Open Recon

(Version Apr 25th 2024)

Introduction

Open Recon provides a generalized interface for the deployment of advanced image reconstruction or analysis algorithms on the scanner with inline integration. Image reconstruction workflows are supported where raw k-space is transformed into images as well as image analysis workflows where images are segmented or otherwise processed. Algorithms can be implemented in any desired programming language and communications follow the standardized open-source ISMRM raw data (MRD) format.

Open Recon processing can be selected for any sequence or protocol through a customizable parameter card on the user interface. This interface is customizable for individual Open Recon applications and allows developer-defined parameters to be adjusted per acquisition for additional control of the processing pipeline.

Open Recon applications are run using the Siemens scanner's Measurement and Reconstruction System (MaRS) computer directly incl. GPU support, requiring no additional hardware to be installed and simplifying deployment. Docker containers are used to allow flexibility in libraries, while isolating Open Recon apps from the MR system and strict NUMARIS software dependency.

Finally and most importantly, the Open Recon interface is qualified to enable the development of image reconstruction or analysis algorithms from research to 3rd party product (software as a medical device - SaMD). **Note**: The creation of a medical product according to MDR and its regulatory pathway lies in the responsibility the of the 3rd party and cannot be supported by Siemens Healthineers.

Availability

All systems with Numaris XA60 and XA61 have the Open Recon option. For getting the Open Recon license the high-end (HE) MaRS is additionally required. An earlier version of Open Recon is available for Numaris XA51 system, but comes with a very limited feature set and is thus less recommended for new developments.

Current features and limitations

The Open Recon interface is intended to provide a flexible, but standardized solution to enable faster innovation cycles in research, clinical evaluation and productization. Thus, its feature set is consciously defined to enable a seemless integration and continuous deployment experience:

- Augmenting the reconstruction pipeline of arbitrary pulse sequences that support the standardized third party processing workflows (see Supported Workflows).
  Note: Since we aim for standardized workflows it is not intended to individually adapt data emitter / injector points within the ICE chain. Currently, it is not possible to process/share reconstruction data accross different pulse sequence acquisitions, nor to temporarily save data on a mapped fileshare.

Open Recon applications cannot be explicitly restricted to specific pulse sequence types or protocol parameters.

- Processing data with the open, standardized and scientifically acknowledged ISMRM raw data (MRD) format. In combination with container deployment a free choice of software stack and high re-use of community tools is feasible (see more details here).
  Note: Currently, the Open Recon MRD datastreaming is limited to raw k-space / image data, only.
- Seemless integration into MAGNETOM systems to enable custom user interface parameters and re-use of reconstruction hardware (MaRS).
  Note: Currently, offloading the reconstruction to external computing devices is not supported.
- For patient safety all DICOM images are labeled with "not for diagnostic use" for research containers as they are added to currently registered patient/volunteer.
- `Prio Recon` is currently not supported and must be de-activated. Please check under `Execution` tab within the protocol properties.

