepareFromPacsMetadata=true.

`prepareFromPacsMetadataForTheseTransferSyntaxes` (another work in progress) lists Transfer Syntax values to which the new "prepareFromPacsMetadata" scenario applies.

`nfsSynchronizationTimeoutInSeconds` and `nfsSynchronizationSingleStepSleepTimeoutInSeconds` are network file system synchronization delays for multiple MedDream instances that perform the preparation. Basically, when a MedDream instance can't find a cached object, then it will repeatedly test (in nfsSynchronizationSingleStepSleepTimeoutInSeconds increments) its presence for up to nfsSynchronizationTimeoutInSeconds, before starting its own preparation job.

If `ignoreStepsFor512Resolution` is true, images with exactly 512 rows will be sent in one step (one HTTP request from frontend to backend) instead of two. Default is false.

`skipTagPreparationOnStudyStructureFetch` disables creation of "tag" files for plugins having prepareFromPacsMetadata=true.

`partitions` configures an additional subdirectory for the specified type of prepared files. This way you can keep seldom-changed, small-sized files like `struct` in a separate "partition", which is scanned by a dedicated temporary files cleaner rule.

- For example, `com.softneta.preparation.partitions.struct = struct` adds a subdirectory "struct" between Storage ID and the first hash subdirectory.
  - before: `C:\cache\stg1\1f\2a\0b\dcf62d47b2...3a27dec14b\struct`

  - after: `C:\cache\stg1\struct\1f\2a\0b\dcf62d47b2...3a27dec14b\struct`

- File types not configured here are still kept together as before. Existing behavior of the temporary files cleaner effectively requires to configure partitions for every file type at once, or else the common location, which is still below the Storage ID subdirectory, needs a pattern like `??` (the first hash subdirectory is an 8-bit hexadecimal value, which means up to 256 subdirectories).
- As of 8.9.0+, recognized file types are:
  - `search` – related to pgwStudyResponseSec;
  - `struct`, `stat` – study-level resources related to studyStructureSec;
  - `cached_metadata` – another study-level resource;
  - `workspace` – also a study-level resource;
  - `meta`, `frame`, `mp4`, `mpeg`, `tag`, `thumb`, `rawsr`, `rawdoc`, `ecg_f`, `ecg`, `overlay`, `min_max`, `cadsr`, `line`, `rawmeas`, `wsi_tile` – instance-level resources.

  Here's a configuration template for all possible partitions that groups instance-level resources into a subdirectory "images" and most study-level resources – into "studies":

```
com.softneta.preparation.partitions.search=search
com.softneta.preparation.partitions.struct=studies
com.softneta.preparation.partitions.stat=studies
com.softneta.preparation.partitions.cached_metadata=studies
com.softneta.preparation.partitions.workspace=workspace
com.softneta.preparation.partitions.meta=images
com.softneta.preparation.partitions.frame=images
com.softneta.preparation.partitions.mp4=images
com.softneta.preparation.partitions.mpeg=images
com.softneta.preparation.partitions.tag=images
com.softneta.preparation.partitions.thumb=images
com.softneta.preparation.partitions.rawsr=images
com.softneta.preparation.partitions.rawdoc=images
com.softneta.preparation.partitions.ecg_f=images
com.softneta.preparation.partitions.ecg=images
com.softneta.preparation.partitions.overlay=images
com.softneta.preparation.partitions.min_max=images
com.softneta.preparation.partitions.cadsr=images
com.softneta.preparation.partitions.line=images
com.softneta.preparation.partitions.rawmeas=images
com.softneta.preparation.partitions.wsi_tile=images
```

89. Adjust number of threads for Spring Batch job processing

```
executor.jobs-task-executor.core-pool-size = 5
```

Default value is shown in the example. The same value is used both for "core size" and "max size" of a corresponding thread pool. Affects handling of preparation and Share jobs, and creating of new jobs over /manage/prepareInstances endpoint.

Before MedDream 8.5, sizes were hardcoded: 5 as core value, 10 as maximum value. Therefore, in case of very many jobs you might need to increase this up to 10 and beyond.

90. Adjust number of threads for Export functionality

```
executor.export-task-executor.core-pool-size = 5
```

Default value is shown in the example. The same value is used both for "core size" and "max size" of a corresponding thread pool.

Before MedDream 8.6, Export was using the same executor as for preparation/Share jobs.

In 8.9+, this setting is used by both kinds of legacy Export, and by MPR Export.

91. Override tags cached for the "metadata" endpoint

Tag values will be available below the "attributes" element of the response. At the moment is reserved for the V2 implementation of hanging protocols.

```
com.softneta.preparation.additionalMetadataTags = 00080070,00191018,2001105F/
↪2005107B
```

This is a comma-separated list of tags (8 hexadecimal digits). If a container (sequence) is specified, then the result will consist of values from all non-container entries found in every Item at this level and joined by a backslash character. A tag number can be prefixed by sequences containing it, as in `2001105F/2005107B` provided above; the topmost sequence, too, is relative to the root level. Private sequences also need to be listed in com.softneta.dicomParser.additionalPrivateTags.

The default value is 00080070,00080080,00081010,00081090,00082228,0008103E,00180010,00181000,00181400,00189506/00181400,00280002,00281300,00500010/00080070,00500010/00081090,00500010/00181000,00540220,52009229/00189412/00181400,52009230/00189412/00181400,00080090,00200010,00100030,00100010,00100040,00200060. If you need to add just one tag to this list, then the entire large list must be specified.

Note: Some physicians might prefer to have "00209241" in this list when viewing CT studies of the heart with ECG triggering.

Example: suppose the tags look like this,

```
00540220  SQ
  FFE000E0
    00281050 DS    [1]
    00281051 DS    [2]
  FFE000E0
    00281050 DS    [3]
```

If additionalMetadataTags contains "00540220", then attributes.00540220 is returned as `1\2\3`. If you use "00540220/00281050" instead, you'll get `1\3`.

92. Give more time for video-related preparation tasks to finish

```
com.softneta.video.preparationTimeLimitSec = 60
```

In order to start FFmpeg for video conversion, MedDream requires some data from related preparation tasks and waits up to this time for them to finish. A heavily loaded server might require more time.

In case of a corresponding timeout, the logged error message suggests to increase this value. There is little sense to do it in advance.

93. Temporary files cleaner

Temporary files are removed by their modification date. As there is no index, it is needed to scan the subdirectories in order to find any sufficiently old files.

You can configure multiple locations with a distinct file name pattern and maximum age. The setting `matchAnyDepth=false` (default is `true`) disallows searching in every subdirectory and beyond, thus helps to avoid needless scanning of underlying large directory trees. The age also applies to directories; if a directory name at some level matches the pattern, and age is suitable, then the entire directory subtree is removed even if some deeper contents are newer. There is,

however, an exception for `pattern` equal to `*`: first the files of suitable age are removed, then their parent directories if they became empty.

The cleaner executes every `cleanRateMilliSec` or according to expression in the `cron` setting. Since 8.2 execution can also start outside this schedule if `freeSpaceThresholdPct` is configured (in percent, larger than zero) and, after verification every `monitoringFrequencyMilliSec` (larger than zero), the amount of free space is found lower; it will stop as soon as there is enough free space under the path specified by `directory`.

The free space is obtained via the FileStore.getUnallocatedSpace() method that is called for a particular directory. Because directories on the same *file system volume* will have the same amount of free space, you should use the same value of freeSpaceThresholdPct for those directories, or else the smaller values might not be in effect.

`cleanRateMilliSec` is relative to the application start. The value of 86400000 (one day) means that if the application was restarted during some peak load at noon, the next cleaning will also start during peak load next day.

If a more flexible schedule is needed, then set `cleanRateMilliSec` to -1 and specify the `cron` parameter instead. The string must contain 6 entries: second, minute, hour, day, month, day of week. Examples:

- `0 0 * * * *`: beginning of every hour.
- `0 0 8-10 * * *`: 08:00, 09:00, 10:00.
- `0 0 6,19 * * *`: 06:00 and 19:00.
- `0 0/30 8-10 * * *`: 08:00, 08:30, 09:00, 09:30, 10:00 and 10:30 every day.
- `0 0 9-17 * * MON-FRI`: from 09:00 to 17:00, once per hour, except weekends.
- `0 0 0 25 12 ?`: on Christmas Day at midnight.

Since 8.2 every cleanable location can have its own values for `cleanRateMilliSec` (or `cron`) and `freeSpaceThresholdPct`.

`corePoolSize` and `maxPoolSize` help to utilize the CPU better. For example, 8 threads on a recent computer makes the operation approximately three times faster. One thread is used by default, which corresponds to pre-8.2 behavior. The default thread priority configured by `poolPriority` is 1 (the lowest possible), you can increase it to 5 (same as other threads in MedDream).

The default configuration, which is duplicated below, lists known locations with possible leftover temporary files.

```
#com.softneta.temp-cleaner.disableCleaner = true
com.softneta.temp-cleaner.cleanRateMilliSec = 1800000
#com.softneta.temp-cleaner.cron = 0 30 0 * * *
#com.softneta.temp-cleaner.monitoringFrequencyMilliSec = 0
#com.softneta.temp-cleaner.freeSpaceThresholdPct = -1
#com.softneta.temp-cleaner.corePoolSize = 1
#com.softneta.temp-cleaner.maxPoolSize = 1
#com.softneta.temp-cleaner.poolPriority = 1

com.softneta.temp-cleaner.tempItems[0].directory = ${com.softneta.\
    preparation.cacheDir}
com.softneta.temp-cleaner.tempItems[0].pattern = *
com.softneta.temp-cleaner.tempItems[0].olderThanSec = 604800

com.softneta.temp-cleaner.tempItems[1].directory = ${com.softneta.\
    meddream.tempDir}
com.softneta.temp-cleaner.tempItems[1].pattern = *_archive
com.softneta.temp-cleaner.tempItems[1].olderThanSec = 7200
```

(continues on next page)

(continued from previous page)

```
com.softneta.temp-cleaner.tempItems[1].matchAnyDepth = false

com.softneta.temp-cleaner.tempItems[2].directory = ${com.softneta.\
    meddream.tempDir}
com.softneta.temp-cleaner.tempItems[2].pattern = *_export
com.softneta.temp-cleaner.tempItems[2].olderThanSec = 21600
com.softneta.temp-cleaner.tempItems[2].matchAnyDepth = false

com.softneta.temp-cleaner.tempItems[3].directory = ${com.softneta.\
    meddream.tempDir}
com.softneta.temp-cleaner.tempItems[3].pattern = forward_*.dcm
com.softneta.temp-cleaner.tempItems[3].olderThanSec = 7200
com.softneta.temp-cleaner.tempItems[3].matchAnyDepth = false

com.softneta.temp-cleaner.tempItems[4].directory = ${com.softneta.\
    meddream.tempDir}/3d
com.softneta.temp-cleaner.tempItems[4].pattern = *
com.softneta.temp-cleaner.tempItems[4].olderThanSec = 7200

com.softneta.temp-cleaner.tempItems[5].directory = ${com.softneta.\
    video.convertedDir}
com.softneta.temp-cleaner.tempItems[5].pattern = *
com.softneta.temp-cleaner.tempItems[5].olderThanSec = 3600

com.softneta.temp-cleaner.tempItems[6].directory = ${com.softneta.\
    dicom.library.upload.tempDir}
com.softneta.temp-cleaner.tempItems[6].pattern = *
com.softneta.temp-cleaner.tempItems[6].olderThanSec = 86400

#com.softneta.temp-cleaner.tempItems[7].directory = ${com.softneta.\
#   dicomStoreService.saveDirectory}
#com.softneta.temp-cleaner.tempItems[7].pattern = *
#com.softneta.temp-cleaner.tempItems[7].olderThanSec = 86400
#com.softneta.temp-cleaner.tempItems[7].freeSpaceThresholdPct = -1
#com.softneta.temp-cleaner.tempItems[7].cleanRateMilliSec = 0
#com.softneta.temp-cleaner.tempItems[7].cron =
```

Take caution with the last entry (related to "saveDirectory"). If you uncomment it and the related `com.softneta.dicomStoreService.saveDirectory` setting is commented out, then the cleaner receives a literal string `${com.softneta...` and might misbehave. When not using the local DICOM receiver, you should either comment out (or remove) this entire cleaner entry, or set `saveDirectory` to an empty string (it is ignored both by receiver and cleaner).

The entry that cleans com.softneta.preparation.cacheDir should have a time limit a few times higher than com.softneta.preparation.cacheTimeToIdleSeconds, or else intermittent cache failures will result.

If you use some PACS plugins that download DICOM files instead of directly accessing them on the PACS file system (like "Orthanc" or "Dicomweb"), then their cache directories need to be added here, too.

If you don't want to replicate the 7+ entries above, then the absolute minimum are 3 independent locations (directories not below each other, like /mnt/DISK1/cache, /mnt/DISK1/dcm, /mnt/DISK1/temp):

1. Configure com.softneta.preparation.cacheDir with an independent directory, and clean it via a dedicated entry. This is the preparation cache that usually deserves a separate large `olderThanSec`. Do not use the default cacheDir as it is below com.softneta.meddream.tempDir.

2. Use another independent directory (not something below tempDir) for the DICOM file cache needed by a non-direct integration plugin. Clean it via a dedicated entry. Usually, its `olderThanSec` is a few days or even hours because the DICOM file cache is not used any more until the corresponding preparation cache expires.
3. This way com.softneta.meddream.tempDir is left for other, truly temporary, things and needs a third cleaner's entry. 7200 seconds might be adequate when things go well; a larger value will help when you're waiting for some bug to be reproduced, and expect to collect any related data.

94. Old jobs cleaner

```
com.softneta.preparation.scheduleJobRemovalInSeconds = 0
com.softneta.preparation.job-cleaner.cron = 0 45 0 * * *
com.softneta.preparation.h2.checkpoint-cron = 0 5 2 * * SUN
```

Completed jobs from Preparation and Share functionality are not removed immediately. The legacy behavior corresponds to `scheduleJobRemovalInSeconds = 30`, `cron = -` (a job is removed 30 seconds after completion) and is suspected to cause problems when jobs are created quite often. Starting from 7.7, MedDream will attempt to remove the jobs at a certain time (00:45 nightly by default) and only those older than one hour.

Additionally, when H2 is still used for jobs (they were not *reconfigured for MySQL/MariaDB or PostgreSQL*), the ...h2.checkpoint-cron setting can schedule execution of a hardcoded CHECKPOINT command at specified intervals, which can reduce the size of the database file after a lot of jobs are removed. "-" (default) disables execution. Scheduling will work when using other databases, too, however they either don't support CHECKPOINT or there is no gain in executing it.

95. Google Analytics ID (when using GA to track user activity)

```
com.softneta.meddream.googleAnalyticsId = UA-000000-2
```

The GA client code is already included into MedDream, you only need to specify your website ID.

96. How often the in-memory index of the cache is cleaned

```
com.softneta.ehcache.cacheCleanRate = 1800000
```

The outdated cache entries (according to com.softneta.preparation.cacheTimeToIdleSeconds) are removed every `cacheCleanRate` *milliseconds*.

97. Minimalistic persistence of the cache index

```
com.softneta.preparation.persistCache = true
com.softneta.ehcache.cachePersistRate = 3600000
com.softneta.cache.studyStructureSec = 300
com.softneta.cache.pgwStudyResponseSec = 0
```

The in-memory cache index can be made partially persistent. If `persistCache` is true, MedDream dumps the cache index to disk every `cachePersistRate` milliseconds, and loads it during startup. This way the cache index is not completely lost when restarting the service, however some orphaned files will remain. This, of course, costs CPU and I/O operations, and slows down the startup.

Experiments will be needed to find out an optimal cachePersistRate that doesn't waste too much resources and still preserves enough changes in case of unplanned application restart.

The preparation cache index is preserved in a file "cache.keys" in tempDir. Other two kinds of caches that can be preserved are: study structure cache (files "struct" in some subdirectories of cacheDir, created if `studyStructureSec` is larger than 24h) and HIS integration response cache (files "search" in some subdirectories of cacheDir, created if `pgwStudyResponseSec` is larger

than 24h). The main purpose of studyStructureSec and pgwStudyResponseSec is for how many seconds entries can remain in the corresponding cache.

The *Temporary files cleaner* above is still needed as a last-resort means. Running every few days might be sufficient.

Note: `studyStructureSec` is 300 by default since 8.0. Therefore, the study structure will not be queried from the PACS more often than this. At least 5 minutes must pass until any known changes at the PACS, like newly added series or removed images, become visible in MedDream. The "Clear cache" functionality can remove the cached structure earlier.

Note: The cache governed by `pgwStudyResponseSec` is subject to cleaning by Clear Cache functionality **but only for HIS integration by Study UID**. If your HIS integration references studies by "accnum", "patient" or "patient" together with "accnum", then the following suboptimal scenario is possible:

1. The study was removed from the PACS.
2. You have used the Clear Cache functionality.
3. The study is opened from HIS once more. At this moment MedDream will behave as if the study is still present on the PACS (will not display the "Access denied" error) but will fail shortly afterwards when fetching the study structure.

For these types of HIS integration, cache entries expire only "naturally" (by becoming too old).

---

98. Automatically expire a cached study structure if its size changes on the PACS

```
com.softneta.cache.studyStructureExpiresOnChange = true
```

The non-interactive equivalent/counterpart of structureListenerConfig in system.json. The latter can periodically detect a change and suggest a full refresh of the study; this one automatically refreshes the study when opening it, but not afterwards.

When true, on every structure request from the frontend, Core itself compares NumberOfStudyRelatedSeries / NumberOfStudyRelatedInstances with cached values (the preparation cache partition named "stat") and, if they don't match, refreshes the cached structure.

The cost: regardless of com.softneta.cache.studyStructureSec, every structure request from the frontend (mainly when the study is being opened) will cause a dedicated light-weight query to the PACS for those two attributes. The plugins implement it differently (Dicomweb even requires special configuration and still too much depends on the PACS), while some like FileSystem don't implement it at all – look for "studyStructureCountersEnabled" in this Manual. Basically, when you choose a very large value of com.softneta.cache.studyStructureSec to minimize the number of structure queries to the PACS (in general they are slow and waste the resources on the PACS machine), it becomes difficult to refresh a changed study at MedDream's side. The Clear Caches functionality is adequate only as a last resort. The dedicated query for those two attributes is, in an ideal case, far more efficient than the full structure fetch, therefore your PACS should be able to afford executing it often.

Disabled (false) by default.

Note: If the preparation cache has been populated by an older version of MedDream, or with this setting disabled, then the cached values in the "stat" partition are missing. MedDream will deliberately not invalidate a cached structure just because they are missing, as otherwise an installation that caches a many years' worth of prepared studies, would effectively require to regenerate this data. In such cases it's recommended to use structureListenerConfig instead.

This behavior might become configurable in the future if different customer expectations become clear.

99. Method used by the Forward functionality

```
com.softneta.meddream.dcmsnd.forwardingMethod = native
#com.softneta.meddream.dcmsnd.forwardingMethod = plugin
#com.softneta.meddream.dcmsnd.forwardingMethod = cmove
```

There are three implementations:

- `native` (default): MedDream reads DICOM files of the study via a PACS integration plugin (like during image display) and sends the contents via the same C-STORE client used for uploading annotations. Elements of "forwardPacs" (system.json) specify destination's IP address, TCP port and AE Title;
- `plugin`: MedDream uses the PACS integration plugin to create a forwarding job on the PACS itself and monitor its progress. TCP/IP address and port in "forwardPacs" are not used. Implemented only in the PacsOne plugin;
- `cmove`: to forward a study found via a particular PACS integration plugin, MedDream uses a C-MOVE client to connect to the PACS according to storeScpIp, storeScpPort, storeScpAet in the configuration of that plugin, and passes Study Instance UID and destination's AET. TCP/IP address and port in "forwardPacs" are not used, however one might need to configure the `sender` element: it lists plugin identifiers and so indicates PACSes whose studies can be sent to this destination (in other words, it lists PACSes that have this destination configured). Empty `sender` means "all PACSes". When multiple studies are selected in Search window, the "cmove" method attempts to send as many of them as possible (does not stop even if errors occur), and the user will be informed about the most severe condition; for example, if one study failed completely (even number of objects is unknown) but another one went with some warnings, then those warnings and total number of objects are considered not important, only that total failure is indicated.

100. Number of threads for Forward functionality

```
com.softneta.meddream.dcmsnd.dicomParseThreadCount = 5
com.softneta.meddream.dcmsnd.dicomSendThreadCount = 1
```

dicomParseThreadCount (5 by default) is used only with forwardingMethod=native and influences the delay after which the network activity begins. During that period MedDream collects paths to DICOM files, or makes local copies of files if the PACS plugin doesn't support paths (for example, an HTTP-based integration like Orthanc with caching disabled). More threads might reduce the delay, however a direct DB integration plugin like Conquest will make more parallel requests to the database.

dicomSendThreadCount (1 by default) is for all kinds of forwardingMethod. `native` will perform up to this number of parallel C-STORE sessions (one DICOM Association per file); note, however, that the limit applies to all destinations simultaneously. `plugin` and `cmove` will be able to send up to this number of studies in parallel, therefore a larger number might improve the experience of Forward for concurrent users.

101. SOCKS5 proxy address for the Forward function

DICOM receivers might validate the source IP address of an incoming session. Suppose the devices already know the address of the PACS machine; a new address (of MedDream machine) cannot be added to them due to some reason, while MedDream must run on a different machine than the PACS. The solution is to set up a SOCKS5 proxy and run its server part on the PACS machine. MedDream, in turn, must know about the client part of the proxy:

```
com.softneta.meddream.dcmsnd.forwardProxyHost = 192.168.111.222
com.softneta.meddream.dcmsnd.forwardProxyPort = 4567
```

If both parameters are specified, then the Forward function and upload of annotations (PR, KO, segmentations, screenshots) will pass all DICOM communication to the specified proxy so that the traffic comes out from a different IP address. For MedDream there is no significant difference where the client part is running: on the PACS machine, on the MedDream machine, or on a totally different machine.

Setting up such a proxy is beyond the scope of this document. OpenSSH is suitable both for Linux and Windows; Shadowsocks is a good alternative.

102. Use the original AE Title when sending annotations

When sending annotations to the PACS, the C-STORE client uses its own AE Title from non-empty value of com.softneta.meddream.pacs.configurations[].storeScuAet, then from com.softneta.meddream.dcmsnd.bind.

However, if this parameter is true, the first value attempted is the original AE Title already associated with the study.

```
com.softneta.meddream.dcmsnd.annotationsFromOrigAet = true
```

NOTE: this was tested only with PacsOne plugin. Some plugins will not return the study's AE Title at all, while some others might return multiple values from different series and this is not handled properly at the moment.

103. Always save a forwarded file (forwardingMethod=native)

```
com.softneta.meddream.dcmsnd.forceLocalSave = true
```

The default is false. When a plugin can't return a path (for example, a non-direct integration plugin is not configured to cache the DICOM files and therefore returns a new online copy every time), Core logs a warning and copies the contents to a temporary file. (forwardingMethod=native always requires some file to work on.)

`true`: assume there is no path support in PACS plugin(s) and always use a temporary file. In an installation where *some* plugins do not support paths, this will prevent the corresponding warning in the log, however will also waste time and disk space for the remaining plugins. (Adjusting the log level, like logging.level.com.softneta.meddream.forward.task.MedDreamForwardTask=ERROR, is a worse idea because it prevents some other messages as well.)

104. Disable sorting of toolbar buttons after a version upgrade

```
com.softneta.meddream.toolbarButtonsOrderingEnabled = false
```

By default toolbar buttons are reordered after the upgrade (when the new JAR reads the files *global.json saved by an earlier version). This way newly added buttons don't go unnoticed.

If `false` is specified here, then no reordering is done and new buttons are added at the end. Useful for some installations with a customized list of buttons where it would be too time-consuming to recreate the custom order manually.

105. Hide series by Series Description

```
com.softneta.meddream.blacklist.seriesDescriptions = AiSeq ,_3D_,REPROJECTION
```

A series will be hidden if its Series Description attribute contains a fragment from this comma-separated list.

106. Preserve new MedDream annotations in the series restriction in the token

```
com.softneta.meddream.blacklist.tokenSeriesRestrictionPreservesAnnotations =␣
↪true
```

The token for HIS integration can contain a restrictions.series element for whitelisting only relevant series by their UIDs. Initially this filter was implemented too strictly and, as a result, was also removing some annotations freshly created by MedDream.

This setting of 8.8.0+ attempts to minimize the interference:

- the value `true` (default) means that the filter is applied as late as possible, thereby producing only a display-related effect. At that point PR, KO and RTSTRUCT annotations are already removed from the study structure as the frontend is not supposed to reference them directly. The Secondary Capture annotations, though, must remain visible, therefore the filter logic doesn't apply to objects with Series Description "Secondary captures" and Modality "OT";
- the value `false` returns the earlier behavior just in case.

107. Calculation of "main modality" for the study structure

Some modality-related decisions become too complicated with studies having multiple modality values (like "CT, KO, PR, SR"), compared to single-modality studies (like "DX").

```
com.softneta.study.modalityWeight.CT = 1200
com.softneta.study.modalityWeight.PT = 100
```

The number of series with a particular modality is multiplied by configurable weight and the modality with the highest score "wins". There is an extensive list of default weights, with 1000 for many popular values like CR or ECG, then 100 for SR, 500 for OT, 2000 for CT/MR/US/XA, 100 000 for PT/NM.

In the example above, a typical study with both CT and PT series is still considered as "CT" because CT has a higher weight assigned. If, however, there are more than 12 PT series for 1 CT series, the result will be "PT".

108. List of modalities to move series to the end of the study structure

```
com.softneta.study.lowerPrioritySeriesModalities = OT,SR
```

Empty by default (no series reordering). The `sCSeriesLowerPriority` setting of Dicomweb plugin has a similar effect but is implemented differently.

109. Disable the "License will expire in . . . days" message

```
com.softneta.license.displayLicenseExpireWarningMessage = false
```

By default is "true" and the message is displayed when less than a few days are remaining.

This warning is displayed for all users so that they, too, can remind the administrator and the management about the upcoming switch to demo mode because of an outdated license.

If you have bought a cheaper time-limited license, then the best way is not to wait until this warning is displayed but extend the license at least 7 calendar days earlier. Although the message is a useful reminder, it might needlessly annoy the end users until the license is updated, and in such cases `false` will help; we recommend to return the value `true` when updating the license.

110. Change the rate of license file changes monitoring

```
com.softneta.license.checkLicenseFileEverySec = 20
```

A changed meddream.lic can be loaded automatically at this rate. Changes will apply immediately in the backend, however the frontend needs to be reloaded manually in every session. At least the administrator himself should open MedDream after replacing the license file, in order to avoid surprises.

The default is 60 for Redis and Ignite, and zero (disabled) otherwise. This was hardcoded in versions 8.1.0 . . . 8.7.0.

111. Redis configuration

This applies to the "redis" application profile.

Before 8.9.0, a setting named `redis.config-yaml` (normally passed via an environment variable) used to pass the path to a non-default configuration YAML file. 8.9.0+ expects *contents of this file* (which are YAML strings, potentially multiline) passed via separate variables.

```
redis.config.server = ...
redis.config.cache = ...
#redis.token-refresh-rate-in-minutes = 30
```

The refresh rate has been added for Azure Entra tokens, which might otherwise become invalid. This parameter is indeed expected in application.properties.

Environment variables AZURE_REDIS_USER_NAME and AZURE_CLIENT_ID must be used to pass Entra credentials.

A possibility to specify a YAML fragment in application.properties hasn't been tested, though. The actual use case involves a Kubernetes installation where this configuration is passed via environment variables REDIS_CONFIG_SERVER and REDIS_CONFIG_CACHE defined in a YAML file (thus multiline values are supported natively via the "pipe" syntax).

The default configuration (YAML format) is provided below, and offers an example on how values of redis.config.server and redis.config.cache should look.

```yaml
redis:
  config:
    server: |
      singleServerConfig:
        address: "redis://127.0.0.1:6379"
    cache: |
      PREPARATION:
        ttl: 0
      RESPONSE:
        ttl: 3600000
      PREPARATION_MESSAGE:
        maxIdleTime: 15000
      TASK_MESSAGE:
        maxIdleTime: 15000
      TOKEN:
        maxIdleTime: 10000
      SESSION:
        maxIdleTime: 60000
      TRACK_TABS:
        ttl: 86400
```

5.1.14. Opening the H2 Console

Warning: MedDream sessions are incompatible with H2 Console sessions. Use the incognito mode or another browser if MedDream must be used at the same time.

Otherwise, you will need to log out fully every time. The H2 Console has a "Disconnect" button. MedDream toolbar has a "Log Off" button, which is renamed to "Close" in HIS integration mode (and logs off, even if the window fails to close afterwards).

1. Add to (or uncomment in) application.properties, then restart the application:

```
spring.h2.console.enabled = true
spring.h2.console.settings.web-allow-others = true
authentication.manager.username = ADD_USER_NAME_HERE
authentication.manager.password = ADD_STRONG_PASSWORD_HERE
```

2. Open SERVER_IP:SERVER_PORT/h2-console in the browser. The browser will request HTTP credentials configured above, then display Console's own graphical login form.
3. In the Console login form, make sure the *JDBC URL* form field contains `jdbc:h2:file:MEDDREAM_ROOT_DIR/sys/database/DB_NAME` (with real path instead of MEDDREAM_ROOT_DIR, and real database name instead of DB_NAME).

   Example: `jdbc:h2:file:C:/PROGRA~1/PacsOne/php/meddream/sys/database/reportdb`.
4. *User Name* is normally "sa", however the "authdb" database requires an empty value. *Password* must always be empty.
5. Press the *Connect* button.
6. You should now see some table names at the left, unless an empty database is expected.

Warning: Leaving H2 Console enabled for a long time is a security risk. Disable it as soon as the work is done, with `spring.h2.console.enabled = false`.

The safest way is to always use `spring.h2.console.settings.web-allow-others = false`: the Console will then open on localhost only. This is OK for Windows machines, as they are often managed via Remote Desktop or similar graphics-capable means, and a browser can be run on the server itself. For Linux machines accessed via SSH, we advise to tunnel the corresponding port (likely localhost:80) to some other port on operator's machine, and open a browser there.

5.1.15. Using MySQL/MariaDB or PostgreSQL for Spring Batch and Quartz Scheduler

By default, MedDream uses H2 for all built-in databases. The reason is simplicity: they can be created automatically due to the nature of this DBMS.

MySQL/MariaDB or PostgreSQL are supported as alternatives. This is advised for installations with high loads (for example, large hospitals with preparation enabled). Switching is easier if done shortly after installation, and involves the following steps:

1. Create two new databases. We suggest names `mdquartz` and `mdbatch` although any names will be OK. It is essential to have different dedicated databases instead of reusing some existing one.
2. Create a dedicated user that has access only to these databases and only from a specific machine. **Avoid using superusers ("root", "postgres").** Example for MySQL/MariaDB:

   ```
   GRANT ALL PRIVILEGES ON mdbatch.*  TO 'mdjobs'@'localhost' IDENTIFIED BY 'PASS';
   GRANT ALL PRIVILEGES ON mdquartz.* TO 'mdjobs'@'localhost' IDENTIFIED BY 'PASS';
   FLUSH PRIVILEGES;
   ```

   Example for PostgreSQL:

   ```
   CREATE ROLE mdjobs WITH LOGIN PASSWORD 'PASS';
   GRANT ALL PRIVILEGES ON DATABASE mdbatch TO mdjobs;
   GRANT ALL PRIVILEGES ON DATABASE mdquartz TO mdjobs;
   ```

   (Reminder: the IP address range allowed for a user in PostgreSQL is specified in the server's `pg_hba.conf`.)

   Substitute a new password for PASS. Do not reuse existing ones as this password will be visible in the configuration file.

3. Add the following to application.properties for MySQL/MariaDB

```
datasource.quartz.url = jdbc:mariadb://SERVER:PORT/mdquartz?user=USER\
&password=PASS
datasource.quartz.driverClassName = org.mariadb.jdbc.Driver
datasource.batch.url = jdbc:mariadb://SERVER:PORT/mdbatch?user=USER\
&password=PASS
datasource.batch.driverClassName = org.mariadb.jdbc.Driver
```

   or for PostgreSQL

```
datasource.quartz.url = jdbc:postgresql://SERVER:PORT/mdquartz?\
user=USER&password=PASS
datasource.quartz.driverClassName = org.postgresql.Driver
datasource.batch.url = jdbc:postgresql://SERVER:PORT/mdbatch?\
user=USER&password=PASS
datasource.batch.driverClassName = org.postgresql.Driver
spring.quartz.properties.org.quartz.jobStore.driverDelegateClass =\
org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
```

   SERVER, PORT, USER and PASS must be changed to real values (in Step 2 above, USER is "mdjobs"). If you assigned different names instead of `mdquartz` and `mdbatch`, then they must be changed as well.

4. Start the MedDream service.
   - In case of fatal errors, like an unrecognized password, the service will fail to start. Check whether it's still running after a couple of minutes.
   - Look for any new ERROR messages or stack traces in the log.
   - Verify that a few tables, including `flyway_schema_history`, have been created in each database.
   - Try using functionality that involves background jobs, like in-advance preparation (triggered either by IAN messages or forwarded files) or Share via DICOM Library.

At the moment we don't provide instructions on how to transfer the contents of these databases, as in most cases they are of little value and can be recreated at virtually no cost.

Note: In fact, the set of ALL PRIVILEGES is only required during:

- the first run of MedDream service in a new installation (it will create the tables),
- the first run of a new version (it might adjust the structure).

Afterwards one can use a smaller set, namely, SELECT, INSERT, UPDATE, DELETE.

This hardening has a cost, however: just before every upgrade of MedDream you will need to restore ALL PRIVILEGES for a short time. If you forget that, you'll get an error like "ALTER command denied to user", and the flyway_version_history table will remain in a broken state: retries will result in the error "Schema ... contains a failed migration". The latter can usually be fixed by removing the newest record from the table (look at the "installed_on" timestamp); however, a safer way is to restore the entire database from a backup.

5.1.16. Changing the HTTP context path

Sometimes one needs a non-default context path, like `http://server.tld/meddream/` instead of just `http://server.tld/` (for example, to isolate multiple installations of MedDream).

A proven solution involves accessing the backend via a proxy that changes every "/meddream" to "/meddream/". For Apache, the following mod_rewrite directive solves the problem:

```
RewriteRule ^/meddream$ /meddream/ [R]
```

5.1.17. Adjusting a reverse proxy

If a reverse proxy is used, it might interfere with websockets. Currently MedDream depends on the latter for configurations with features.liveShare=true or openTabsTrackingMethod=SOCKETS (both can be enabled in system.json).

A most common failure is due to the Origin header added by proxies, that results in a message "Handshake request rejected, Origin header value https://127.0.0.1 not allowed" in MedDream logs. To correct that, you can add the following to the **proxy configuration**:

- Apache: `RequestHeader unset Origin` (requires mod_header);
- Nginx: `proxy_set_header Origin "";`

A full example – Apache's httpd-ssl.conf

This reverse proxy is used to add HTTPS to the MedDream backend. "java-app" is the Docker-local domain name of the corresponding machine, and "8080" is its port.

```
Listen 443
SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4:!3DES
SSLProxyCipherSuite HIGH:MEDIUM:!MD5:!RC4:!3DES
SSLHonorCipherOrder on
SSLProtocol all -SSLv3
SSLProxyProtocol all -SSLv3
SSLPassPhraseDialog  builtin
SSLSessionCache        "shmcb:/usr/local/apache2/logs/ssl_scache(512000)"
SSLSessionCacheTimeout  300

<VirtualHost _default_:443>
    ServerName testme.meddream.com
    ErrorLog /proc/self/fd/2
    TransferLog /proc/self/fd/1

    SSLEngine on
    SSLCertificateFile "/usr/local/apache2/conf/server.crt"
    SSLCertificateKeyFile "/usr/local/apache2/conf/server.key"
    ProxyPreserveHost On
    ProxyRequests Off

    # Proxy HTTP requests
    ProxyPass / http://java-app:8080/
    ProxyPassReverse / http://java-app:8080/

    <Location "/liveshare-websocket">
        RequestHeader unset Origin
    </Location>

    # Proxy WebSocket requests
```

(continues on next page)

(continued from previous page)

```
    ProxyPass /ws ws://java-app:8080/
    ProxyPassReverse /ws ws://java-app:8080/

    RewriteEngine On
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteCond %{HTTP:Connection} upgrade [NC]
    RewriteRule /(.*) "ws://java-app:8080/$1" [P,L]
</VirtualHost>
```

A full example – Apache's httpd-vhosts.conf

A companion file from the same example, which allows the end users to additionally access MedDream without SSL at the same time. This circumvents the existing limitation of Core, which requires choosing either HTTP or HTTPS, but not both.

```
<VirtualHost *:80>
    ServerName testme.meddream.com
    ProxyPreserveHost On
    ProxyRequests Off

    # Proxy HTTP requests
    ProxyPass / http://java-app:8080/
    ProxyPassReverse / http://java-app:8080/

    # Proxy WebSocket requests
    ProxyPass /ws ws://java-app:8080/
    ProxyPassReverse /ws ws://java-app:8080/

    RewriteEngine On
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteCond %{HTTP:Connection} upgrade [NC]
    RewriteRule /(.*) "ws://java-app:8080/$1" [P,L]
</VirtualHost>
```

5.1.18. Caching studies via DICOM Listener

Note: MedDream 8.0.0+ enables most options by default; sender's AE Title still needs to be specified manually.

The DICOM Listener (local Store SCP) can run in parallel to any kind of PACS integration.

Non-direct integration plugins like Orthanc, Dicomweb and QR have their own caches with directory tree compatible to that of the SCP. A plugin cache directory setting can point to a corresponding subdirectory of the SCP cache, then the plugin will pick up files sent to the SCP in advance.

The Listener can also automatically start the preparation of a received file. This works even with direct integration plugins, at a cost of some disk space because the file is then received only for triggering the preparation (based on UIDs from the file) and can be removed shortly afterwards; the preparation reads its own copy of the file from the PACS file system. So far, the direct integration plugins can't reuse the SCP cache (for example, this could prevent a slower on-demand reading from network storage).

Note: Starting from 7.8.0, there is an alternative preparation trigger that saves both disk space and network traffic: the PACS can send DICOM Instance Availability Notifications instead of files. (The Listener implements both Store SCP and IAN SCP.)

Therefore, for a significant performance boost one can configure the PACS to forward studies to MedDream, and adjust related MedDream parameters:

```
# These two must be configured or else the Listener won't start up.
com.softneta.dicom-store-service.port=11116
com.softneta.dicomStoreService.acceptAETitles=pacs

# It's possible to assign a different AE Title to MedDream.
com.softneta.dicom-store-service.localAETitle=MEDDREAM

# Cache root directory. A subdirectory with sender's AE Title (only values from
# acceptAETitles) is created automatically.
com.softneta.dicom-store-service.saveDirectory=file:///opt/meddream/temp/STORE/

# If true, the Listener creates preparation jobs with study/series/image UIDs
# from received files or IANs.
# True by default since 8.0.0.
#com.softneta.dicomStoreService.prepareReceivedFile=true

# If true, then preparation jobs are executed; otherwise, they remain in paused state
# after creation.
#True by default.
#com.softneta.preparation.enabled=true

# If true, the cache index is periodically saved into a file "cache.keys" under
# com.softneta.meddream.tempDir, and loaded during application startup.
# True by default since 8.0.0.
#com.softneta.preparation.persistCache=true

# Plugin configuration. An AET listed in acceptAETitles is needed in storeScpAet,
# and dicomCacheDirectory must end with a corresponding subdirectory.
com.softneta.meddream.pacs.configurations[0].storeScpAet=pacs
com.softneta.meddream.pacs.configurations[0].dicomCacheDirectory=file:///opt/meddream/\
    temp/STORE/pacs/

# More plugin configuration: when the DICOM Listener triggers preparation for a
# "direct integration" plugin, the received file is removed afterwards unless
# the following is false. Might be needed for plugins like Dcm4chee2 and Dcm4chee5
# which fall back to Query/Retrieve mode in some scenarios.
#com.softneta.meddream.pacs.configurations[0].clearPrepareReceivedFile=false
```

Warning: The .storeScpAet setting is used to find the plugin that will be used during preparation, and decide whether the received file is to be removed afterwards.

If you have multiple plugins, then configure a different and not empty .storeScpAet for each plugin. It might be required to assign some fake and non-realistic AETs. Consequently, a setup where the same PACS is accessed both via a native plugin and via QR, becomes problematic as only one of them can upload annotations/screenshots via the universal mechanism. Probably using .storageApiEnabled=true for the QR plugin can help there, as the plugin will then use its own .remoteAET instead.

5.1.19. Monitoring via Prometheus/Grafana

1. Enable the status endpoint in application.properties and restart the service:

```
management.endpoint.prometheus.enabled=true
management.endpoints.web.exposure.include=prometheus
management.server.port=8081
management.server.address=127.0.0.1
```

   This configuration makes /manage/prometheus available on 127.0.0.1:8081. Non-default port and address are important because this endpoint (unlike others under /manage) does not require authentication.

   For a quick test, start a web browser (or a tool like curl) on the server and open http://127.0.0.1:8081/manage/prometheus. It should return certain text-like data:

```
# HELP jvm_gc_memory_allocated_bytes_total Incremented for an increase in the␣
↪size of the (young) heap memory pool after one GC to before the next
# TYPE jvm_gc_memory_allocated_bytes_total counter
jvm_gc_memory_allocated_bytes_total{application="MedDream 7.8.0",} 1.807134632E9
...
```

2. Install Prometheus on the same server as MedDream

   See https://prometheus.io/docs/prometheus/latest/installation.

   Because MedDream exposes the data on the same host only, software from other hosts won't be able to access it. Exposing the data publicly would be a serious security risk.

3. Add MedDream status endpoint to prometheus.yml

   See https://prometheus.io/docs/prometheus/latest/configuration/configuration.

   scrape_configs.metrics_path must be `/manage/prometheus`. scrape_configs.static_configs .targets must contain the IP and port pair of the status endpoint. Example:

```
scrape_configs:
  - job_name: 'prometheus'
    metrics_path: '/manage/prometheus'
    scheme: 'http'
    static_configs:
      - targets: ['127.0.0.1:8081']
```

   By default, Prometheus listens on all IP addresses and port 9090. Like the status endpoint, Prometheus itself should be available on localhost only. Address and port are changeable not in prometheus.yml but via command line parameter `--web.listen-address` (for example, in the `prometheus.service` wrapper under Linux with systemd).

   For a quick test, open http://127.0.0.1:9090/ on the server, enter "system_cpu_usage" as Expression, press Execute and choose the Graph tab below.

4. Install Grafana on the same server as MedDream

   See https://grafana.com/docs/grafana/latest/setup-grafana/installation/.