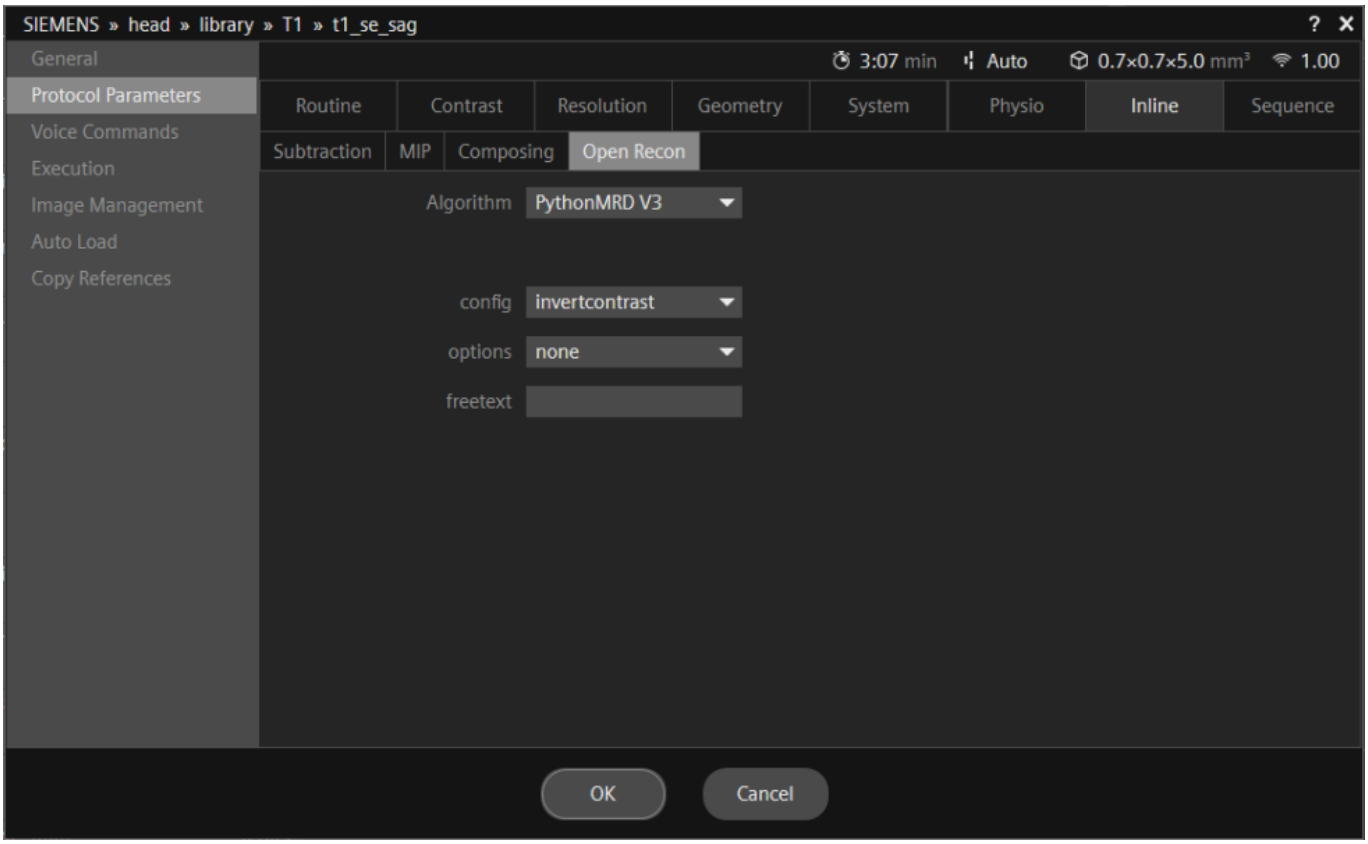


Figure: Example Open Recon user interface on the scanner console

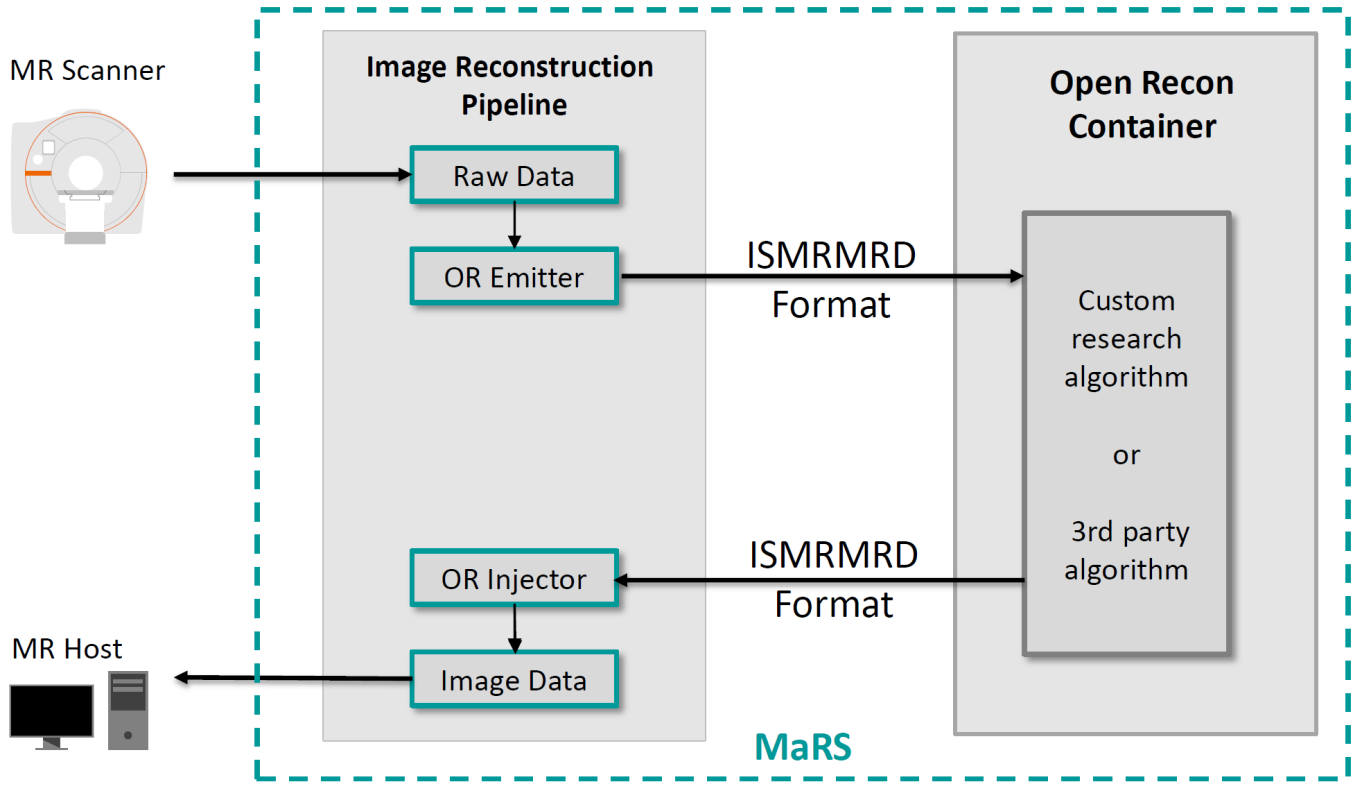


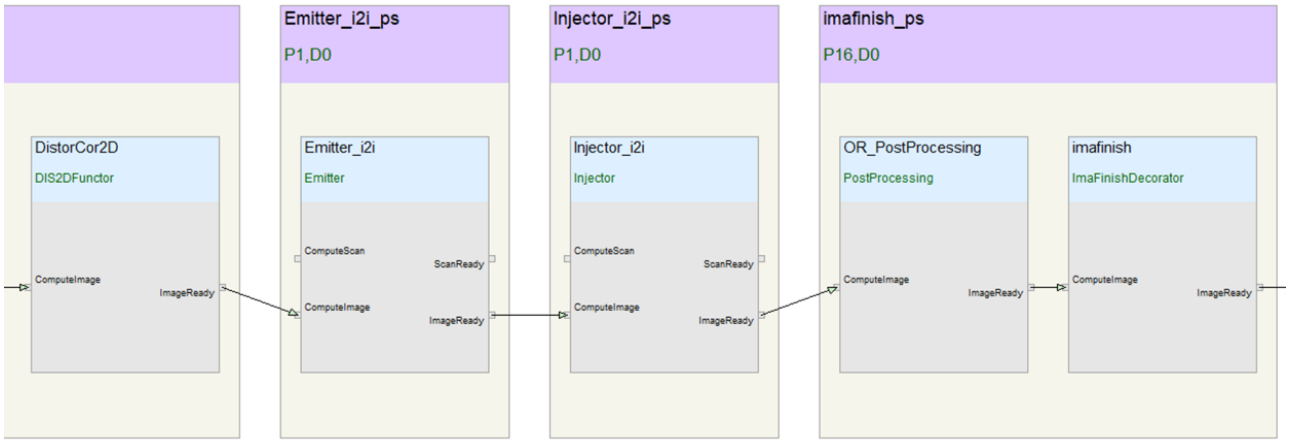
Figure: Schematic overview of interaction between Open Recon and standard product reconstruction in ICE

Supported workflows in Open Recon

Open Recon is designed to combine processing from both the standard Siemens Healthineers reconstruction pipeline (Ice Calculation Environment, ICE) and third party code. ICE is structured as a series of processing modules (functors) and Open Recon adds functors to send out (emit) data and receive (inject) data back into the pipeline. An Open Recon app therefore augments an existing reconstruction by performing additional processing at location of the emitter/injector.

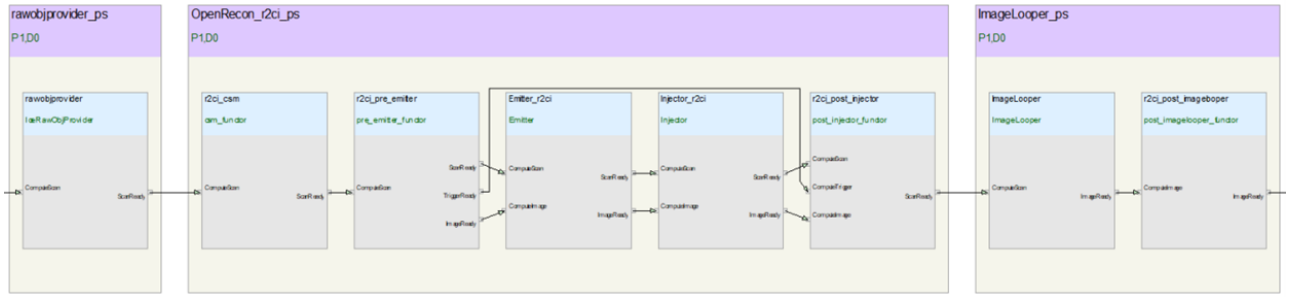
There are two supported workflows:

- **Image to image (i2i)**: Images are emitted near the end of the Siemens ICE pipeline, after distortion correction and pre-scan normalization (if selected). This workflow is similar to offline processing of DICOM images.