5. Add Prometheus as data source

   Open http://localhost:3000/ in the browser. Choose "Add your first data source" on the first page. Choose "Prometheus" from the list. Enter "http://localhost:9090/" as URL, and press the "Save & test" button at end of the page. A message "Data source is working" should appear above.

   See https://grafana.com/docs/grafana/latest/administration/data-source-management/ for more details.

6. Add a dashboard

Choose "Create your first dashboard" on the first page. Press the plus sign on the left and choose "Import". Under "Import via grafana.com", enter `4701` (or `10280`) and press Load. Specify the data source created earlier, and press Import.

See https://grafana.com/docs/grafana/latest/visualizations/dashboards/build-dashboards/import-dashboards/ for more details.

5.1.20. Implementing per-user settings

When the identity of a user without the "ADMIN" permission is known (the user has logged in via the login form, or a HIS integration token provides the "id" field), then MedDream attempts to load a custom file `<login>_global.json`.

If that fails, and the user belongs to one or more groups, then the first group name is chosen for the second attempt with another custom file `<group>_g_global.json`.

As a last resort there is one more attempt with the traditional file `global.json`.

Note: Some settings are global by design and will be ignored in a per-user settings file. The Settings window warns about that when per-user settings are being edited.

As of 2026, these are:

- *Left Side Information* and *Right Side Information* under *Viewer* > *Info Label properties*.

Note: The legacy "insecure" HIS integration (URL-based, not token-based) does not support user identities and always uses the common global.json file.

Note: The user account that will use a custom settings file, must not have the "ADMIN" permission. A login form account can have this permission individually like `authorization.users[0].role=ADMIN`, however a value common to all users is possible in `authorization.defaultLoginPermissions`. In case of HIS integration, the `.permissions` array in the token is the only possible source.

Note: If global.json was renamed via com.softneta.settings.globalFileName, then `<login>_` or `<group>_g_` is prepended to the configured file name.

Note: Before 8.6, per-group settings were stored in `<group>_global.json`, which becomes a problem when using an authentication method that allows a group name to match a user name. You need to rename per-group files during the upgrade.

Starting from 8.1, a user **with "USER_SETTINGS" permission (but without "ADMIN" permission)** can save to `<login>_global.json` directly, even if the file doesn't exist yet. This implies that afterwards he won't see any changes made by the administrator to the common file `global.json`, as his own settings file is loaded first.

Starting from 8.2, a user with "SHORTCUTS_EDIT" permission can edit keyboard/mouse shortcuts. If "ADMIN" is also present, then the global file `shortcuts.json` is edited, while users without "ADMIN" permission will save to `<login>_shortcuts.json` instead. Furthermore, even without "SHORTCUTS_EDIT" the user will load an existing individual file which then is read only.

Note: Starting from 8.6, you can also have per-group shortcuts file that is named `<group>_g_shortcuts.json`.

This file is handled at a lower priority than `<user>_shortcuts.json`. Particularly, if group shortcuts are loaded and then the "Reset to default" button is used, the per-user file becomes empty and still overrides the per-

group file. You'll need to manually remove this per-user file afterwards. This situation is only possible if SHORTCUTS_EDIT permission is present. **Currently, SHORTCUTS_EDIT is incompatible with per-group shortcuts.**

8.2+ also keeps the hanging protocols settings in separate files, `hangingProtocols.json` and `hangingProtocolsV2.json`. The logic is similar to other customizable files. Users with "ADMIN" permission are always editing this file. Other users can load `<login>_hangingProtocols*.json` in read only mode, and if the "HANGING_PROTOCOLS_EDIT" permission is present, then this file becomes editable. If a user has "HANGING_PROTOCOLS_EDIT" instead of "ADMIN", and the individual file doesn't exist yet, then it's created when saving the hanging protocols settings. The per-group version needs to be named `<group>_g_hangingProtocols.json` or `<group>_g_hangingProtocolsV2.json`, respectively.

As of 8.2+, columns.json is not personalizable.

An administrator (who has the "ADMIN" permission) can only view/edit the common `global.json` file. If giving the USER_SETTINGS permission is unsuitable due to some reason, then it is still possible to prepare personalized read-only settings files as follows:

1. Make a copy of global.json.
2. Log in as an administrator (the account must have the "ADMIN" permission), open the Settings window, make all planned changes and save the settings.
3. Rename the updated global.json to `<login>_global.json`. This `<login>` is an exact value of the user name from the login form, or of `.user.id` from the HIS integration token. Leave the custom file in the same directory.
4. Restore global.json from the copy.
5. To verify, log in as an ordinary user (without the "ADMIN" privilege) from the login form, or open a study from HIS via the token.

Warning: When using PacsOne authentication ("auth-pacsone" in spring.profiles.include), the "System Administration" privilege is converted to the "ADMIN" permission of MedDream. Consequently, such an user will be able to access and change the common settings only.

Warning: You need to plan in advance whether the file system, where the settings files are stored, is case sensitive.

- If the system is case insensitive (by default under Windows, however this can be changed since Windows 10), then you won't be able to create separate files for "User" and "user";
- If the system is case sensitive (in most cases under Linux), then you need to keep attention when creating per-user files in advance, or renaming users/groups and adjusting on-disk names.

A safer way is to always assign user/group names using the same naming convention – for example, PascalCase or just plain lowercase.

Warning: In case of the shortcuts file (not applicable to other .json files), user and group names should contain only characters "A"... "Z", "a"... "z", "0" ... "9", "-" and "_". Any other character will be discarded before trying to load the corresponding file. This prevents problems with characters not allowed in file names (like "/") or national characters converted to rubbish on server's file system.

The manual duplication steps above are valid not only for global.json but also for other personalizable settings files.

Example of a HIS integration token with the login name:

```
{
    "items": [
        {
            "studies": {
                "study": "1.2.826.0.1.3680043.8.1055.1.2020050611210409.446757335",
                "storage": "OrthancDicomweb"
            }
        }
    ],
    "permissions": [ "PATIENT_HISTORY", "FORWARD", "DOCUMENT_VIEW" ],
    "user": {
        "id": "jdoe",
        "name": "John Doe"
    }
}
```

This token suggests using settings from the file jdoe_global.json, etc.

5.1.21. Custom columns in Patient History / Patient Studies

MedDream 8.2 offers customizable columns in the Patient History (opens from the thumbnail strip) and Patient Studies (opens automatically in a certain token-based scenario). The Search window is not customizable so far.

Customization must also be supported by PACS integration plugins. So far those are only QR and Dicomweb plugins.

A new file sys/settings/columns.json configures the GUI. The default contents are:

```
{
    "patientHistoryColumns": [
        {
            "name": "openStudyColumn",
            "parameters": [
                "uid",
                "storageId",
                "sensitive",
                "restricted"
            ],
            "type": "openStudy",
            "customClass": "flex-basis-9 center"
        },
        {
            "name": "reportColumn",
            "parameters": [
                "hasReport"
            ],
            "type": "report",
            "customClass": "center"
        },
        {
            "name": "patientId",
            "parameters": [
                "patientId"
            ],
            "type": "string",
            "labelTranslation": "common.id"
        },
```

(continues on next page)

(continued from previous page)

```
        {
            "name": "patientName",
            "parameters": [
                "patientName"
            ],
            "type": "string",
            "labelTranslation": "common.name"
        },
        {
            "name": "modality",
            "parameters": [
                "modality"
            ],
            "type": "string",
            "customClass": "flex-basis-11",
            "labelTranslation": "common.modality"
        },
        {
            "name": "description",
            "parameters": [
                "description"
            ],
            "type": "string",
            "labelTranslation": "common.description"
        },
        {
            "name": "dateTime",
            "parameters": [
                "date",
                "time"
            ],
            "type": "dateTime",
            "labelTranslation": "common.dateTime"
        },
        {
            "name": "sourceAE",
            "parameters": [
                "sourceAE"
            ],
            "type": "string",
            "customClass": "flex-basis-15",
            "labelTranslation": "common.sourceAE"
        }
    ],
    "patientStudiesColumns": [
        {
            "name": "openStudyColumn",
            "parameters": [
                "uid",
                "storageId",
                "sensitive",
                "restricted"
            ],
            "type": "openStudy",
            "customClass": "flex-basis-9 center"
        },
```

(continues on next page)

(continued from previous page)

```
        {
            "name": "patientId",
            "parameters": [
                "patientId"
            ],
            "type": "string",
            "labelTranslation": "common.id"
        },
        {
            "name": "patientName",
            "parameters": [
                "patientName"
            ],
            "type": "string",
            "labelTranslation": "common.name"
        },
        {
            "name": "modality",
            "parameters": [
                "modality"
            ],
            "type": "string",
            "customClass": "flex-basis-11",
            "labelTranslation": "common.modality"
        },
        {
            "name": "description",
            "parameters": [
                "description"
            ],
            "type": "string",
            "labelTranslation": "common.description"
        },
        {
            "name": "dateTime",
            "parameters": [
                "date",
                "time"
            ],
            "type": "dateTime",
            "labelTranslation": "common.dateTime"
        },
        {
            "name": "sourceAE",
            "parameters": [
                "sourceAE"
            ],
            "type": "string",
            "customClass": "flex-basis-15",
            "labelTranslation": "common.sourceAE"
        }
    ]
}
```

Values of “name” known to the GUI: `openStudyColumn` (special handling), `reportColumn` (special handling), `patientId`, `patientName`, `modality`, `description`, `dateTime`, `sourceAE`.

Values of "parameters" (field names) known to the plugins: `uid`, `storageId`, `sensitive` (certain non-public plugins only), `restricted` (certain non-public plugins only), `hasReport` (QR plugin only), `patientId`, `patientName`, `modality`, `description`, `date`, `time`, `sourceAE`, `receivedDate`, `custom1`, `custom2` ... `custom10`.

At the plugin, the corresponding fields are set up like this (. . . columns.PARAMETER_NAME.*):

```
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.patientId.tag=1048608
#...
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.sourceAE.tag=1048624
com.softneta.meddream.pacs.configurations[PLUGIN_INDEX_NUMBER].columns.sourceAE.type=string
```

"parameters" in columns.json lists fields from the plugin that will be combined into the value (multiple parameters are joined with a space character). At the plugin, . . . columns.PARAMETER_NAME.* configure what DICOM attribute to use for this field (documentation of QR and Dicomweb plugins provides more details).

5.1.22. Passing configuration via environment variables

In some installations, especially Docker-based, it is more convenient to have a fixed application.properties file where a few settings are missing, and pass those settings via environment variables instead.

For this to work, the environment variable must be named according to Spring Boot "relaxed binding" rules.

Example configuration variable: `com.softneta.meddream.pacs.configurations[0].baseUrl`.

Corresponding environment variable: `COM_SOFTNETA_MEDDREAM_PACS_CONFIGURATIONS_0_BASE_URL`.

Most plugins also support `COM_SOFTNETA_MEDDREAM_PACS_CONFIGURATIONS_0_BASEURL` etc. (without any underscores as uppercase hints) in their variables.

Warning: As of 8.9.0+, plugin settings containing dot separators, like `columns.sourceAE.tag`, can't be passed via environment variables, as implementation of the above convention doesn't fully account for dot separators: they are lost, underscores only represent uppercase hints.

Note: Due to Spring limitations, settings passed via environment must always include . . . _ID and . . . _TYPE. For example, when a single varying setting like baseUrl is missing from application.properties, the environment has to provide three variables in total.

5.1.23. Setting up Azure SSO with SAML 2.0

You will need to have the following:

- a MedDream installation
- Microsoft Azure account
- SSL set up either in MedDream or in a reverse proxy (Apache or Nginx)

5.1.23.1. Set up Azure AD SAML Toolkit

Follow instructions how to set up Azure AD SAML Toolkit: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/add-application-portal .

After creating the Toolkit, click **2. Set up single sign on** and add the following URLs in Basic SAML Configuration section:

Parameter	Value
Identifier (Entity ID)	https://MEDDREAM_URL/saml2/service-provider-metadata/a8c544d-c442-4e94-aee4-2df9cb2992aa
Reply URL (Assertion Consumer Service URL)	https://MEDDREAM_URL/saml2/login/sso/a8c544d-c442-4e94-aee4-2df9cb2992aa
Sign on URL	https://MEDDREAM_URL/saml2/authenticate/a8c544d-c442-4e94-aee4-2df9cb2992aa

Change MEDDREAM_URL to real IP address or domain.

Change a8c544d-c442-4e94-aee4-2df9cb2992aa to Application ID of the SAML Toolkit.

In the **SAML Certificates** section, copy the *App Federation Metadata Url*, for example, https://login.microsoftonline.com/a6a6746e-14bd-4e68-9c12-7593d8871c3c/federationmetadata/2007-06/federationmetadata.xml?appid=9a8c544d-c442-4e94-aee4-2df9cb2992aa. It will be required in the next step.

5.1.23.2. Set up MedDream Viewer with Azure AD

Azure doesn't support plain HTTP hyperlinks, therefore either MedDream itself must use SSL, or you can use a reverse proxy.

(Option 1) MedDream Viewer with SSL

Configure the following in application.properties for SSL:

```
server.ssl.key-store=${com.softneta.meddream.configRoot}/keystore.jks
server.ssl.key-store-password=password
server.ssl.key-alias=desktop-alias
security.require-ssl=true
server.port=443
```

. . . and for SAML:

```
spring.profiles.include=auth-saml
spring.security.saml2.relyingparty.registration.a8c544d-c442-4e94-aee4-2df9cb2992aa.
↪assertingparty.metadata-uri=https://login.microsoftonline.com/a6a6746e-14bd-4e68-9c12-
↪7593d8871c3c/federationmetadata/2007-06/federationmetadata.xml?appid=9a8c544d-c442-4e94-
↪aee4-2df9cb2992aa
spring.security.saml2.relyingparty.registration.a8c544d-c442-4e94-aee4-2df9cb2992aa.acs.
↪location={baseUrl}/saml2/login/sso/{registrationId}
```

As in previous section, change a8c544d-c442-4e94-aee4-2df9cb2992aa to real Application ID.

Change the value of . . . metadata-uri= to App Federation Metadata Url copied earlier.

Restart MedDream service after saving changes.

(Option 2) MedDream Viewer with Apache-based reverse proxy that provides SSL

Configure the following in application.properties for proper interaction with the reverse proxy:

```
server.port=8080
server.use-forward-headers=true
server.forward-headers-strategy=native
server.tomcat.remote-ip-header=x-your-remote-ip-header
server.tomcat.protocol-header=x-your-protocol-header
```

. . . and for SAML:

```
spring.profiles.include=auth-saml
spring.security.saml2.relyingparty.registration.a8c544d-c442-4e94-aee4-2df9cb2992aa.
↪assertingparty.metadata-uri=https://login.microsoftonline.com/a6a6746e-14bd-4e68-9c12-
↪7593d8871c3c/federationmetadata/2007-06/federationmetadata.xml?appid=9a8c544d-c442-4e94-
↪aee4-2df9cb2992aa
spring.security.saml2.relyingparty.registration.a8c544d-c442-4e94-aee4-2df9cb2992aa.acs.
↪location={baseUrl}/saml2/login/sso/{registrationId}
```

As in previous section, change a8c544d-c442-4e94-aee4-2df9cb2992aa to real Application ID.

Change the value of . . . metadata-uri= to App Federation Metadata Url copied earlier.

Change MEDDREAM_URL to real IP address or domain.

Restart MedDream service after saving changes.

Add the following to Apache's httpd.conf:

```
Listen 443
<VirtualHost *:443>
    ServerName MEDDREAM_URL
    ServerAlias MEDDREAM_URL
    ServerAdmin MEDDREAM_URL

    SSLEngine on
    SSLProxyEngine on
    SSLCertificateFile      "${SRVROOT}/conf/ssl/cert.crt"
    SSLCertificateKeyFile   "${SRVROOT}/conf/ssl/cert.key"

    #MedDream
    ProxyPreserveHost On
    RequestHeader add X-Forwarded-Proto https
    ProxyTimeout 800
    ProxyPass / http://127.0.0.1:8080/
    ProxyPassReverse / http://127.0.0.1:8080/
    ErrorLog "|bin/rotatelogs.exe -l logs/meddream.error.%Y.%m.%d.log 86400"
    CustomLog "|bin/rotatelogs.exe -l logs/meddream.access.%Y.%m.%d.log 86400" common
</VirtualHost>
```

The most important ones are ProxyPreserveHost and RequestHeader.

Restart the Apache service after saving changes.

5.1.23.3. Test login from Azure AD to MedDream Viewer

Log in to https://portal.azure.com with test user account which you have created when you set up Azure AD SAML Toolkit, then go to Active Directory > Enterprise Applications. Choose “Azure AD SAML Toolkit” created earlier. In Manage section choose Single sign-on.

Scroll down till you see section **5 Test single sign-on with Azure AD SAML Toolkit** and press the Test button.

Select *Sign in as current user* and click *Test sign-in* button. It will open a new tab for sending login information to Microsoft. Enter test user email address and password.

After a successful login, the page will change to MedDream Viewer search window.

5.1.24. Setting up Keycloak SSO with SAML 2.0

(This is a summary that assumes the reader is already familiar with Keycloak. Please contact support@meddream.com for the detailed version with screenshots.)

1. In Keycloak, create a Realm, a Client, at least one test user and export the certificate of that client.
2. In order to map MedDream permissions to groups, the latter need to be returned in the token as a “groups” object (other popular names like “memberOf” won’t work):

```
{
    "exp": 1735152069,
    ...
    "groups": [
      "meddream-group"
    ],
    ...
    "preferred_username": "ag"
}
```

3. Copy the exported certificate (files .crt and .key) to MedDream Viewer installation directory, for example, C:\MDPACS\MedDream.
4. Add the following to application.properties

```
spring.profiles.include=auth-saml
authorization.defaultLoginPermissions=SEARCH,DOCUMENT_VIEW

spring.security.saml2.relyingparty.registration.REALM_NAME.signing.
↪credentials[0].private-key-location=file:///C:/MDPACS/MedDream/myalias.key
spring.security.saml2.relyingparty.registration.REALM_NAME.signing.
↪credentials[0].certificate-location=file:///C:/MDPACS/MedDream/myalias.crt
spring.security.saml2.relyingparty.registration.REALM_NAME.assertingparty.
↪metadata-uri=http://KEYCLOAK_ADDR_PORT/realms/meddream/protocol/saml/
↪descriptor
```

   and update REALM_NAME to the actual realm name, KEYCLOAK_ADDR_PORT – to the address (and possibly port) of Keycloak server. Then adjust paths to .key and .crt files.
5. Save the changes and restart the MedDream service.
6. Open MedDream root URL in the web browser (for example, http://127.0.0.1:8082). It will be automatically redirected to Keycloak login page.
7. Enter the username and password created in Keycloak during the first step.
8. If credentials and configuration are correct, the Search window of MedDream will open.

5.1.25. Avoiding service restart after some configuration changes

8.3.1+ supports a "configuration server": a cloud-dedicated mechanism where a separate JAR is running and provides a fresh copy of application.properties for multiple MedDream instances. Currently you can run it on the same machine for a single MedDream instance.

1. Create a dedicated subdirectory (for example, "C:\MDPACK\MedDream\configserver"). Move the file md-config-server-0.0.1-SNAPSHOT.jar to that directory.
2. Create a couple of deeper subdirectories `config/md` ("C:\MDPACK\MedDream\configserver\config\md") and an empty file **md.properties** (not application.properties) there.
3. Copy storage definitions (PACS plugin configuration) from MedDream to this file. Another supported setting is com.softneta.dicom-store-service.acceptAETitles. **Other settings were not tested so far, and might result in undefined behavior, or not apply as expected.** For example,

```
com.softneta.dicom-store-service.acceptAETitles=PACS4

com.softneta.meddream.pacs.configurations[0].type=Dcm4chee2
com.softneta.meddream.pacs.configurations[0].id=d4c2Local
com.softneta.meddream.pacs.configurations[0].url=\
jdbc:mariadb://172.16.32.11:3306/pacsdb
com.softneta.meddream.pacs.configurations[0].username=pacs
com.softneta.meddream.pacs.configurations[0].password=pacs
com.softneta.meddream.pacs.configurations[0].defaultStoragePath=\
C:/PACS/dcm4chee/server/default/archive

com.softneta.meddream.pacs.configurations[1].type=FileSystem
com.softneta.meddream.pacs.configurations[1].id=files1
com.softneta.meddream.pacs.configurations[1].rootDirectory=C:\\upload
```

4. From the dedicated directory, start the JAR: `java -Dspring.profiles.active=native -jar C:\MDPACS\MedDream\md-config-server-0.0.1-SNAPSHOT.jar` (include the full path to JRE executable if needed).
5. At MedDream side, add the following to application.properties (or change existing lines), then restart the MedDream service:

```
authentication.manager.username = admin
authentication.manager.password = <USE A STRONG PASSWORD>
management.endpoints.enabled-by-default = true
management.endpoints.web.exposure.include = refresh
```

   This is similar to other parts of this Manual which also suggest configuring manager's credentials and exposing some management endpoints. For this scenario a strong password and the endpoint name "refresh" are sufficient.

6. After adding more plugin configurations at the configuration server (`C:\MDPACK\MedDream\configserver\config\md\md.properties`), or changing the acceptAETitles setting there, make a POST call to <MedDream URL>/manage/refresh, with the configured credentials passed via Basic Authentication.

   The configuration will refresh immediately. After reloading the GUI in browser, or logging in anew, you'll see the updated set of storages.

Current limitations:

- MedDream requires at least one plugin in order to start, therefore you can't begin with no plugin configurations in its application.properties. Include at least one initially (at both sides), then add more at the configuration server.
- Plugin configurations present in MedDream's application.properties can't be removed via the configuration server. Furthermore, their copies should always remain in md.properties to avoid unexpected and hard to troubleshoot behavior.

For example, a particular plugin was defined as the second one (`...[1]`) and you remove the first one. Then this new [0] coming from configuration server will be merged with the original [0] at MedDream, which means overwriting properties with the same names.

The configuration server starts on port 8888, and MedDream expects it there. If this port can't be used on your machine, then more configuration is necessary. Suppose you chose the port 12345:

- create a dedicated application.properties file in the configuration server's directory (`C:\MDPACK\MedDream\configserver\application.properties`) and add a single server.port setting:

```
server.port=12345
```

- in MedDream's application.properties, specify the port using a different syntax:

```
spring.config.import=configserver:http://127.0.0.1:12345
```

5.1.26. Changing the log level without a restart

Similar to the previous chapter, the "loggers" endpoint under /manage supports changing some system-global settings, too. **The /manage endpoint must be enabled in advance.**

Minimum configuration (please change the password to a stronger one):

```
management.endpoints.web.exposure.include=loggers
management.endpoints.enabled-by-default=true
authentication.manager.username=admin
authentication.manager.password=admin
```

It allows you to open http://MEDDREAM_URL/manage/loggers in the browser (after entering the credentials when asked) and see the available loggers. The response format will look like this:

```
{
  "levels": [
    "OFF",
    "ERROR",
    "WARN",
    "INFO",
    "DEBUG",
    "TRACE"
  ],
  "loggers": {
    "ROOT": {
      "configuredLevel": "INFO",
      "effectiveLevel": "INFO"
    },
    "com.softneta": {
      "configuredLevel": "INFO",
      "effectiveLevel": "INFO"
    },
```

You can access a particular logger directly, too. For example, http://MEDDREAM_URL/manage/loggers/com.softneta.

With a more restricted configuration like management.server.address=127.0.0.1, even this step might need to be done directly at the MedDream machine where tools like cURL suit better.

Changing the logging level for a particular package involves a POST request so you'll need a more specialized tool. Example under Linux for the "com.softneta" logger (logging.level.com.softneta):

```
curl -u admin:admin -X POST -H "Content-Type: application/json" -d \
    "{\"configuredLevel\": \"DEBUG\"}" http://MEDDREAM_URL/manage/loggers/com.softneta
```

Now you can examine /manage/loggers once more and confirm that the value for a particular logger has changed.

Hint: under Windows, due to the way its shell handles double quotes, it might be more feasible to keep the JSON data in a file and reference it as `-d @FILENAME`.

Hint: Ostara is a desktop application that allows changing the levels (and much more) in a more convenient way.

5.2. Running as a service

Troubleshooting incomplete configuration is more difficult when a service is involved. You should first make sure that MedDream already works as expected when Java Core is started from a console window.

5.2.1. Windows

1. Open an elevated Command Prompt;
2. Run the command `MedDream.exe install`;
3. Start the "MedDream" service from the Windows service manager.

When configuring the Token Service this way, a different directory is required. It's because both Java applications attempt to read the same application.properties file, however the Token Service needs either a separate file or none at all.

In the MedDream installation archive, all files related to Token Service already are in a separate subdirectory. We suggest moving the entire subdirectory (not just files in it) to a suitable place.

1. Open an elevated Command Prompt there;
2. Run the command `token-service.exe install`;
3. Start the "MedDream Token Service" from the Windows service manager.

Note: Starting from MedDream 8.9.0, only a single version of MedDream.exe and token-service.exe is provided, as most installations did not use the files for .NET v2.

In case yours still needs the older one after the upgrade, you can keep using your existing file: the copy bundled with MedDream hasn't been updated for a long time anyway.

5.2.2. Linux (System-V init)

If your system is RedHat-like (Fedora, RHEL, etc.), then copy {INSTALL_DIRECTORY}/meddream.redhat to {INSTALL_DIRECTORY}/meddream. If it's Debian-like (Ubuntu, Debian itself, etc.), then copy {INSTALL_DIRECTORY}/meddream.debian to {INSTALL_DIRECTORY}/meddream.

Execute the following commands:

```
sudo ln -s {INSTALL_DIRECTORY}/meddream /etc/init.d/meddream
sudo chmod +x {INSTALL_DIRECTORY}/meddream {INSTALL_DIRECTORY}/meddream-jar-wrapper
sudo service meddream start
sudo chkconfig meddream on
```

Note: The variable BASEDIR in the script is dynamically initialized with the directory of the script itself. This means that you must leave the script near the JAR file, and place only a softlink in /etc/init.d, like in the above example. The entire script can be copied to /etc/init.d only if you adjust BASEDIR, too.

As Debian-like systems don't have chkconfig, the last command can be replaced with `sudo update-rc.d meddream defaults`.

By changing the line `USER=root` in {INSTALL_DIRECTORY}/meddream, you can force a less privileged user that can still access DICOM files etc. However this will likely require updating the line `PIDF=/var/run/$PROG.pid`, too, as /var/run is usually a privileged location (a chown'ed subdirectory will suffice in that case).

The file {INSTALL_DIRECTORY}/meddream also contains the JRE command line, for example:

```
MD_JAR_COMMAND="java -cp ./MedDream-8.9.1.jar -Xmx1G -Dloader.path=./sys/plugins -DANTLR_USE_
↪DIRECT_CLASS_LOADING=true org.springframework.boot.loader.launch.PropertiesLauncher"
```

To configure the Token Service this way, change the file names and command parameters accordingly:

- `/opt/token-service/token-service.redhat` or `/opt/token-service/token-service.debian` is copied to `/opt/token-service/token-service`;
- in the console,

  ```
  sudo ln -s /opt/token-service/token-service /etc/init.d/token-service
  sudo chmod +x /opt/token-service/token-service /opt/token-service/\
      token-service-jar-wrapper
  sudo service token-service start
  sudo chkconfig token-service on
  #sudo update-rc.d token-service defaults
  ```

- files `/opt/token-service/token-service` and `/opt/token-service/application.properties` will likely not need any changes.

5.2.3. Linux (systemd)

Create the file /etc/systemd/system/meddream.service with the following text:

```
[Unit]
Description=MedDream Viewer
After=syslog.target

[Service]
WorkingDirectory=/opt/meddream
User=root
ExecStart=/usr/bin/java -cp MedDream-8.9.1.jar -Xmx1G -Dloader.path=./sys/plugins -DANTLR_
↪USE_DIRECT_CLASS_LOADING=true --add-opens=jdk.management/com.sun.management.internal=ALL-
↪UNNAMED --add-opens=java.base/jdk.internal.misc=ALL-UNNAMED --add-opens=java.base/sun.nio.
↪ch=ALL-UNNAMED --add-opens=java.management/com.sun.jmx.mbeanserver=ALL-UNNAMED --add-
↪opens=java.base/sun.reflect.generics.reflectiveObjects=ALL-UNNAMED --add-opens=java.base/
↪java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.
↪util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.
↪time=ALL-UNNAMED --add-exports=java.desktop/sun.awt.image=ALL-UNNAMED org.springframework.
↪boot.loader.launch.PropertiesLauncher
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

Note: Update `User=root` with the actual username under which the Java application will run.

Execute the following command for service autostart:

```
sudo systemctl enable meddream.service
```

To configure the Token Service this way, change the names accordingly: the file is /etc/systemd/system/token-service.service, and contains

```
[Unit]
Description=MedDream Token Service
After=syslog.target

[Service]
WorkingDirectory=/opt/token-service
User=root
ExecStart=/usr/bin/java -Xmx1G -Dspring.profiles.active=ip-whitelist-authentication -jar /
↪opt/token-service/token-service.jar
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

The service autostart of course needs

```
sudo systemctl enable token-service.service
```

5.2.4. Linux (upstart)

Create a file /home/{user name}/.config/upstart/meddream.conf with the following text:

```
description "MedDream Viewer"
respawn
exec java -cp /opt/meddream/MedDream-8.9.1.jar -Xmx1G -Dloader.path=/opt/meddream/sys/
↪plugins -DANTLR_USE_DIRECT_CLASS_LOADING=true --add-opens=jdk.management/com.sun.
↪management.internal=ALL-UNNAMED --add-opens=java.base/jdk.internal.misc=ALL-UNNAMED --add-
↪opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/com.sun.jmx.
↪mbeanserver=ALL-UNNAMED --add-opens=java.base/sun.reflect.generics.reflectiveObjects=ALL-
↪UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-
↪UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-
↪UNNAMED --add-opens=java.base/java.time=ALL-UNNAMED --add-exports=java.desktop/sun.awt.
↪image=ALL-UNNAMED org.springframework.boot.loader.launch.PropertiesLauncher
```

To configure the Token Service this way, change the names accordingly: the file is /home/{user name}/.config/upstart/token-service.conf and contains

```
description "MedDream Token Service"
respawn
exec java -Xmx1G -Dspring.profiles.active=ip-whitelist-authentication -jar /opt/token-
↪service/token-service.jar
```

5.3. JRE or JDK?

A JRE (Java Runtime Environment) is the minimum sufficient for most installations.

If your case requires spring.profiles.include=ignite, then the related Apache Ignite library requires a JDK. Furthermore, the JVM command line then requires the option `--add-opens=jdk.internal.jvmstat/sun.jvmstat.monitor=ALL-UNNAMED`; do not forget to *update it where appropriate* (MedDream.xml under Windows, meddream.debian/meddream.redhat under Linux with System-V init, etc).

5.4. Verification checklist

Suppose you already went through the 5.1.1 Essential configuration and the first run chapter, configured the authentication method and PACS plugins, set up Java applications as services, adjusted system.json and even read the 5.1.13 Reference for remaining configuration parameters superficially.

This is a good opportunity for initial security tightening (see 7 Security considerations, especially the "Summary" part). If during that process you accidentally disable too much of MedDream functionality, this will be noticed in subsequent steps.

Afterwards it's time for a final functionality check-up.

- For installations with an interactive login and Search window:
  - The login form does not produce an error message and is successfully replaced by the Search window. An administrative login (with the ADMIN permission) also succeeds and gives access to the Settings window.

    Remember that the login form is unavailable by default (results in "403 Forbidden" message) if no authentication method is configured.

    In 7.5.1+, there is no sense to have an interactive login if the Search window is disabled (the text "404" on dark background, due to features.search=false in system.json).

    Even if the integration doesn't offer a working search functionality, you might still decide to keep the window enabled, at least for a short time – for convenient access to the Settings window via an administrative login.
  - The Search window shows studies from every configured PACS plugin (except FileSystem).
    * Results can be restricted by date interval, Patient ID, etc. Keep in mind, however, that some plugins do not support returning some fields at all, and some plugins do not support filtering by some returned fields.
    * If the plugin supports filtering by user identity, then results are different depending on who is logged in. Consult configuration of the PACS and MedDream for a suitable identity.
  - If Export or Forward are enabled (both globally and by current user's permissions), then the operation can be started from the Search window and succeeds.
- For all installations:
  - Opening a study succeeds: thumbnails are displayed instead of exclamation marks and the chosen image opens as well.

    Choose the studies wisely and cover at least different modalities.

    * Attempt to open MPEG2/MPEG4 video studies if any exist.
    * For example, if the PACS stores all images from a particular Source AE, or some historic images, at a totally different location, then you might not notice a faulty or incomplete "mappedStorageLocation" plugin setting until those images are accessed.

    When opening from the Search window, repeat the attempt for every PACS plugin that returned some search results.

  - From an open study, all functionality allowed by 1) configuration, 2) license and 3) current user's permissions, is accessible and succeeds:
    * Export (both its versions, "Burn" and "Export", also including the reports as PDFs),
    * Forward,
    * Share to DICOM Library,
    * Oblique MPR,
    * display of PDFs and SRs,
    * the expected set of measurements is available.

    Repeat with less-privileged users if any exist.
  - Verify that the example "demo" / "demo" user account is not present any more.
  - The Core service (and the token validation service, if used) starts successfully after rebooting the entire machine.
  - If rebranding is used, then the About window etc. looks as expected.
  - *. . . side information* settings under *Info Label properties* correspond to end users' requirements. Changing these settings often is not desired: it will also require cleaning of temporary files at the server, therefore studies will again open slower like during the first time.
- For HIS integration installations:
  - In case of "unsafe" integration, check if every enabled URL parameter ("study" etc.) results in opening of the right studies. If the "storage" parameter is not listed in the validHisParams setting, it will be silently ignored.
  - In case of token-based integration, also check whether MedDream correctly handles all integration options implemented in your HIS: expected set of object identifiers ("study", "accnum", etc. – no less and no more), token-specified patient history, permission overrides, object restrictions (like by Patient ID), one-time tokens, limited-lifetime tokens. This will obviously involve different tokens and experiments at the HIS side.
  - If MedDream is supposed to open in an IFRAME at your webpage, then also attempt to open it from the dedicated network and using the dedicated server hostname.
  - If the Token Service is accessible from the Internet (not recommended) for whitelisted client IPs, then verify if the whitelisting behaves as expected for "good" and "bad" IPs. Do not forget the special case when the requests are coming through a proxy.
- For integration into PacsOne web interface:
  - MedDream successfully opens from the list of studies (not from the list of patients, series or images).
  - Multiple studies are allowed. You should know how many of them the users might want, and configure the limit accordingly.
  - After MedDream opens this way, the expected set of features (Export, Reporting, etc.) is still available to all users.

A few first days after going online can involve multiple restarts of the Core service. Each restart can leave a lot of unused temporary files that will be removed some time later. The cleanest way to verify this removal is to lower *every* "olderThanSec" setting (for example, to 1 second) temporarily and confirm that all accumulated files disappear. Nevertheless, monitoring of the disk space is still advised.

5.5. Option: dockerized environment

We provide two ready-to-use Docker images that might save you some time, especially during evaluation. See

- https://hub.docker.com/r/meddream/dcm4chee-dicom-viewer for DCM4CHEE 5.x, or
- https://hub.docker.com/r/meddream/orthanc-dicom-viewer for Orthanc.

5.6. Updating the license

After purchasing a commercial license or more concurrent connections, it is necessary to update the license file (meddream.lic).

On installations with an Internet connection, the "Register" window is sufficient. MedDream will automatically download the file and place it where appropriate. The new file comes into effect immediately, though restarting the service is still recommended.

In other cases, you must update meddream.lic manually:

- by default it resides in the same directory as application.properties or meddream*.jar, likely it's `/opt/meddream` or `C:\MedDreamPACS-Premium\MedDream`;
- in dockerized installations a dedicated subdirectory is often used, like `/opt/meddream/license` – that way it's easier to store any changes outside the container;
- when in doubt, look for com.softneta.license.licenseFileLocation in application.properties. If this setting is not present, it defaults to the directory of the JAR file.

Restarting the service is mandatory after a manual update.

5.7. Cleaning server-side caches

MedDream 8.1+ has the "Clear cache" function for interactive removal of a single study (or series, or image) from the preparation cache. It must be enabled globally in system.json (features.clearCache=true) and for a particular user (CLEAR_CACHE permission).

When "Clear cache" can not be used due to some reason, the entire preparation cache will need cleaning if:

- you have changed the setting com.softneta.preparation.compressPixelsBeforeSave;

  Currently the setting controls whether the pixels are cached in compressed or non-compressed form, and the frontend attempts decompression based on its value (instead of detecting the existing format). As a result, cached pixels become incompatible after changing the setting.

- you have changed the *... side information* settings under *Info Label properties* (the latter, in turn, might be affected by `personNameConfiguration` in system.json);

  Info Labels are generated and cached using the configured patterns. This includes a particular person name format configured elsewhere. Changed patterns will be visible only on studies that haven't been opened before.

  Plan ahead when choosing the values, as losing many days' worth of prepared files might be too costly.

- `trackDicomModifications` in application.properties is off (the default value) and some DICOM files have been changed on the PACS.

  For example, PacsOne supports modification of certain DICOM attributes and this is done not only in the database but in the files, too. The operation updates the file timestamps, therefore trackDicomModifications=true together with a *direct-integration plugin* is enough for automatic regeneration.

Modalities are an unlikely reason, as they normally use new UIDs after changing patient demographics etc. in a study that has already been sent to the PACS.

When a new version introduces minor changes in image processing, then cleaning the preparation cache is recommended, too.

- Example 1: v8.4 fixes a bug where a particular color image had distorted colors due to inadequate YCbCr / RGB conversion. Because the old version has cached the result of this inadequate conversion, the new version will still display the same distorted colors unless the cache is cleared at least for this study ("Clear cache" is sufficient).
- Example 2: v8.4 allows interactive control of DICOM overlays present in the image. However, the old version has cached a version of the image with them burned in permanently. Until the cache is cleaned at least for a particular study, the "Show DICOM Overlay" menu button will not be able to turn the overlays off.

The default location of the preparation cache is "temp/cache" in the directory of the JAR file. Your installation might redefine it directly via com.softneta.preparation.cacheDir; if only com.softneta.meddream.tempDir is redefined, then a subdirectory "cache" under it is used by default. To confirm that it's the right one, examine the contents: you should see a storage subdirectory (same as .id in configuration of a PACS plugin), then three two-digit subdirectories like `6f/3e/1b`, then a long subdirectory like `89a44027269f77b1cb3948290635077461` and various files below it.

After removing the cached files, the MedDream service must be restarted in some cases.

5.8. Creating a dump of an H2 database

1. Open H2 Console in the browser, then a corresponding database there (see 5.1.14 Opening the H2 Console for details).
2. Execute the command `SCRIPT TO 'db-backup.sql'`, where "db-backup.sql" can have any name and extension (an SQL file is always created), and wait for its completion.

   By default, the file is created in the same directory as the application JAR file; you should be able to use any other writeable location, too, like `/tmp/export.sql` or `d:\temp\backup.sql`.
3. Disable the Console afterwards.

5.9. Restoring an H2 database from a dump

Note: The restore process will destroy current database contents. If in doubt, make a backup.

A binary backup is sufficient. A rather simple way is to copy the database file (it has an extension .mv.db) **when the MedDream service is stopped**.

1. Open H2 Console in the browser, then a corresponding database there (see 5.1.14 Opening the H2 Console for details).
2. Remove all content by using the command `DROP ALL OBJECTS`.
3. Execute the command `RUNSCRIPT FROM 'db-backup.sql'`, using the relevant path to a dump file. Make sure there are no errors reported.
4. Restart the application so that it has the opportunity to upgrade the database schema, then verify the functionality related to this database.
5. Disable the Console afterwards.

5.10. File-based H2 storage and non-default configRoot under Windows

By default MedDream uses H2 databases for Reporting, persistent jobs and related Quartz-based scheduling. The corresponding files are in the directory sys/database because of these default settings:

```
report.datasource.url = jdbc:h2:file:${com.softneta.meddream.configRoot}/sys/database/
↪reportdb;USER=sa
datasource.batch.url = jdbc:h2:file:${com.softneta.meddream.configRoot}/sys/database/batchdb;
↪USER=sa
datasource.quartz.url = jdbc:h2:file:${com.softneta.meddream.configRoot}/sys/database/
↪quartzdb;USER=sa
```

Since approx. MedDream 7.9.0, in this configuration H2 crashes under Windows if configRoot contains the "file:///" prefix. This is because officially the database supports only absolute Windows paths without the prefix, like jdbc:h2:file:C:/db/mdreports.

Consequently, if you must use `file:///C:/...` instead of just `C:/...` in configRoot due to some reason, and some database is still H2 instead of MySQL/MariaDB or PostgreSQL, then the corresponding JDBC URL must be explicitly added to application.properties without the "file:///" prefix, like `jdbc:h2:file:C:/db/mdreports;USER=sa`.

5.11. Using strong passwords in MySQL/MariaDB accounts

Passwords with special characters, like `NN"A_B7;Q{G\.}q5+XTuw=^q`, can be difficult to enter correctly in SQL queries ("IDENTIFIED BY" clause, "SELECT PASSWORD(...)" query, etc.) because some characters must be escaped or else their meaning becomes different. As a result, the password might not be accepted afterwards even by the same MySQL client.

A workaround is to use an online MySQL password calculator/generator *that is entirely browser-based* (does not perform a request to the server during the calculation), then enter the resulting hash into the query that updates the password.

At the time of this writing (January 2023), we can recommend https://www.browserling.com/tools/mysql-password; it yields `*6BED9D9BC91C5F5994B2F7D4AE105E2E5BBC262F` for the example string above.

5.12. Validation of MedDream.jar signatures

Since 8.6.0 MedDream*.jar is a signed file. The integrity is verified automatically, for example, when accessing the /manage/health endpoint. Any detected problems are also logged at ERROR level together with a message "Consistency check".

If there is a need to verify checksums of internal resources manually in "offline" mode, the following command can be used (`jarsigner` is part of the JDK):

```
jarsigner -verify -verbose:summary MedDream-8.9.1.jar
```

5.13. Adding/updating a third-party library in MedDream.jar

1. Make a backup of MedDream-*.jar. (For simplicity we'll call the working copy just "MedDream.jar" from now on.)
2. In the same directory, create a subdirectory BOOT-INF/lib and copy the libraries there. Example: `BOOT-INF/lib/iso9660-writer-2.1.0.jar`.
3. In the same directory, run the `jar` utility from the JDK to add/update everything below this BOOT-INF subdirectory:

```
jar -uf0 MedDream.jar BOOT-INF
```

It might be possible to apply a security hotfix this way, too. However,

- not every version of a library will be compatible enough so that a drop-in replacement is possible. In theory, a different "X" in x.y.z might need a serious amount of adaptation work and a new build of MedDream. In practice, even a different "Z" might result in surprises if too big changes have been versioned incorrectly;
- do not keep different versions at the same time, like "iso9660-writer-2.1.0.jar" together with "iso9660-writer-2.1.1.jar". Unless reusing the old file name is OK to you, the old one needs to be removed from the JAR first. The widespread "zip" utility is sufficient for that. Example for the same library:

```
zip -d MedDream.jar BOOT-INF/lib/iso9660-writer-2.1.0.jar
```

Note, however, that a modification of the JAR file invalidates its signature. If your infrastructure (like docker containers on AWS) utilizes the /manage/health endpoint to check whether the backend is running, then the status "DOWN" from a modified JAR will prevent starting it.

5.14. Backing up MedDream

MedDream doesn't include any backup functionality on its own. The customer is responsible for backing up MedDream and for any losses caused by missing or inadequate backups.

5.14.1. Locations and files

Sometimes the installation has an infrastructure that allows backing up an entire MedDream Docker container or an entire virtual machine. This simplifies things but might still not cover everything:

- in containers, some locations are often bindmounted to locations on the host for easier access to configuration or to keep any large caches outside;
- in separate machines, some large caches might still be external (on a network storage).

The further chapters focus on traditional installations where maintenance is performed from the inside of the machine and backups must be configured at directory or file level.

5.14.1.1. The Viewer itself

You can basically back up the entire MedDream installation directory. A few exceptions should be made inside it:

- exclude the subdirectory `temp`

  By default it's referenced by com.softneta.meddream.tempDir = ${com.softneta.meddream.configRoot}/temp. The temporary files are naturally transient, of little value, and often just waste the disk space. However, if you have configured this option with a directory outside MedDream installation directory (or on a different partition/disk altogether), then the stock "temp" remains unused and doesn't need an exception.

By default this directory contains a subdirectory “converted-videos” due to com.softneta.video.convertedDir = ${com.softneta.meddream.tempDir}/converted-videos. It’s usually empty but might contain large broken video files from a failed conversion. There’s likely no need to back up even your custom com.softneta.video.convertedDir.

By default this directory contains a subdirectory “upload-to-dicom-library” due to com.softneta.dicom.library.upload.tempDir = ${com.softneta.meddream.tempDir}/upload-to-dicom-library. When using the “Share” functionality on very large studies where the upload takes hours, there is a chance that the corresponding database job is still being executed during the backup; if you managed to back up the database (batchdb + quartzdb, see further in this chapter) **and** this subdirectory, then the upload will continue after everything is restored and Core is started. It is unlikely that you want that, therefore even a custom com.softneta.dicom.library.upload.tempDir does normally not need backing up.

By default this directory contains a subdirectory “cache” due to com.softneta.preparation.cacheDir = ${com.softneta.meddream.tempDir}/cache. The preparation cache is often infeasible to back up due to very large amount of data. If you can afford it, the “cache” subdirectory or your custom com.softneta.preparation.cacheDir can be separately mirrored to a dedicated backup storage; keep in mind, though, that the tree doesn’t contain any date-based subdirectories and must be scanned entirely every time.

By default this directory contains a subdirectory “STORE” due to com.softneta.dicomStoreService.saveDirectory = ${com.softneta.meddream.tempDir}/STORE. DICOM files received by the local Storage SCP in a Query/Retrieve integration might occupy a lot of space just for faster operation in some scenarios. Even a custom com.softneta.dicomStoreService.saveDirectory is normally not worth backing up.

This directory might also contain a subdirectory “dcm” due to suggestions in sample application.properties like com.softneta.meddream.pacs.configurations[0].dicomCacheDirectory = ${com.softneta.meddream.tempDir}/dcm/PLUGIN_ID. Its purpose is similar to that of the “STORE” subdirectory, and normally not worth backing up. You should also review your custom values for corresponding plugins, namely, . . . [*].dicomCacheDirectory and . . . [*].cacheDir.

- consider skipping any open H2 database files

  If some database is still of H2 kind, then this will be visible in datasource.quartz.url, datasource.batch.url or report.datasource.url. By default these files are sys/database/*.mv.db.

  An H2 file is locked even for reading under Windows and would require a shadow copy; under Linux this is not enforced but integrity of an open file still can’t be guaranteed. We recommend to copy any H2 files as a separate step, using traditional means that fail due to locking – **and when Core is temporarily shut down**.

  The *dump-based scenario* does not require a Core shutdown but is interactive-only – there is no known way to automate it.

  If some of those 3 settings refer to a traditional database like PostgreSQL, then you can choose from a lot of industry-standard database backup tools. *In production, you should use a traditional database at least for Reporting*; the difficulties while backing up H2 databases are an additional source of motivation.

- if some of the following options are configured and point outside MedDream installation directory, then corresponding locations are to be backed up separately:
  - com.softneta.license.licenseFileLocation
  - com.softneta.settings.fileLocation
  - com.softneta.statistics.fileLocation

- consider excluding the log files

  By default they’re named meddream.main.log* and meddream.preparation.log* (logrotated files will have extensions like . . . log.1 and so on) due to logging.file.name = meddream. Please review names of existing files if this option is configured in your installation.

If your configuration refers to the SSL keystore/truststore files outside this directory, then do not forget to back them up, too.

Afterwards review your choice for implementing a system service, as detailed in 5.2 Running as a service. Under Linux with systemd, the corresponding unit “meddream.service” contains the established JVM command line, is stored outside MedDream installation tree and therefore should be backed up separately.

5.14.1.2. The Token Service (if used)

This Manual suggests using a separate installation location outside the MedDream directory tree, therefore usually the Token Service needs a separate backup.

You can back up the entire installation directory. In an unlikely event when the log files tend to take too much space, they can be excluded (the default name is “token_service.log”).

If your configuration refers to the SSL keystore/truststore files outside this directory, then do not forget to back them up, too.

Finally, a system-dependent service implementation, like a systemd unit under Linux, might also need backing up.

5.14.1.3. The config-server service (if used)

As this Manual suggests a subdirectory “configserver”, it would be backed up together with MedDream above. If you didn’t go that way, then back up your entire custom location.

Review the system-dependent service implementation and add a corresponding file/directory location if needed.

5.14.1.4. Any remaining solutions (if used)

Although these are niche cases, you should also back up configuration of Redis, Ignite or similar services sometimes needed by MedDream.

5.14.2. Backup frequency

The backup strategy is a generic topic and mainly depends on institution’s security policy.

Regarding MedDream, the files that are modified at least during the installation,

- application.properties
- sys/settings/*.json
- sys/rebranding/*
- sys/reports/*
- sys/ffmpeg/*/flv.sh
- MedDream.xml, meddream.debian and other system service files
- md5/css/md.default.*.css if you use some custom GUI colors

– occupy a lot less space and could therefore be backed up more often.

5.15. Cleaning preparation cache via persistent index (experimental)

8.7.0 introduces an experimental possibility to index the preparation cache in a database. This is supposed to result in an attractive alternative to the temporary files cleaner: the latter must blindly scan the entire directory tree, while a database can quickly select the outdated entries.

Note: The current functionality is not quite ready for general public; this chapter is mainly dedicated to MedDream's own support staff.

Architectural key points:

- A dedicated database is initialized automatically. MariaDB/MySQL and PostgreSQL are supported.
- Core creates a new record when preparing a DICOM file, and updates this record when returning the structure of the corresponding study. This allows extending the lifetime of studies that are accessed more often; the traditional MedDream policy, though, is to remove files by their age.
- A separate cleaner service needs to be run in parallel to Core. It periodically reads from the index database and removes any matching records.

On the very first use the cleaner service should be run first, as it initializes the database.

1. Move preparation-db-*.jar to a dedicated subdirectory – for example, "preparation-db". This is needed because this JAR expects its own application.properties file with different contents.
2. Create a new file application.properties in that directory. Example for PostgreSQL:

```
spring.datasource.cacheDb.url = jdbc:postgresql://127.0.0.1:5432/md_cache?
↪user=postgres&password=postgres
spring.datasource.cacheDb.driverClassName = org.postgresql.Driver
com.softneta.meddream.dbCleanerProperties.cleanTasks[0].cronExpression = 0 1 *␣
↪* * *
com.softneta.meddream.dbCleanerProperties.cleanTasks[0].sqlSelect = select f_id␣
↪as hid from t_cacheh where f_access_time < now() - interval '1 week' and f_
↪status = 1 fetch first 100 rows only
#com.softneta.meddream.dbCleanerProperties.internalCacheDuration = 1h
#com.softneta.meddream.dbCleanerProperties.neverDeleteSuffixes = meta,struct,
↪workspace,search
```

Example for MariaDB:

```
spring.datasource.cacheDb.url = jdbc:mariadb://127.0.0.1:3306/md_cache?user=MD_
↪CACHE_USER&password=MD_CACHE_PASS
spring.datasource.cacheDb.driverClassName = org.mariadb.jdbc.Driver
com.softneta.meddream.dbCleanerProperties.cleanTasks[0].cronExpression = 0 1 *␣
↪* * * *
com.softneta.meddream.dbCleanerProperties.cleanTasks[0].sqlSelect = select f_id␣
↪as hid from t_cacheh where f_access_time < now() - interval '1 week` and f_
↪status = 1
#com.softneta.meddream.dbCleanerProperties.internalCacheDuration = 1h
#com.softneta.meddream.dbCleanerProperties.neverDeleteSuffixes = meta,struct,
↪workspace,search
```

`url` needs to include credentials of a sufficiently powerful database account – at least capable of creating new tables in the specified database.

`cronExpression` specifies how often the service should look for outdated records (every midnight in this example).

`sqlSelect` provides an expression for fetching the relevant records. This part will change in the future:

1. **The examples are dedicated to access-based aging**; so far no recipe for the traditional creation-based aging is provided. As a result, the experience on existing installations might significantly differ, and it will be hard to compare with the performance of the legacy temporary files cleaner or an equivalent external script.
2. The maximum age is hardcoded to 1 week.

`internalCacheDuration` (experimental) is "1h" by default and controls how long records of "t_cacheh" will be cached in memory. Longer time means less database load and some performance improvement, but costs some memory.

`neverDeleteSuffixes` (empty by default) prevents contents of these partitions from removal; at the moment only `search`, `meta`, `struct`, `workspace` are allowed and affected files will be kept permanently instead of applying a different threshold. However, removal works at directory level and implies *the recommended partitioning*; a different kind of partitioning can result in too many entries removed despite neverDeleteSuffixes, or some entries unexpectedly remaining.

3. Start this JAR in its own console window, which should be enough for a short test. Example for PostgreSQL:

```
java -Dspring.profiles.active=standalone,postgres-cache-db -jar preparation-db-
↪0.0.1.jar
```

(For MySQL/MariaDB, change "postgres-cache-db" to "mysql-cache-db".)

If you must use distributed locking for multiple instances of preparation-db, then 8.9.0+ is compatible with the "redis-locks" profile: `-Dspring.profiles.active=standalone,postgres-cache-db,redis-locks`.

A more permanent installation involves a separate system service, like explained in 5.2 Running as a service. It will have to use the same command line.

To enable population of the database at Core side, configure an application profile "mysql-cache-db" (or "postgres-cache-db") and initialize the corresponding database connection. Example for PostgreSQL:

```
spring.profiles.include = postgres-cache-db
spring.datasource.cacheDb.url = jdbc:postgresql://127.0.0.1:5432/md_cache?user=postgres&
↪password=postgres
spring.datasource.cacheDb.driverClassName = org.postgresql.Driver
```

Example for MySQL/MariaDB:

```
spring.profiles.include = postgres-cache-db
spring.datasource.cacheDb.url = jdbc:mariadb://127.0.0.1:3306/md_cache?user=MD_CACHE_USER&
↪password=MD_CACHE_PASS
spring.datasource.cacheDb.driverClassName = org.mariadb.jdbc.Driver
```

This time `url` can use a limited database account which only selects, inserts and updates the records in the specified database.

Furthermore, one should disable the temporary files cleaner rule for com.softneta.preparation.cacheDir (for example, by scheduling it too far in the future), especially when evaluating the performance.

5.16. DICOM Print

The legacy solution works through integration with third-party DICOM printing software. Compatible with MedDream PACS, PacsOne, Orthanc, dcm4chee v2, and dcm4chee v5 PACS.

DICOM printing software can be integrated by external Links Plugin (see 5.1.12 The external links plugin for more details). Please contact support support@meddream.com for assistance.

However, 8.9.0 has introduced an experimental native solution that prints 8-bit rendered images. See `features.dicomPrint` in system.json.

5.17. High-Availability configuration

This chapter provides a reference architecture and configuration steps that ensure:

- no single point of failure;
- horizontal scalability (1..N MedDream instances);
- consistent user sessions across nodes;
- safe concurrent access to cache and lock files.

The setup consists of the following components:

- Load Balancer – Entry point for all user traffic
- MedDream Viewer Nodes (1..N) – Stateless MedDream Viewer instances
- MedDream Token Service (1..N) – Stateless MedDream Token Service instances
- Shared Fast Storage – Centralized cache storage
- Redis or Valkey – Centralized session and lock management

All MedDream Viewer and Token Service nodes are identical and can be added or removed without service interruption.

5.17.1. Load balancer configuration

Any standard L4 or L7 load balancer can be used, for example:

- NGINX / HAProxy
- F5 / Citrix ADC
- Cloud Load Balancers (AWS ALB/NLB, Azure LB, GCP LB).

Balancer requirements:

- Forward HTTP/HTTPS traffic to all MedDream nodes
- Enable health checks on MedDream service port
- Sticky sessions are **not required** (sessions are stored in Redis/Valkey)

Example Logic: Users –> Load Balancer –> MedDream 1..N.

5.17.2. MedDream Viewer nodes configuration

General principles:

- all MedDream Viewer nodes must use the same configuration;
- nodes must be stateless;
- shared temp drive should be used.

Licensing: in order to use MedDream Viewer in HA architecture, you must use a special load balancing license. Please contact out sales: info@meddream.com.

1. Open MedDream Viewer application.properties and add the following lines (these lines need to be added to all MedDream Viewer instances):

```
authentication.manager.username=random_user_name
authentication.manager.password=very_long_and_secure_password
management.server.port=8081
management.server.address=127.0.0.1

management.endpoints.enabled-by-default=true
management.endpoints.web.exposure.include=health

spring.profiles.include=redis,stateless

authentication.jwt.enabled=true
authentication.jwt.secureKey=meddream_very_secret_key_1234567

com.softneta.meddream.encryptionPassword=DiVx8G6zjXjlt2kc7IjBbmwYzlImPdRA
com.softneta.meddream.encryptionSalt=K3fPv0Czj1dL57fd
```

   Remember that these values of password, secureKey, encryptionPassword and encryptionSalt are already known to anyone reading this Manual and therefore should be changed to random/unique ones.

2. Save the application.properties file and restart MedDream Viewer service.

   After the service starts, the file redis-config.yaml will be created automatically in the Viewer's installation directory.

3. Open redis-config.yaml file and change this line to actual IP address and port of REDIS, for example:

```
singleServerConfig:
  address: "redis://192.168.1.165:6379"
```

   NOTE: If you install MedDream using dockers, you can pass Redis URL by the following environment variable: `REDIS_URL:redis://192.168.1.165:6379`.

4. Restart MedDream Viewer service one again and it will start to work with REDIS.

5. Use the following URL for the health check of the MedDream Viewer:

   https://meddreamurl:8081/manage/health/md

   or

   https://meddreamurl:8081/manage/health

Verify the final result: each node should be able to serve any user request independently.

5.17.3. Redis / Valkey configuration

Redis or Valkey is used for user session sharing, distributed locks and tokens sharing.

There are two deployment options: small environments – single Redis/Valkey instance; production HA – Redis/Valkey cluster or Sentinel.

For testing purposes, you can install REDIS on Docker with the following command:

```
docker run --name redist-test -p 6379:6379 -d redis:alpine
```

REDIS will work on all network interfaces (0.0.0.0) on port 6379.

NOTE: This example is just for testing! You want to make more secure installation and configuration of REDIS in your production environment.

NOTE: Besides Redis or Valkey distributions, you can also use Cloud services which provide Redis/Valkey.

As a result, all MedDream Viewer nodes will share user sessions and safely coordinate file and resource locks.

5.17.4. Shared fast storage configuration

Shared storage is used for Core cache files ("preparation cache") and, if needed, plugin cache files.

Requirements:

- high-performance storage (NAS, SAN, or distributed filesystem);
- low latency access from all MedDream nodes;
- read/write access for MedDream service user.

Configuration:

1. Provision the shared storage.
2. Mount it on all MedDream nodes at the same path. Example: `/mnt/meddream/cache`.
3. Configure all MedDream nodes to use this path as cache directory:

```
com.softneta.meddream.tempDir=file:///mnt/meddream/cache
```

5.17.5. MedDream Token Service nodes configuration

In order to setup MedDream Token Service to work with a load balancer, the following steps should be performed:

1. Open MedDream Token Service application.properties and add the following lines (these lines need to be added to all MedDream Token-Service instances):

```
authentication.manager.username=random_user_name_for_token_service
authentication.manager.password=very_long_and_secure_password_for_token_service
management.server.port=8086
management.server.address=127.0.0.1

management.endpoints.enabled-by-default=true
management.endpoints.web.exposure.include=health


# When using spring.profiles.include=redis, same encryption parameters must
# be configured for all instances. Exact lengths are important:
# 32 characters for the key, 16 characters for the IV.

spring.profiles.include=redis
```

(continues on next page)

(continued from previous page)

```
com.softneta.token.encryption.secret-key=WJ8WUCP8sFTxccxzshpN2prwgKZVFABc
com.softneta.token.encryption.initialization-vector=enR9vu5HLx8cBabC
```

2. Save the application.properties file and restart MedDream Token Service service.

   After the service starts up, a file redis-config.yaml file will be created in the MedDream Token Service installation directory.

3. Open redis-config.yaml file and change this line to actual IP address and port of REDIS, for example:

   ```
   singleServerConfig:
     address: "redis://192.168.1.165:6379"
   ```

   NOTE: If you install MedDream Token-service using dockers, you can pass Redis URL by the following environment variable: `TOKEN_REDIS_URL:redis://192.168.1.165:6379`.

4. Restart MedDream Token Service once again and it will start to work with REDIS.

   NOTE: If you are using Secure Redis you need to use `rediss://` string. And it is possible to pass the password, too: `rediss://:password=@redis-server-url:6380/0?ssl_cert_reqs=CERT_REQUIRED`

5. Use the following URL for the health check of the token service: `https://tokenserviceurl:8086/manage/health`

Verify the final result: each MedDream Token Service node should be able to generate/validate independently.

5.17.6. High Availability behavior

With this configuration:

- if one MedDream Viewer or MedDream Token Service node fails, traffic is routed to remaining nodes;
- users do not lose sessions;
- cache remains consistent across nodes;
- system can be scaled horizontally by adding more nodes.

5.17.7. Validation Checklist

[ ] Load balancer health checks pass

[ ] All MedDream Viewer nodes are accessible through load balancer

[ ] All MedDream Token Service nodes are accessible through load balancer

[ ] Shared storage is mounted and writable on all nodes

[ ] Redis/Valkey is reachable from all nodes

[ ] Sessions persist across node restarts

5.17.8. Notes & best practices

- Use HTTPS termination at the load balancer
- Monitor Redis/Valkey availability
- Monitor shared storage latency
- Use identical MedDream Viewer and MedDream Token Service versions on all nodes

6. Additional software

See the summarized list below of third-party software that must or can be used in combination with MedDream:

1. Authentication and authorization. For secure user authentication (if study list access needed) its recommended to use external tools like Keycloak, Okta, AWS Cognito, Azure AD (or other tools supporting OAuth or SAML).
2. Log aggregator for ATNA logs. MedDream can generate ATNA-compliant audit logs. To support centralized logging, monitoring, and compliance requirements, it is recommended to use an external log aggregation system (for example, ELK Stack, Splunk, Graylog, or similar solutions). The system should support secure log collection, storage, access control, and retention according to organizational and regulatory requirements.
3. Monitoring and performance metrics. For system monitoring and performance analysis, it is recommended to use a monitoring solution such as Prometheus for metrics collection and Grafana for visualization. These tools allow monitoring of JVM metrics (heap usage, garbage collection, thread count), system load (CPU, memory, disk I/O), and application performance indicators. Proper monitoring helps ensure system stability, capacity planning, and early detection of performance issues.
4. Regular backups of the operating system, MedDream configuration, logs, and related data are recommended to ensure system recovery and business continuity. Backup tools depend on the deployment environment. For physical or virtual servers, enterprise backup solutions (e.g., Veeam, Bacula, Acronis) may be used. In virtualized environments, hypervisor-level backups or snapshots can be applied. In containerized or cloud deployments, platform-native backup and storage snapshot mechanisms should be used according to infrastructure best practices.
5. Deployment-related services. Depending on the deployment architecture and security requirements, additional infrastructure services may be required, including a supported 64-bit operating system platform (e.g., Linux or Windows Server), Web Application Firewall (WAF) and secure firewall for network protection, containerization platforms (such as Docker or Kubernetes) for deployment and scaling, time synchronization services (NTP/Chrony) to ensure consistent timestamps, storage layer services (SAN, NAS, cloud storage) for reliable data access, and enterprise identity services (Active Directory, LDAP, or equivalent) for centralized user and access management.

6.1. Browser plugin

A plugin for the Chrome browser that expands a newly opened tab across several monitors. The URL is scanned for an entered keyword and if a match is found, a new window is created and expanded across the selected monitors.

The plugin is also required for proper behavior when calling MedDream from external Programs. It can be installed using the Chrome extensions tab (from the browser store).

When using the plugin, it is important to give the MedDream website the “window placement” or “window management” permission. When opening MedDream for the first time on a particular browser, you should get a prompt that the website wants “manage windows on all your displays”, then click “Allow”. If you clicked “Block” back then, this can be adjusted by clicking the “lock” icon near the URL, choosing “Site settings” and looking for “Window management” there.

Since approx. MedDream 8.6.0, new versions of Chrome do not allow full control of windows that display non-SSL content. Consequently, MedDream must be accessed via an https:// URL, see 5.1.10 Enabling SSL.

In case of difficulties installing or configuring the plugin, please contact support@meddream.com.

6.2. Writing an ISO file to a USB stick

The legacy solution "MedDreamBurn" is/was for writing ISOs to optical disks. There is no direct equivalent for writing to USB sticks.

It might be possible, however, to work around using these freeware alternatives:

- **ISO to USB** (http://www.isotousb.com/) – no command line support and therefore not possible to start automatically with an ISO file already specified; *you will need to specify the file manually*. Because formatting removable media is a privileged operation, Windows UAC prompt might be triggered on every start.
- **Rufus** (https://rufus.ie/en/) – again no command line support. Might require a bootable ISO file although MedDream never creates these.

6.3. Alternative to MedDreamBurn

On Windows it might be enough to use the built-in functionality for writing optical disks. The following steps need to be performed on every client machine where the actual writing to removable medium takes place:

- Right-click on the file (ISO or BURN) and select "Open With".
- Select "Choose another app".
- Select "More apps".
- Scroll to the bottom and select "Look for another app on this PC".
- Browse to folder C:WindowsSystem32 and select `isoburn.exe`
- Click Open.

From now on clicking on those files will start the built-in burner. You might also want to select the checkbox "Open when done" while the file is still being downloaded.

7. Security considerations

RATIONALE: It is an old dilemma of using "security through obscurity". As a matter of fact, most MedDream installations tend to have a few security holes due to non-paranoid Web administrators. The very publication of this knowledge makes every old/unmaintained MedDream installation an easy target. But we must draw a line one day so that at least new installations are secure.

7.1. Default passwords

The default application.SAMPLE.properties sets up the `auth-inmemory` authentication method with a username "demo" and password "demo". It is included **only** to ease installation and might be known to everyone who has downloaded a recent MedDream release.

Do not forget to change it (or remove altogether) after the installation is finished.

7.2. Search engines

MedDream is shipped with the following sys/settings/robots.txt:

```
User-agent: *
Disallow: /

User-agent: AdsBot-Google
Disallow: /
```

This should prevent finding Internet-exposed MedDream login forms by searching for "MedDream" or "Softneta" (as most customers don't use rebranding). Afterwards an attacker could check for known issues or misconfigurations.

If you still would like to index your installation, or allow indexing for a particular search engine only, then sys/settings/robots.txt can be modified accordingly (see http://www.robotstxt.org/robotstxt.html) or even renamed/removed. The backend treats the file as a static anonymous resource and simply returns contents of the file if it is found.

7.3. DCM4CHEE 2.x

Official installation instructions offer some defaults that sometimes are left unchanged.

After a mindless installation there also will be some default user accounts:

- a database user "pacs" with password "pacs". Suitable for MedDream when using auth-db-connection;
- an internal user "admin" with password "admin" – suitable for MedDream and DCM4CHEE's web interface when using auth-dcm4chee2;
- an internal user "user" with password "user" – suitable for MedDream and DCM4CHEE's web interface when using auth-dcm4chee2.

Any of these accounts can be used to access sensitive patient data. We suggest to change passwords for all three, as soon as possible after the installation. The first one is used by DCM4CHEE to connect to the database; therefore, you will also need to update the file server/default/deploy/pacs-*-ds.xml (name depends on database used).

7.4. Embedding into IFRAME etc.

Availability to embed MedDream into FRAME, IFRAME or OBJECT containers allows untrusted websites to use MedDream in fraudulent manner ("clickjacking"). Since v6.2.1 this is restricted by default: only the server that hosts MedDream is allowed to wrap it with an additional container.

It is possible to adjust IFRAME permissions in the application.properties file. Valid values for `frameOptionsPolicy`: NONE, DENY, SAMEORIGIN. When NONE is used, then the X-Frame-Options and Content-Security-Policy headers are not sent. Another parameter, `security.contentSecurityPolicy`, can be used to explicitly control the Content-Security-Policy header.

```
security.frameOptionsPolicy=POLICY_VALUE
#security.contentSecurityPolicy = frame-ancestors 'self' IP_ADDRESS HOST_ADDRESS; default-src
↪'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; child-
↪src 'self' blob:; connect-src 'self' *; img-src 'self' data:; worker-src 'self' blob:;
```

Functionalities "Copy the image to the clipboard" and "Copy measurements to the clipboard" depend on IFRAME parameters, too. One must add the permissions "clipboard-read" and "clipboard-write" to the tag attributes:

```
<iframe src="meddream/index.html" allow="clipboard-read; clipboard-write">
</iframe>
```

7.5. php.ini

When the same machine also contains a PHP/Apache installation (needed by third-party software like PacsOne), then some PHP settings might make it less secure. This chapter lists most important recommendations and caveats.

Error handling

- `expose_php = Off`

  The value "Off" hides the version of PHP from webserver response headers or error messages and therefore makes it more difficult to scan for known vulnerabilities of PHP itself.

- `error_reporting = E_ALL`

  "E_ALL" is recommended for production. However, during installation/troubleshooting of any server, including a production one, a more detailed choice like "E_ALL | E_STRICT" might be needed. One could have two versions of this line and comment the inactive one out.

- `display_errors = Off`
- `display_startup_errors = Off`

  Use "Off" in production for both, or else error messages visible to the end user might reveal paths to files and other sensitive information.

- `track_errors = Off`

  Can remain disabled as MedDream's applet.php doesn't use $php_errormsg.

- `html_errors = Off`

  "On" is not needed and, due to display_errors=Off, has no effect anyway.

- `log_errors = On`
- `error_log = /valid_path/PHP-logs/php_error.log`

  Logging to a file is preferred over `STDERR`. The latter means Apache's error_log where PHP messages will be intermixed with Apache messages and therefore harder to read.

- `ignore_repeated_errors = Off`

  "On" is OK only as a temporary measure if there are lots of identical errors, they are filling up the log and nothing else can be done about that right now.

General settings

- `doc_root = /PacsOneInstall/php`

  In 7.5+, the only part of MedDream served over the Web is applet.php when copied to PacsOne's "php" directory. In this configuration you can specify this directory or some parent part of the path.

- `open_basedir = /PacsOneInstall/php`

Not relevant to MedDream anymore.

A non-empty setting provides more benefits (and influence) to other software, like PacsOne GUI, running on the same Apache server. Take caution and test all functionality.

- `include_path = .:/path/PHP-pear/`

  Keep this list as short as possible. This setting is not related to applet.php of MedDream, but might be used by other software if hosted on the same server.

- `extension_dir = /path/PHP-extensions/`

  Use a full path.

  This parameter is not relevant to MedDream anymore.

- `mime_magic.magicfile = /path/PHP-magic.mime`

  Use a full path.

  Not relevant to MedDream anymore. When using MedDream with PacsOne on the same host, PacsOne might still benefit on correct value.

- `allow_url_fopen = Off`

  "On" is not required by MedDream 7.5+.

- `allow_url_include = Off`
- `allow_webdav_methods = Off`

  "Off" for both is safer and doesn't interfere with any MedDream functionality.

- `variables_order = "GPCS"`

  A general recommendation, not relevant to MedDream anymore.

- `session.gc_maxlifetime = 600`

  Not relevant to MedDream anymore as sessions are now handled at Java side. Will influence behavior of PacsOne GUI, though.

  A smaller value will log the user off faster if the browser is closed, the computer is suspended or the Internet connection is cut. 10 minutes is a sane default and the legacy default is 1440 (24 minutes).

File uploads

- `file_uploads = Off`

  MedDream 7.5+ does not contain PHP-based Reporting functionality so this setting doesn't apply.

  "On" might be important for PacsOne, though, as its Study Notes and Upload Dicom Image functions work in a similar way.

- `upload_tmp_dir = /path/PHP-uploads/`

  Use a full path. A separate directory might make cleaning leftover files easier.

- `upload_max_filesize = 10M`
- `post_max_size = 40M`

  Files larger than upload_max_filesize will fail to upload. To account for overhead, ensure post_max_size > 4*upload_max_filesize and memory_limit > 16*post_max size.

- `max_file_uploads = 1`

  PacsOne always uploads a single attachment together with a Study Note so "1" is a safe choice.

Executable handling

- `enable_dl = Off`

Not relevant to MedDream anymore, and “Off” is a safer choice.

- `disable_functions = system, shell_exec, passthru, phpinfo, show_source, highlight_file, fopen_with_path, dbmopen, dbase_open, putenv, exec, popen, proc_open, move_uploaded_file, mkdir, rmdir, chmod, rename, chdir, filepro, filepro_rowcount, filepro_retrieve, posix_mkfifo`

  These functions are generally considered dangerous and applet.php of MedDream does not use them. However, some might be needed for proper functioning of PacsOne. Take caution and test the configuration.

Session handling

- `session.save_path = /path/PHP-session/`

  Use an absolute path. This will ensure that both Apache-integrated and CLI versions of PHP keep session files in the same location.

- `session.name = mysessid`

  You can rename the session cookie to something less obvious.

  Not relevant to MedDream anymore but other software on the same Apache server might benefit from this.

- `session.auto_start = Off`

  Not relevant to MedDream anymore, and “Off” is a safer choice.

- `session.use_trans_sid = 0`
- `session.use_cookies = 1`
- `session.use_only_cookies = 1`

  Not relevant to MedDream anymore, and “Off” is a safer choice. Take care with PacsOne, though.

- `session.cookie_domain = my.website.tld`

  Not relevant to MedDream anymore, and “Off” is a safer choice. Take care with PacsOne, though.

- `session.use_strict_mode = 1`

  “1” or “On” is more secure. There is no effect on MedDream functionality in any case.

- `session.cookie_lifetime = 0`

  Zero is the legacy value (the session exists while the browser is not closed) and end users are accustomed to it. This is more relevant to PacsOne than MedDream.

- `session.cookie_secure = 1`

  Not relevant to MedDream anymore. If Apache also hosts other software that is accessed only via HTTPS (plain HTTP is undesirable), then set this to “1”.

- `session.cookie_httponly = 1`

  “1” is a general recommendation. Not relevant to MedDream anymore, but might influence other software if hosted on the same server.

- `session.cache_expire = 30`

  Not relevant to MedDream anymore but might influence other software if hosted on the same server.

- `session.cookie_samesite = Strict # PHP 7.3+`
- `session.sid_length = 256 # PHP 7.1+`
- `session.sid_bits_per_character = 6 # PHP 7.2+`
- `session.hash_function = 1 # PHP 7.0-7.1`
- `session.hash_bits_per_character = 6 # PHP 7.0-7.1`

The above values are recommended and are understood by newer PHP versions.

- `session.referer_check = http://server/path`

  Not relevant to MedDream anymore but might influence other software if hosted on the same server. For PacsOne, you can set here a typical URL pointing to its GUI (directories only, without home.php etc).

Less obvious settings

- `memory_limit = 50M`
- `max_execution_time = 30`

  Not relevant to MedDream anymore. The setting provides more benefits (and influence) to other software running on the same Apache server.

- `report_memleaks = On`

  "On" might help to detect memory leaks earlier. Not relevant to MedDream anymore.

7.6. Browser's cache

Note: The browser cache is not encrypted by default, which may expose sensitive data. To protect sensitive information indirectly, it is recommended to:

1. Enable full disk encryption on the system.
2. Regularly clear your browser cache.
3. Avoid using public OS accounts or devices (e.g., shared PCs).
4. Use secure browser profiles or private browsing modes when handling sensitive data.

MedDream is designed with "zero footprint" concept in mind. From a security perspective this also means that no patient data is left on a client machine: after the end user logs out from MedDream, its cache does not contain any server responses with patient data.

Normally browsers obey the "do not cache" headers. Firefox, however, has a bug that allows someone to extract potentially sensitive data from browser's memory cache after the user logs out and doesn't close the entire browser application. On a shared computer, an attacker could later examine the memory cache and access data that isn't available to him via MedDream and HIS.

Shared computers make protecting sensitive data a lot more difficult. If this is unavoidable, then we recommend at least to avoid Firefox or teach users to also close the entire browser (not just a particular tab or one of the windows) after logging out from MedDream.

7.7. Firewalls

Every server, especially the ones that handle medical data, should have a working firewall.

MedDream obviously needs permission to listen at port configured by `server.port` for a typical HTTP(s) traffic. When using websockets (system.json: features.liveShare=true, or openTabsTrackingMethod=SOCKETS), they operate over the same port.

Similarly, the receiver for QueryRetrieve plugin will continuously listen at `com.softneta.dicomStoreService.port` for a typical DICOM traffic.

Outgoing network accesses might need to be whitelisted in a corporate firewall:

- the Register function downloads a license file from lic.meddream.com;
- the About window verifies availability of a new version at www.meddream.com;

- the Share function sends anonymized files to dicomlibrary.com.

7.8. Summary: Minimum relevant IT security requirements

- The MedDream installation package must be validated against the SHA-256 value published at meddream.com. (Scanning it with VirusTotal, https://www.virustotal.com/, is a good idea, too.)
  - Windows: `certutil -hashfile MedDream-DICOM-Viewer-8.9.1.zip SHA256`
  - Linux: `sha256sum MedDream-DICOM-Viewer-8.9.1.zip`
  - Mac OS: `shasum -a 256 MedDream-DICOM-Viewer-8.9.1.zip`

  If the displayed hash is the same as provided on the website for that file, then the file has been downloaded without corruption and can be used safely.

  The hash can also be downloaded as a separate file with the extension ".sha256" instead of ".zip": https://www.meddream.com/files/meddreamviewer/891/MedDream-DICOM-Viewer-8.9.1.sha256
- The operating system should be up to date with latest security patches.
- All not needed software should be removed from the system to reduce points of possible security breaches.
- Use the latest Java 17 version with all security patches.
- Server version should be hidden in HTTPS responses.
- Open only those firewall ports that are necessary to access MedDream (default Web port 80 or 443, and default DICOM port 11116 if used) or PACS servers.
- For HIS integration the secure token-based integration is highly recommended, instead of the legacy URL integration.
- The MedDream backend should not run under a privileged user, like Linux "root" user or Windows "Administrators" user group.
- MedDream files should have limited permissions (Linux example: 0775 for files, 0664 for directories).
- The MedDream backend should always run over HTTPS (with SSL encryption). Do not use self-signed certificates; if your company doesn't have one, then https://letsencrypt.org/ will provide it for free.
- Set up *ATNA event logging* to provide more details in case of a security incident.
- Main security-related settings in application.properties:
  - `server.servlet.session.cookie.sameSite = strict`
  - `server.servlet.session.cookie.secure = true`
  - `security.xssProtectionDisabled = false`
  - `security.contentTypeOptionsDisable = false`
  - `security.contentSecurityPolicy = default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; child-src 'self' blob:; connect-src 'self' *; img-src 'self' data:; worker-src 'self' blob:;`
  - `server.servlet.session.timeout` – set a reasonable duration based on network and system performance.
  - `authentication.manager.*` should be disabled. Enable it only for a short amount of time, and use a strong password (see below).
  - `management.server.port = 8081` (or similar) together with `management.server.address = 127.0.0.1`, if some software is monitoring the status of MedDream.
  - `spring.profiles.include = auth-inmemory`, `authentication.inmemory.*`, `authorization.users[]` should not be used in production configuration.

- MedDream should use strong passwords everywhere: at least 12 characters with uppercase, lowercase, numbers, special characters, and not listed in https://haveibeenpwned.com/Passwords.
- User roles must be only those that are absolutely necessary for a particular user. Less roles = more strict and safe account.
- `authorization.defaultLoginPermissions` should also be as minimum as possible, to prevent data breach from extended functionality abuse. User permissions should be based on user authentication and role control.
- The demo/demo user should be removed from the system.
- When using the bundled Token Service, it should run over HTTPS, too. See end of 5.1.10 Enabling SSL.
- When the Token Service is using the "redis" profile for round-robin parallelization, make sure to not use the example values of encryption parameters in its own application.properties; new random values must be generated.
- The bundled Token Service needs security.serviceUnderProxy=true in its own configuration file when requests are coming via a proxy, or else their IP addresses will be misleading. (Note that this setting effectively bans the requests that come not via a proxy; a "hybrid" infrastructure is currently not possible.)
- Search engines like Google should be disabled in robots.txt.
- Every application.properties file (remember that there are separate files for Token Service, config-server, etc) should not be readable by a regular user. Give the read permission only to a dedicated system account impersonated by the JVM process.
- Main security-related settings in system.json are the MedDream features. Disabling them also disables corresponding HTTP(s) endpoints:
  - turn off `search` if users won't use the Search window;
  - turn off `history` if users won't use the Patient History window;
  - turn off `reports` if users won't use the Reporting function;
  - turn off `hangingProtocols` if users won't use the Hanging Protocols function;
  - remaining features can also be disabled if not needed.
- Review other MedDream settings to disable any not used functionality.
- The installation should be backed up regularly, see 5.14 Backing up MedDream.

We suggest printing and filling a corresponding form available at https://meddream.com/files/meddreamviewer/doc/MedDream-DICOM-Viewer-Security-Checklist.pdf. It will provide a tangible evidence of system's security state.

Warning: IT security can be compromised in the event of:

1. Unauthorized user (hacker) got the access to the operating system and connected to the operating system where the medical device is installed. In such event the hacker can access sensitive patient data, can modify or impact medical device functionality and/or performance. MedDream cannot detect such event. Medical device user must apply best security practices to prevent such event by installing WAF, Firewall, Antivirus and other applications to detect possible security breach.
2. Unauthorized user (hacker) got user/password (or got access token) and connected to the system. In such event the hacker can access sensitive patient data, can impact medical device functionality and/or performance. MedDream cannot detect such event. Medical device user must apply best user authentication practices, provide user trainings to improve social engineering knowledge. Medical device user can use VPN, WAF, Firewall to limit unauthorized users' access to the medical device. Medical device user can track web server logs or proxy logs to track and measure possible security attacks (including DDoS attacks) to the system.

Warning: If the IT security is compromised, MedDream does not guarantee any functionality. If there is a risk that a security issue can impact patient safety or cause sensitive data leak, then it is recommended to stop using product as soon as possible until the issue is resolved. Depending on the risk impact (for example, when the issue cannot impact patient safety or leak sensitive data), the medical device user can continue using the device on his own risk. However, it is recommended to have an external standby system (redundant, spare system) which could provide full functionality (or part of it) instead of the compromised system. MedDream product is fully compatible with the possibility to deploy one or more offline/online standby systems to increase healthcare process safety, security and reliability.

Warning: "InMemory" users (the `auth-inmemory` profile) are recommended to use only for testing proposes. In production, please use DB users or token authentication.

If you found that IT security is compromised, for Emergency work while the system is being fixed, we recommend to create InMemory user(s) with needed permissions: for an administrator, add ADMIN and SEARCH permissions; for ordinary users, add the SEARCH permission.

Note: By default, MedDream does not detect a change of the client IP address during the session. There are legitimate cases when this change is possible (for example, because a mobile device is moving). You can experiment with the `security.detectRemoteIpChange` setting, though.

Note: **Incident Reporting** – We kindly ask you to report any incidents related to MedDream software by completing the incident report form (available in DOC or PDF format) and submitting it to support@meddream.com and dpo@meddream.com.

Any serious incident that has occurred in relation to the device should be reported also to the competent authority of the Member State in which the user and/or patient is established (included, but not limited to, the Therapeutic Goods Administration (www.tga.gov.au), as applicable).

8. Localization

By default, MedDream is available in English, Lithuanian and Russian; user documentation is only in English.

Warning: The manufacturer is not responsible for not verified translations made by licensed users. Please note that Medical Device certificates are not valid for not verified translations.

The default value of "languages" setting in sys/settings/system.json is just "en". You can specify any available language or their combination. For example, `:["en"]` can be changed to `:["lt","en","ru"]`.

9. Rebranding

MedDream can be rebranded and use custom branding information (e.g., logos, product name, some certification details). Four scenarios are possible:

1. **No rebranding** was the default in older licenses (v7.8.0 and older). MedDream displays the default logo pictures, while "Distributor" and "End user" data come from the license;
2. **Simple rebranding** is usually enabled and allows you to change logotypes, system name, default color theme;
3. **Full rebranding** (also called "Rebranding as a module", requires a special license) allows you to change more options, however these changes will invalidate CE, FDA and other medical certificates;
4. **Viewport rebranding** can be used together with Viewports API, where a large portion of MedDream user interface is not present and can be implemented by the customer. Additional analysis is required to determine which parts of the product will be certified, please contact support@meddream.com for assistance.

Since 7.9, the product version number can't be rebranded and MedDream will always check for the latest version online.

All rebranding-related files must be located in the directory `sys/rebranding/` below configRoot. The configuration file is "rebranding_configuration.json", it must be in valid JSON format (UTF-8, no comments, no embedded newlines).

Note: Since 8.9, the default rebranding configuration is shipped in form of multiple ZIP archives and is in effect missing initially. **You'll need to choose a file that corresponds to the license (human or VET, "used as a module" or not) and extract its contents.**

For your convenience, each example lists only non-ignored options.

The most detailed example in module.zip doesn't mimic MedDream any more, because in a rebranded product, the original MedDream certification information becomes invalid anyway.

An override for the Web resource `/favicon.ico` is always expected under a name "favicon.ico". This file must exist if the license allows rebranding.

Example configuration file for simple rebranding:

```
{
  "systemName":"MyHospital",
  "companyLogoFile":"companylogo.jpg",
  "productLogoFile":"productlogo.jpg",
  "loginLogoFile":"loginlogo.jpg",
  "defaultTheme": "red"
}
```

- `systemName`

  (optional) Name of the system into which MedDream is integrated. Displayed at top of About dialog if not empty, and overrides the product name in browser tab titles, below login form and below search results.

- `companyLogoFile`

  Name of a JPG/PNG file with the company logo displayed in the About dialog.

  Recommended size is 134 x 28 px. Images wider than 540 pixels do not always fit into the dialog. A too high image might move the buttons outside the screen so the dialog becomes unusable. Make sure to check your customization on various client systems (different display DPI, etc.) from where MedDream will be opened.

- `productLogoFile`

  Name of a JPG/PNG file with the product logo displayed in Viewer window (top left corner).

  Recommended size is 180 x 38. In the viewer the height can be up to 52 px, anything more is clipped; if the width exceeds 215 px, the rest will overlap with thumbnails. When your picture is near these limits, it might need checking on various client systems to ensure consistent behavior.

- `loginLogoFile`

  Name of a JPG/PNG file with the login logo displayed in Login window.

  Maximum dimensions are 270 x 50. Wider images will not be centered. Higher images will shift the form towards the bottom of the page.

- `defaultTheme`

  (optional) Selects a predefined color theme from values `red` (default for non-VET license), `blue`, `green` (default for VET license), `dark-blue`, `dark-green`, `orange`, `teal`. A custom name will work, too, if defined under `themes` below.

  Please note that this red/green adjustment depending on the license works only if this setting is absent or an empty string. A non-empty string always overrides the license.

Remaining configuration options are used in rare cases. Most of them (except safetySign) are ignored in case of a VET license.

- `productName`

  (Required when full rebranding) Replaces the text “MedDream” in the bottom of the login window, the “Product” part of the About dialog, the title of the browser tab, below login form and below search results; an empty value is used as is, and results in a strange look.

  For ordinary licenses (rebranding without isUsedAsModule), the product name still comes from the license.

  Example:

  ```
  "productName":"MyViewer"
  ```

- `manufacturedBy`

  (optional, full rebranding only) When true, the Distributor contacts from the license are displayed as “Manufactured By”.

  Example:

  ```
  "manufacturedBy":true
  ```

- `uniqueDeviceId`

  (optional, full rebranding only) Displayed as “Unique device identifier” in the About dialog. Consult your certification documents.

  Example:

  ```
  "uniqueDeviceId":"444444455"
  ```

- `medicalDeviceClass`

  (optional, full rebranding only) Displayed as “Medical device class” in the About dialog. Typically includes both class and legislation where this class is defined; consult your certification documents.

  Example:

  ```
  "medicalDeviceClass":"Example Class"
  ```

- `notifiedBodyId`

(optional, full rebranding only) Displayed as “ID of the notified body” in the About dialog. Consult your certification documents. A related image at the bottom is taken from md5/css/icons/certification-logo2.png when rebrandCertified=true.

Example:

```
"notifiedBodyId":"0123"
```

- `rebrandFdaClarification`

  (optional, full rebranding only) Displayed as “FDA Cleared” in the About dialog. Consult your certification documents. A related image at the bottom is taken from md5/css/icons/certification-logo.png when rebrandCertified=true.

  Example:

  ```
  "rebrandFdaClarification":"FDA Cleared B12345"
  ```

- `rebrandCertified`

  (optional, full rebranding only) When true, md5/css/icons/certification-logo.png and md5/css/icons/certification-logo2.png are still displayed despite rebranding.

  Example:

  ```
  "rebrandCertified":true
  ```

- `catalogueNumber`

  (optional, full rebranding only) Displayed as “Catalogue number” in the About dialog.

  Hardcoded value “MDSY” when the license does not allow full rebranding.

- `mobileDisplayLimitations`

  (optional, full rebranding only) Displayed as “Warning” in the About dialog.

  Hardcoded value “about.mobileDisplayLimitations” (a particular translation code) when the license does not allow full rebranding.

- `showRepresentativesInfo`

  (optional, full rebranding only) Enable display of the “REP” tab in the About dialog.

  Hardcoded value `true` when the license does not allow full rebranding.

- `safetySign`

  (optional, compatible with VET license) Object with additional properties for overriding the warning/caution/note symbols and a corresponding comment in the About dialog:

  - `warning`

    Name of the warning icon file (in the same directory as JSON file).

    For full rebranding, there is no default image and this setting must be configured in order to display an icon.

    For other kinds of rebranding, a default image is displayed if this setting is not configured.

  - `caution`

    Name of the caution icon file (in the same directory as JSON file).

    For full rebranding, there is no default image and this setting must be configured in order to display an icon.

    For other kinds of rebranding, a default image is displayed if this setting is not configured.

  - `note`

Name of the note icon file (in the same directory as JSON file).

For full rebranding, there is no default image and this setting must be configured in order to display an icon.

For other kinds of rebranding, a default image is displayed if this setting is not configured.

- `safetySignDescription`

  (Optional) Any text about the safety signs.

  Hardcoded value "about.safetySignDescription" (a particular translation code) when the license does not allow full rebranding.

- `safetySignLabel`

  (Optional) A text that replaces warning, caution and note icons.

- `operatingRequirements`

  (Optional) Underlying URL for the text "Operating requirements" below the safety symbols. The text is displayed under same conditions as the safety symbols.

- `certRegistrationNo`

  (optional, full rebranding only) Certificate registration number.

- `themes`

  (optional) A map with color codes for each theme.

  If this setting is missing or empty, the traditional 7 themes (with names documented at "defaultTheme" above) are still available. If you add at least one theme, then you should also provide the one named "blue" because the Theme button toggles between the default theme and the blue one. Missing themes "blue" (for the Theme button), "green" (default for VET license), "red" (default for non-VET license) are added automatically, too, but with a warning in the log.

  The file rebranding_configuration.json that comes with new MedDream versions contains details of all 7 traditional themes in order to provide a starting point for customers wanting to adjust them. The example below reveals supported attributes just in case; unfortunately there is still no public specification to which parts of the GUI they apply.

```
"themes": {
  "red": {
    "primaryColor": "#e51b48",
    "secondaryColor": "#d0d0d0",
    "gray1": "#383842",
    "gray2": "#323237",
    "gray3": "#27272b",
    "gray4": "#2f2f33",
    "gray5": "#4d4d4d",
    "gray6": "#8e8c8c",
    "lightGray1": "#afafaf",
    "lightGray2": "#c4c4c4",
    "white": "#fff",
    "black": "#000",
    "orange": "#f8691d",
    "red": "#e51b48",
    "blue": "#07aae5",
    "green": "#1fcd6c"
  },
  "blue": {
    //...
```

(continues on next page)

(continued from previous page)

```
  }
}
```

9.1. Changing the progress bar color

The files md5/css/md.default.*.css is minified and difficult to navigate. You should pretty print the file first, even free online tools like https://www.cleancss.com/css-beautify/ can handle that.

The following example changes the color of the progress bar – only this GUI component, in all themes – to pure yellow. From now on, themes.*.primaryColor in rebranding_configuration.json will still be used for other relevant GUI components, but not for this one.

```
--- a/md.default.45f29cfe5ead.css   2025-12-02 14:21:24.117685800 +0200
+++ b/md.default.45f29cfe5ead.css   2025-12-02 14:28:27.842127800 +0200
@@ -4015,7 +4015,7 @@
   background: var(--light-gray-1-color);
   border: none;
   border-radius: 0.5rem;
-  color: var(--primary-color);
+  color: #ffff00;
   display: block;
   height: 1rem;
   margin: 0.5rem 0;
@@ -4024,11 +4024,11 @@
   z-index: 1;
 }
 .series-progress-bar::-moz-progress-bar {
-  background: var(--primary-color);
+  background: #ffff00;
   border-radius: 0.5rem;
 }
 .series-progress-bar::-webkit-progress-value {
-  background: var(--primary-color);
+  background: #ffff00;
   border-radius: 0.5rem;
 }
 .series-progress-bar::-webkit-progress-bar {
@@ -6970,12 +6970,12 @@
 .grid-viewports-layout
   .grid-viewport-container
   .md-progress-bar::-moz-progress-bar {
-  background: var(--primary-color);
+  background: #ffff00;
 }
 .grid-viewports-layout
   .grid-viewport-container
   .md-progress-bar::-webkit-progress-value {
-  background: var(--primary-color);
+  background: #ffff00;
 }
 .grid-viewports-layout .instant-notifications {
   display: flex;
```

10. Troubleshooting

10.1. Preventive measures

The most important step in preventing problems is monitoring, especially for some time just after the installation.

System administrators should regularly watch:

- amount of free space on disks used by MedDream;

  Currently MedDream does not stop creating temporary files if amount of free space is dangerously low, and will proceed until its amount drops to zero. This might result in corrupt databases (reindexing usually takes a long time) or corrupt cached files (difficult to identify which ones are corrupt, and too costly to remove everything). If other software is using the same disks, it might get similar issues. To minimize impact, it is quite important to timely react to low disk space situations.

  The size of temporary/cached files is proportional to user activity. In some installations the daily number of new studies sent to the PACS might have a larger influence. Both are not constant and MedDream therefore might need a periodic review of data age thresholds in *temporary files cleaner configuration*. But sometimes the only solution is to add more storage.

- amount of free memory, and memory used by the Java process;

  If your license limits number of concurrent connections, then one-time implementation of recommendations in 5.1.9 Adjusting Java memory limits should be sufficient.

  In case of unlimited connections, it is important to know the actual usage statistics first. The easiest way is to collect the activeConnectionCount property from MedDream's offline statistics files (YYYY-MM.stat).

- CPU and I/O load from the Java process;

  Helps to identify hardware inadequate for the task.

- whether a time synchronization service is successfully adjusting the system time;

  A correct system time on the server is important for many scenarios:

  - analysis of logs during troubleshooting (incorrect time zone increases the confusion even more);
  - caching of application resources in the client-side web browser because it is based on resource timestamps reported by server;
  - validation of SSL certificates in the MedDream backend when it securely connects to other systems as a client;
  - integration with Google Cloud Healthcare might fail if time difference is more than one hour;
  - if the JWT security layer for HIS integration is enabled, then it allows only 5-second difference by default (see the setting `authentication.his.allowedClockSkewSeconds`).

  Other software on the server might exhibit different effects if the time is off by a too large amount.

- number and nature of "ERROR" messages in MedDream logs;

  Not every problem is worth an error message seen by the end user. Some errors are visible only in the logs, and indicate suboptimal configuration, corrupt or non-compliant DICOM files, or simply MedDream bugs.

- the `/manage/health/md` endpoint can be used to track concurrent connections and other licensing details in real time. This is especially useful if the license has a limited number of connections. Example response:

```
{"status":"UP","components":{"licensing":{"status":"UP","details":{"cc":1}},
↪"ping":{"status":"UP"}}}
```

10.2. Antivirus software

Excessive virus scanning may severely impact performance. There is no sense to scan the MedDream preparation cache (com.softneta.preparation.cacheDir, or "cache" under tempDir by default) so this directory tree should be excluded, especially when using the in-advance preparation and expecting a large amount of data there.

You should also consider whitelisting the DICOM files themselves on the PACS. Even if some are routinely uploaded over a custom Web interface, their on-demand scanning in a corresponding temporary directory (before importing to the PACS) might be more adequate.

10.3. Log files of the Java-based core

In the directory of the Java application, the file `meddream.main.log` will be created.

To control the overall logging level, adjust the following line in `application.properties`:

```
logging.level.root=DEBUG
```

Alternatives to "`DEBUG`" are INFO, ERROR etc. as per Log4j specification.

More specific setting, `logging.level.com.softneta`, refers only to internal MedDream logic and in some cases might be insufficient even at the TRACE level. On the other hand, `logging.level.root` yields so much data that it becomes impractical after a few minutes.

When investigating SQL queries in direct integrations, the following might be useful:

```
logging.level.org.hibernate=DEBUG
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
spring.jpa.show-sql=true
```

Lots of third-party information related to user authentication:

```
logging.level.org.springframework.security=DEBUG
```

Logic of the temporary files cleaner is spread among the following modules:

```
logging.level.com.softneta.meddream.service.temp.TempCleanerService=DEBUG
logging.level.com.softneta.meddream.util.PathUtils=DEBUG
logging.level.com.softneta.meddream.util.DirTreeRemover=DEBUG
```

Dicomweb and Orthanc plugins depend on the following third-party library that can display HTTP request/response headers (but not POST data):

```
logging.level.sun.net.www.protocol.http=DEBUG
```

Some parts of MedDream, like QueryRetrieve plugin, are using the third-party Dcm4che2 Toolkit for DICOM network operations. It has its own logging facility:

```
logging.level.org.dcm4che2.net=DEBUG
```

PACS integration plugins themselves (pick the relevant one):

```
logging.level.com.softneta.pacs.gateway.plugin.pacsone=DEBUG
logging.level.com.softneta.pacs.gateway.plugin.dcm4chee5=DEBUG
logging.level.com.softneta.pacs.gateway.plugin.dcm4chee2=TRACE
logging.level.com.softneta.pacs.gateway.plugin.clearcanvas=DEBUG
logging.level.com.softneta.pacs.gateway.plugin.conquest=TRACE
logging.level.com.softneta.pacs.gateway.plugin.orthancplugin=TRACE
logging.level.com.softneta.pacs.gateway.plugin.dicomweb=DEBUG
logging.level.com.softneta.pacs.gateway.plugin.qr=TRACE
logging.level.com.softneta.pacs.gateway.plugin.filesystem=DEBUG
logging.level.com.softneta.pacs.gateway.plugin.amazon.s3=DEBUG
```

The levels listed above correspond to the maximum level used by a particular plugin.

When troubleshooting cache behavior, it is sometimes needed to know what UIDs are hidden beyond those random-looking subdirectories, like `.../bb/b2/ef/eedc376fdb0d545d9344b67092ee2e78d2/meta`. This is revealed by

```
logging.level.com.softneta.meddream.util.HashedPath=TRACE
```

Unfortunately, there will be a lot of corresponding messages, as they appear multiple times even when loading a single-image study.

Under Windows, the MedDream.exe service wrapper duplicates the logged information into files MedDream.err.log and MedDream.out.log. The latter is sometimes more informative than meddream.main.log, and its size can be controlled separately. To adjust the size and number of files MedDream.out.*.log, add the `<log>` element to MedDream.xml, then modify its `sizeThreshold` and `keepFiles`:

```
    <log mode="roll-by-size">
        <sizeThreshold>10240</sizeThreshold>
        <keepFiles>10</keepFiles>
    </log>
</service>
```

10.4. Video conversion

When a video file is being converted to MPEG4, then stdout and stderr of the FFmpeg process are redirected to a troubleshooting file `${com.softneta.video.convertedDir}/<SOP Instance UID><storage ID>.out`; the converted video goes to a temporary file `${com.softneta.video.convertedDir}/<SOP Instance UID><storage ID>.tmp.mp4`. When FFmpeg finishes, the temporary file is renamed to the final version, `${com.softneta.video.convertedDir}/<SOP Instance UID><storage ID>.mp4`. By default, convertedDir points to a subdirectory “temp/converted-videos”.

The conversion has no formal time limit. But, while the temporary file still exists, its modification time should always be recent (which is the case if FFmpeg still writes to it). If the time doesn’t change for more than 120 seconds, then MedDream assumes FFmpeg is likely stuck or its wrapper script has crashed, and attempts to remove all related files (.out, .tmp.mp4, .mp4).

On success the troubleshooting file is removed, and the final file is moved to the preparation cache for later use. On failure both files remain for analysis. An empty final file is considered a failure, too.

Since 8.2.0 **subsequent retries are not possible (the converter doesn’t even start) while the final file (*.mp4) still exists**. You must remove it manually in order to try again. The suggested configuration of *Temporary files cleaner* removes any files older than one hour under convertedDir, every 30 minutes.

10.5. Client side: browser download path selection

10.5.1. Google Chrome

You can choose a location on your computer where downloads should be saved by default or pick a specific destination for each download.

- On your computer, open Chrome.
- At the top right, click More and then **Settings**.
- At the bottom, click **Advanced**.
- Under the “Downloads” section, adjust your download settings:
- To change the default download location, click **Change** and select where you’d like your files to be saved.
- If you’d rather choose a specific location for each download, check the box next to “Ask where to save each file before downloading”.

If you didn’t change your default download location, then Google Chrome downloads files to the following locations:

- **Windows 10, 8, 7 and Vista:** \Users\<username>\Downloads
- **Mac:** /Users/<username>/Downloads
- **Linux:** /home/<username>/Downloads

10.5.2. Mozilla Firefox

Note: Changing the location of your downloads affects **all** downloaded files in this Web browser.

- Click the menu icon on the top right corner of the browser.
- Click **Preferences**.
- Click **General**.
- Click **Choose...** next to Save files to.

10.5.3. Microsoft Edge

- Open Microsoft Edge.
- Select **Settings and more** -> **Settings**.
- Under **Downloads**, select **Change**.
- In the dialog box, select a new location for your downloaded files.

10.5.4. Safari

To change the default download location of your Safari browser:

- Click on the **"Edit Menu"** -> **Preferences** -> **General tab**
- Locate the **"Save downloaded files to"** section, Click on **"Downloads"** > **"Other"...**
- Browse and indicate your new download location.

10.6. Client side: enabling clipboard operations

The buttons "Copy the image to the clipboard" and "Copy measurements to the clipboard" **become hidden** if the conditions below are not met:

1. the connection must be HTTPS or to a "localhost" address (when MedDream is opened on the server itself). This is a limitation in all Web browsers;
2. Firefox also requires `dom.events.asyncClipboard.clipboardItem` with value "true" in about:config. This setting must be created or adjusted on every client machine;
3. when MedDream is integrated into another website via an IFRAME, then the latter must have the `allow="clipboard-read; clipboard-write"` attribute. This, however, is a server-side change and needs to be done at the website that integrates MedDream.

10.7. Client side: monitor calibration

Note: Proper display of color images usually requires the "sRGB" color profile set up in the browser. For Chrome, this can be changed at the chrome://flags/#force-color-profile page.

If you suspect that the monitor does not show small differences in brightness, especially at the dark or at the bright end of the entire range, then the first step is to confirm the problem using third-party resources:

- https://www.photofriday.com/info/calibrate
- http://www.lagom.nl/lcd-test/

Afterwards you may attempt to calibrate the monitor.

- Windows: https://support.microsoft.com/en-us/windows/calibrate-your-built-in-display-for-hdr-content-in-windows-de1c66fa-6cc0-b327-e73a-1bac6bd46bc0
- Linux: https://help.ubuntu.com/stable/ubuntu-help/color-calibrate-screen.html.en
- MacOS: https://support.apple.com/en-us/105101

10.8. Symptom: on Linux the loading pauses just after logging in, possibly with wrong webpage colors

On some Linux installations, the Java backend might pause for up to several minutes when a new concurrent connection is made. Immediately after the login, the user sees either an empty dark page with a progress bar spinning at the top, or an empty white page, or a white page with a black-and-white DEMO message.

The cause: the default configuration of OpenJDK suggests reading from /dev/random that can block due to exhausted entropy pool. Some systems are refilling the pool more slowly so this problem becomes more common.

Solution 1: add `-Djava.security.egd=file:///dev/./urandom` to the command that starts the Java backend. Refer to the 5.2 Running as a service chapter for hints where this command might be located.

It is recommended to stop the service, then start the updated command manually and verify the effect. You might want to force entropy exhaustion so that the problem appears as soon as possible; use `dd if=/dev/random of=/dev/null bs=512 count=1` for that (on a different terminal as it will block, too).

Solution 2: the above might still not help with some OpenJDK builds, therefore you'll need to update system-wide configuration in the java.security file (traditionally it's `$JAVA_PATH/jre/lib/security/java.security`):

```
securerandom.source=file:///dev/./urandom
```

10.9. Symptom: queries to MS SQL Server take seconds from Java but milliseconds from Management Studio

A very poor performance was observed in a particular installation of Conquest over ODBC, even for simple queries by object UID. Adding `sendStringParametersAsUnicode=false;` to com.softneta.meddream.pacs.configurations[].url fixed that.

The default `true` means that all query parameters are sent to server as NVARCHAR, even for columns of VARCHAR type. The type conversion results in a full table scan so performance with large tables drops down.

If correct representation of non-Latin characters is important, then you should leave `true` and the database schema should use NVARCHAR in all MedDream query key columns (including UIDs). When national characters are not used, then `false` is a simpler solution.

10.10. Symptom: video not playable on Firefox under Windows

Some DICOM MPEG4 videos play on recent Chrome and Edge browsers but not on Firefox under Windows. The "loading" animation (spinning dots in a circle) is displayed forever, the browser's console contains "Decoder may not have the capability to handle the requested video format with YUV444 chroma subsampling". Older Firefox versions tell that the file is corrupt.

Unfortunately, Firefox team claims that under Windows support is not possible due to legal limitations. This video format is desirable, though, as it might provide a higher image quality.

If too many end users of yours can't use another browser, then you should force a format downgrade by adding `-pix_fmt yuv420p` to FFmpeg command line. Afterwards all browsers will encounter the more popular 4:2:0 chroma format, and there will no benefit for those that support 4:4:4.

Example for software encoder (default command line) with this additional parameter:

```
com.softneta.video.commandLineTemplates.generic =-i {INPUT} -y -vcodec libx264 \
-preset medium -pix_fmt yuv420p -movflags faststart {OUTPUT}
```

Example for Nvidia hardware encoder with this additional parameter:

```
com.softneta.video.commandLineTemplates.generic = -hwaccel cuda -i {INPUT} -y -vcodec \
h264_nvenc -preset slow -pix_fmt yuv420p -movflags faststart {OUTPUT}
```

8.5.0+ offers more flexibility. It is possible to define both lower-quality and higher-quality versions, for example,

```
com.softneta.video.commandLineTemplates.codec_64001E =-i {INPUT} -y -vcodec libx264 \
-preset medium -pix_fmt yuv420p -movflags faststart {OUTPUT}
com.softneta.video.commandLineTemplates.codec_F4001F =-i {INPUT} -y -vcodec libx264 \
-profile:v high444 -pix_fmt yuv444p -movflags faststart {OUTPUT}
```

Note, however, that this requires a traditional “non-free” build of FFmpeg, which you’ll need to install separately in 8.6.0+ (the bundled FFmpeg is quite limited in functionality).

10.11. Symptom: strange “class not found” exceptions when restrictions are present in the token

When the token-enabled integration specifies restrictions, like

```
{
    "items": [
        {
            "studies": {
                "study": "1.2.826.0.1.3680043.8.1055.1.2020050611210409.446757335",
                "storage": "OrthancDicomweb"
            }
        }
    ],
    "restrictions": {"patient": ["pt-014597"]},
}
```

then MedDream might spend more time removing an old and incompatible session. If the same end user attempts to quickly open multiple tokens, without waiting for a particular study to load fully, those studies might fail due to a race condition, which is indicated by strange errors in the log: “Something went wrong cache key not found”, License connections NNN exceeded”, “Failure to copy resource”, “Error reading request, ignored”, etc. Failure is more likely with multiple concurrent users and high system load.

In case of URL integration, one could work around with authentication.his.useSameSession=true. The token-enabled integration has no equivalent, however removing the “restrictions” element from the token lessens the chance that a new session will be created.

Furthermore, the proper way to compare multiple studies is to specify them in a single token at once.

10.12. Symptom: “Unable to determine Dialect without JDBC metadata” when connecting to a database

Versions of the Hibernate library used in MedDream 8.3+ are trying to guess the dialect based on real communication with the server. (To make the configuration simpler, the PACS integration plugins do not accept a dialect configuration option, while a hardcoded value is specified only when absolutely needed and unambiguous.)

As a result, the error “Unable to determine Dialect” can misleadingly appear even when there is a TCP/IP communication problem (wrong address or port, a firewall is interfering, etc.). You should first check the address/port combination from the same machine via a telnet client.

Furthermore, in some cases the connection URL must refer to a correct database type, like “jdbc:mariadb:. . . ” for MariaDB or MySQL.

10.13. Symptom: MPR views show as black screen in Chrome

If the MPR views are black unless flipped or significantly rotated, then the problem is likely in OpenGL.

As of MedDream 8.8 and Chrome ~141, the default rendering engine under Windows is DirectX and it doesn't have this issue. The machines with older browsers where OpenGL was chosen for better performance, are affected and should switch to DirectX. Note that recent Chrome builds for Windows offer only DirectX, therefore an update will fix this, too.

Under Linux the default is OpenGL, and it should be possible to change to Vulkan.

For switching to a different engine, open the Chrome flags page (chrome://flags), then search for "ANGLE".

10.14. Known limitations and their solutions

10.14.1. Share functionality skips some objects

By design, Share to DICOM Library does not send the following objects in "Active study" mode:

- those that are hidden due to blackListedSopClasses in system.json, and due to another hardcoded list of known unsupported SOP Classes;
- old versions of Bounding Box, Free Draw and Smart Paint annotations.

If they are important, then one should use the Export function and upload the resulting archive to DICOM Library manually.

A. Appendix: Configuration changes since 8.8.1

Note: Names marked with # (for example, "#datasource.quartz") are internal properties, not to be changed by the end user. Most of them are also not offered in application.SAMPLE.properties to discourage any changes.

A.1. Core service command line

The fragment `-Djava.library.path=./lib` is no more required and in some cases might interfere. Please remove it from systemd unit files or any scripts you might have, do not use it when starting the JAR from the console temporarily, etc.

A.2. application.properties, generic

A.2.1. New options or values

`com.softneta.preparation.partitions.*`

Every type of file in the preparation cache ("meta", "struct", "workspace_*", etc.) can be kept in a separate directory tree that starts below the Storage ID directory, by adding one additional subdirectory configured here.

`com.softneta.meddream.dbCleanerProperties.neverDeleteSuffixes`

Adjusts what files are usually removed via the DB index by the standalone "preparation-db" application. Depends on a particular non-default "partitions" configuration above.

`com.softneta.meddream.dbCleanerProperties.internalCacheDuration`

The standalone "preparation-db" application caches some database read results in memory for this amount of time.

`security.allowedAuthErrors`

Now supports values `reachedMaximumAttemptsMin` and `reachedMaximumAttemptsSec` for the message "Reached maximum login attempts". They are included in the default set; the default now also contains `connectionLimitReached` (6 codes in total).

`com.softneta.preparation.prepareFromPacsMetadataForTheseTransferSyntaxes`

Use the new "prepareFromPacsMetadata" scenario (enabled via a plugin setting with this name) for the specified values of Transfer Syntax.

Contains two values by default, ...2.4.202 and ...2.4.50. Before 8.9.0, a single value ...2.4.202 was hardcoded.

`com.softneta.cache.studyStructureExpiresOnChange`

If true, on every structure request from the frontend, Core fetches NumberOfStudyRelatedSeries / NumberOfStudyRelatedInstances to automatically expire a cached study structure. Then the related functionality implemented earlier (structureListenerConfig in system.json) becomes less important: it will only indicate that a study has changed while being viewed in MedDream.

`com.softneta.meddream.mpr.maxMemoryBytes`, `com.softneta.meddream.mpr.maxConcurrentJobs`

An optional limit on MPR Export jobs. Multiple jobs and larger series bring a risk of exhausting the memory on the server, which should be avoided as Core sometimes needs a restart to recover fully.

`com.softneta.meddream.enabledColorPalettes`

4 new palettes are included below `sys/clut`, and one of them is enabled by default. The default list now includes all standard palettes, too.

`com.softneta.settings.dicomPrintFileName`

Possibility to rename the settings file for DICOM Print functionality.

`com.softneta.meddream.dicomPrint.printScuAet`, `com.softneta.meddream.dicomPrint.taskThreadCount`

Global, not user-modifiable, settings of DICOM Print functionality.

`redis.config.server`, `redis.config.cache`

A replacement for `redis.config-yaml` which was the path to a YAML-formatted configuration file.

The new implementation is meant to be configured on a Kubernetes cluster via environment variables. The corresponding variables, REDIS_CONFIG_SERVER and REDIS_CONFIG_CACHE, will then contain *a multiline fragment* of the previous YAML file.

`redis.token-refresh-rate-in-minutes`

When using Redis, a periodic refresh of Azure Entra tokens prevents them from becoming stale.

A.2.2. Options that are handled differently

`com.softneta.audit.requireUserIdentity`

A bug has been fixed that is related to Login window scenarios. Now you're welcome to always use `true`. (Just in case, `false` is still the default but this might change in the future.)

`security.contentSecurityPolicy`

There is a safer default value; however, some things might break, especially custom implementations that use the Links plugin.

A part of the default value has changed as follows:

```
- default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-
↪eval' 'unsafe-inline'; child-src 'self' blob:; connect-src 'self' *; img-src 'self
↪' data:; worker-src 'self' blob:;
+ default-src 'self'; base-uri 'self'; object-src 'none'; style-src 'self' 'unsafe-
↪inline'; script-src 'self' 'unsafe-eval'; img-src 'self' data: blob:; connect-src
↪'self'; frame-src 'self' blob:; worker-src 'self' blob:; form-action 'self';
```

Because the full default value depends on a different setting (security.frameOptionsPolicy), you might need to read on *Browser's security policy* even if you want to blindly return the previous behavior.

`com.softneta.preparation.nfsSynchronizationSingleStepSleepTimeoutInSeconds`

A bugfix: in versions 8.8.0 . . . 8.8.1, the value was interpreted as milliseconds and therefore the actual delay was 1000 times smaller. If your system was impacted by this and the value was adjusted accordingly, now you'll need to change it back to seconds.

`com.softneta.dicomStoreService.aetMappingAppliesToPrepare`

If true, `...aetToFolderMapping` remaps not only folders but also AE Titles in order to avoid the message "No plugin found with storeScpAet=. . . , will not prepare".

A.2.3. Removed options

`redis.config-yaml`

Has been replaced with two separate options (in fact, Kubernetes environment variables) that themselves contain YAML fragments.

A.3. application.properties, AmazonS3 plugin 3.8.1

A.3.1. New options or values

`storageApiEnabled`, `uploadServiceBaseUrl`, `checkInstanceAvailableUrl`, `checkInstanceCustomHeader`, `customParameters`

Support for saving annotations back to the server; only studyStructureType=TELEPAXX so far.

`studyStructureCountersEnabled`, `studyCountersUrl`, `studyCountersCustomHeader`

Support for study structure changes detection via Core's structureListenerConfig or studyStructureExpiresOnChange.

A.4. application.properties, AWSHealthImaging plugin 1.3.11

A.4.1. New options or values

`maxStringLengthOfJsonField`

The Jackson library (used to parse server's JSON responses) enforces a length limit on string values – practically a measure against potential DoS attacks via maliciously crafted resources, when there is no corresponding limiting at the PACS side. A higher limit can be configured if some valid attributes are that large.

The earlier version was using a library default, 20 000 000. The configurable default is now 25 000 000. (Spaces were added here for clarity; do not use them in the value.)

`studyStructureCountersEnabled`, `studyCountersUrl`

When fetching the study structure in QIDO-RS mode, it can contain NumberOfStudyRelatedSeries / NumberOfStudyRelatedImages needed by structure changes polling (structureListenerConfig in system.json).

A.5. application.properties, Dicomweb plugin 6.16.1

A.5.1. New options or values

`studyCountersUrl`

A QIDO-RS query dedicated to returning only NumberOfStudyRelatedSeries and NumberOfStudyRelatedInstances. Helps updating the cached structure (both in preparation and plugin caches) automatically.

A.6. application.properties, FileSystem plugin 7.3.0

The plugin now supports environment variables, see 5.1.22 Passing configuration via environment variables.

A.7. system.json

Note: For brevity, pseudo-code with hierarchical property names will be used instead of quoting hierarchical chunks of JSON. For example, `features.export: false` means the same as

```
{
    "features": {
        "export":false
    }
}
```

A.7.1. New options or values

`features.allowForwardSeriesAndInstanceDicomToPacs`

By default MedDream in some scenarios allows selecting series and images to be forwarded (a new feature of 8.9.0). If false (disabled), only the entire study can be forwarded like before.

`features.callCachedMetadataForTransferSyntax`

Alternative to features.callCachedMetadata=true: /cachedMetadata is called when a Transfer Syntax listed here was found in the study structure.

`features.mprExport`

Globally disables or enables the MPR Export functionality.

`features.dicomPrint`

Globally enables or disables the DICOM Print functionality.

`features.dicomPrintSettings`

Enables the “DICOM Print” page in the Settings window.

`features.largeVolumeMprSplit`

If true, MIST MPR rendering is performed in parts (up to 4 separate rendering jobs) so that Chrome’s texture size limit is effectively 4 GB.

`textureInterpolationType`

This old setting now supports the value 5 (Lanczos algorithm). Furthermore, the setting now provides just a default value; the actual interpolation depends on end user’s choice via a new toolbar button.

`segmentationConfig.modality`

Allows to configure for what kinds of images the segmentation functionality is enabled. The default value was hardcoded before 8.9.0.

`segmentationConfig.sopClass`

A different way to allow the segmentation functionality. The default value is empty.

Before 8.9.0, it was allowed to segment some RGB images with grayscale content due to an additional hardcoded condition: modality=PX **and** sopClass=1.2.840.10008.5.1.4.1.1.1.1. Because the default list of modalities already includes “PX”, and the “and” condition is not supported, the functionality is now available for more images; the full equivalent of earlier behavior is not

possible. On the other hand, MedDream now attempts to convert RGB pixels to 16-bit values during segmentation.

`multiframeCacheRetention`

To combat excessive browser memory consumption by multiframe images, pixels in display-ready form are now cached for this time (20 seconds by default) after their last display.

The overall effect is less memory at a cost of slightly slower initial display of a multiframe image.

It is not possible to return the previous behavior; both zero and very large values have a different effect compared to 8.8.1 with any value of preloadAllFrames.

`preloadVtkNatives`

Core includes the VTK library currently needed by the MPR Export functionality. If the latter will be used often in the installation, `true` here loads the library in advance (during Core startup), which means no delay during first use and any problems are be known beforehand.

A.8. rebranding_configuration.json

Starting from 8.9.0, by default there is no rebranding_configuration.json (multiple versions of it are stored in example archives) and rebranding is effectively disabled. Furthermore, the examples no more try to mimic an original look of MedDream.

Your old JSON file should work as before. There might be subtle effects due to how missing, null or empty values are now handled; please verify the result at least if your file contains some empty values on purpose.

A.8.1. Options that are handled differently

`productLogoFile`, `loginLogoFile`, `companyLogoFile`, `productName`

In partial rebranding, empty values are ignored so that the frontend can use its defaults. In full rebranding, missing or empty values are returned as empty.

`systemName`

Empty value is always ignored.

A.9. medcadPlugins/links.json

This MedDream version is definitely not compatible with the old plugin. Please use the new links.jar.

A.9.1. Removed options or values

`position`

Was unused for a long time already. Existing plugins can only be hosted in a Viewer window, there are no plans for plugins designed for a Search window.

B. Appendix: Upgrading 8.8.1 to 8.9.1 (Linux)

1. Stop the Core service (and the Token Service if it is used). The method depends on how you are running it, see 5.2 Running as a service.
2. Back up the old MedDream folder:

   ```
   mv /opt/meddream /opt/meddreamBKP
   ```

3. Extract new MedDream version to /opt/meddream/ so that the file /opt/meddream/MedDream-8.9.1.jar exists.
4. Copy the following old files to the new version:
   - meddream.lic
   - sys/settings/*.json (except system.json and hangingProtocolsConfig.json)
   - sys/settings/robots.txt
5. Review JVM parameters in your service wrapper script (5.2 Running as a service). For instance, the example System-V scripts bundled with MedDream won't contain changes made by you, and an example might still need renaming to the final file.

   Do not forget to remove -Djava.library.path=./lib from the command line.
6. If application.SAMPLE.properties was used during the initial installation, then copy the new file to application.properties and for every setting, copy the value from old application.properties.

   (During every upgrade, it is advised to keep the original application.SAMPLE.properties: later it can be compared with the new version and therefore save some time.)

   If the old file does not contain the same setting, then consult *Appendix: Configuration changes since 8.8.1* or the entire list 5.1.13 Reference for remaining configuration parameters. Probably it has been renamed, or is totally new.

   Warning: You might be tempted to simply copy the old application.properties to the new version and then review it. This is not recommended:
   - A higher risk of keeping settings not supported any more, which might lead to confusion later.
   - No opportunity to encounter default values (or totally new settings) in application.SAMPLE.properties that are safer security-wise, especially if overlooked.
   - No opportunity to encounter comments/explanations that might be more comprehensive in newer versions of application.SAMPLE.properties.
7. If, however, your application.properties is made from scratch instead, then consult *Appendix: Configuration changes since 8.8.1* or even the entire list 5.1.13 Reference for remaining configuration parameters.
8. If rebranding is used, then first choose a suitable ZIP file in sys/rebranding and extract its contents. For every setting in the new sys/rebranding/rebranding_configuration.json, copy the value from a setting in the old version.

   If the old JSON file does not contain the same setting, then consult *Rebranding*. It might be renamed or totally new.

   Afterwards copy all referenced image files from the old sys/settings/rebranding directory to the new one.
9. Update file permissions (example below is for MedDream running as user "meddream"):

```
chown -R meddream:meddream /opt/meddream
find /opt/meddream/ -type d -exec chmod 775 {} \;
find /opt/meddream/ -type f -exec chmod 664 {} \;
chmod 600 /opt/meddream/application.properties
chmod +x /opt/meddream/sys/ffmpeg/linux_x86-64/flv.sh
chmod +x /opt/meddream/sys/ffmpeg/linux_x86-64/ffmpeg
#chmod +x /opt/meddream/sys/ffmpeg/linux_arm64/flv.sh
#chmod +x /opt/meddream/sys/ffmpeg/linux_arm64/ffmpeg
```

(the commented-out lines are for installations on ARM platform).

DO NOT RUN MedDream AS ROOT.

10. Start the JAR so that sys/settings/system.json is created automatically. For every setting, copy the value from the old system.json. Consult the documentation (5.1.2 System.json) and list of configuration changes (*Appendix: Configuration changes since 8.8.1*).

> **Warning:** Unmodified system.json from older versions might be incompatible and result in a crash when starting the JAR.
>
> If you went the non-recommended way and are upgrading in-place, then old files sys/database/*.mv.db might result in errors while the JAR is starting. In case of errors related to H2 database engine, some of the files will be automatically renamed but the report database must be moved elsewhere manually so that new ones can be created.

11. Open the Settings window, review/adjust the configuration and save it. Without toolbarButtonsOrderingEnabled = false, the ordering of toolbar buttons and actions might differ from the saved one, and this new order will be forced until settings are manually reordered and saved by the new version.
12. Synchronize the newly created hangingProtocolsConfig.json with your old file. 8.9.0 brings 3 new study conditions that will be obvious when comparing the files. If your old file wasn't customized, then no additional actions are needed.

C. Appendix: Upgrading 8.8.1 to 8.9.1 (Windows)

1. Stop the Core service (and the Token Service if it is used).
2. Back up the old MedDream folder: rename/move C:\MDPACS\MedDream to C:\MDPACS\MedDreamOLD.
3. Make sure Java 17 is available on the system, and the command `java -version` indicates version 17. Even better is to always upgrade Java (17.x) when upgrading MedDream.
4. Extract new MedDream version to C:\MDPACS\MedDream so that the file C:\MDPACS\MedDream\MedDream-8.9.1.jar exists.

   Starting from 8.9.0, there is only a single service wrapper file, MedDream.exe (and token-service.exe, correspondingly), which requires .NET v4.

   In case your system still uses .NET v2, then please use your old version of the executable and consider upgrading the framework.
5. Copy the following old MedDream files to the new version:
   - meddream.lic
   - sys\settings*.json (except system.json and hangingProtocolsConfig.json)
   - sys\settings\robots.txt
6. Apply to MedDream.xml any changes made in the old file. Do not forget to remove -Djava.library.path=./lib from the command line.
7. If application.SAMPLE.properties was used during the initial installation, then copy the new file to application.properties and for every setting, copy the value from old application.properties.

   (During every upgrade, it is advised to keep the original application.SAMPLE.properties: later it can be compared with the new version and therefore save some time.)

   If the old file does not contain the same setting, then consult *Appendix: Configuration changes since 8.8.1* or the entire list 5.1.13 Reference for remaining configuration parameters. Probably it has been renamed, or is totally new.

   Warning: You might be tempted to simply copy the old application.properties to the new version and then review it. This is not recommended:
   - A higher risk of keeping settings not supported any more, which might lead to confusion later.
   - No opportunity to encounter default values (or totally new settings) in application.SAMPLE.properties that are safer security-wise, especially if overlooked.
   - No opportunity to encounter comments/explanations that might be more comprehensive in newer versions of application.SAMPLE.properties.
8. If, however, your application.properties is made from scratch instead, then consult *Appendix: Configuration changes since 8.8.1* or even the entire list 5.1.13 Reference for remaining configuration parameters.
9. After finishing with application.properties, make sure the file is readable only by the user who will be impersonated by the JVM process (not by the entire “Users” group).
10. If rebranding is used, then first choose a suitable ZIP file in sys\rebranding and extract its contents. For every setting in the new sys\rebranding\rebranding_configuration.json, copy the value from a setting in the old version.

    If the old JSON file does not contain the same setting, then consult *Rebranding*. It might be renamed or totally new.

Afterwards copy all referenced image files from the old sys\settings\rebranding folder to the new one.

11. If MedDream is running under some dedicated user, reapply the “Modify” permission for this user on C:\MDPACS\MedDream and all its contents.
12. Start the JAR so that sys\settings\system.json is created automatically. For every setting, copy the value from the old system.json. Consult the documentation (5.1.2 System.json) and list of configuration changes (*Appendix: Configuration changes since 8.8.1*).

> **Warning:** Unmodified system.json from older versions might be incompatible and result in a crash when starting the JAR.
>
> If you went the non-recommended way and are upgrading in-place, then old files sys/database/*.mv.db might result in errors while the JAR is starting. In case of errors related to H2 database engine, some of the files will be automatically renamed but the report database must be moved elsewhere manually so that new ones can be created.

13. Open the Settings window, review/adjust the configuration and save it. Without toolbarButtonsOrderingEnabled = false, the ordering of toolbar buttons and actions might differ from the saved one, and this new order will be forced until settings are manually reordered and saved by the new version.
14. Synchronize the newly created hangingProtocolsConfig.json with your old file. 8.9.0 brings 3 new study conditions that will be obvious when comparing the files. If your old file wasn’t customized, then no additional actions are needed.