- **Raw to complex image (r2ci)**: Raw k-space data is emitted after noise pre-whitening and complex image-space data is injected. The Open Recon app therefore replaces the Fourier transform step. The injected images undergo any additional processing as specified in the normal reconstruction pipeline,

e.g. phase difference and Maxwell correction for phase contrast protocols.



In both workflows, only images that are returned by the Open Recon app are saved to DICOMs and displayed on the scanner. The standard Siemens reconstructed images are not automatically saved.

Mandatory MRD message types and header attributes

Open Recon requires some mandatory MRD streaming messages and header entries to ensure a valid image reconstruction.

MRD message types

Please consider the following mandatory MRD Message Types (for more details see also here). All other incomming MRD message types will be treated as an error, and will result in an abortion of the reconstruction.

- **Image to image (i2i)**:
  - `MRD_MESSAGE_IMAGE(1022)`
  - `MRD_MESSAGE_TEXT(5)`
  - `MRD_MESSAGE_CLOSE(4)`
- ***Raw to complex image (r2ci)**:
  - `MRD_MESSAGE_IMAGE(1022)`
  - `MRD_MESSAGE_TEXT(5)`
  - `MRD_MESSAGE_CLOSE(4)`

Regarding incomming `MRD_MESSAGE_TEXT(5)`: Incomming `MRD_MESSAGE_TEXT(5)` messages will be accepted, and its content will be logged. The content of such message will not be further processed.

Regarding incomming `MRD_MESSAGE_CLOSE(4)`: It is expected that this message is only received when the `MRD_MESSAGE_CLOSE(4)` message has already been sent out by Open Recon to indicate that no futher outgoing data will be sent. It is considered an error if this message is received prior sending it out.

Regarding incomming `MRD_MESSAGE_IMAGE(1022)`: Accepted data types are `MRD_USHORT`, `MRD_SHORT`, `MRD_FLOAT`, `MRD_CXFLOAT` (see also here).

MRD header MetaAttributes

The following MRD image header MetaAttributes are required to generate a valid image and finally DICOM by Open Recon:

- `field_of_view`: all 3 dimensions must not be zero
- `ImageRowDir`: must be set for valid image orientation
- `ImageColumnDir`: must be set for valid image orientation
- `Keep_image_geometry`: must be set to 1, as otherwise all images will get flipped

Dimension indices and boundaries can be specified by setting the according MetaAttribute flags. This boundary properties will be provided for each Image in its MetaAttributes, and are also expected as part of the Image MetaAttributes received by Open Recon. All indices must be within (<) the set boundaries. Invalid indices will result into abortion of the reconstruction.

- `slice_count`
- `contrast_count`
- `set_count`
- `phase_count`
- `average_count`
- `partition_count`
- `repetition_count`
- `ida_count`
- `idb_count`
- `idc_count`
- `idd_count`
- `ide_count`

Note: Unfortunately, the ImageHeader MetaAttribute `ROI` cannot be used. Providing the ROI MetaAttribute will result in an abortion of the reconstruction.

Selecting a development environment

Open Recon communicates using the ISMRM raw data (MRD) format and supports any programming environment that adheres to this standard. There are several development environments that are commonly used, e.g.:

- Python is a widely used programming language for data science and machine learning which is open-source and free, including commercial use. Many image processing algorithms and packages are available for Python, although individual packages may be differently licensed. A Python-based MRD server is provided at https://github.com/kspaceKelvin/python-ismrmrd-server and is distributed as an example for Open Recon. Both image reconstruction and image analysis workflows are supported. PyTorch and TensorFlow integration is supported in Open Recon through this repository. Examples in this guide will primarily use the Python server due to its simplicity, open-source license, and easy modification without recompilation.

- MATLAB is a commonly used development environment for image analysis and visualization. It requires a paid license but can be compiled into a standalone executable and packaged into a Docker image such that it can be run on the scanner without an active network connection. A MATLAB-based MRD server designed similarly to the Python example is available at https://github.com/kspaceKelvin/matlab-ismrmrd-server and provides documentation and examples for compilation into a Docker image.

- Gadgetron is a C++ based open-source framework designed for image reconstruction. It contains a large library of reconstruction algorithms and pioneered the MRD streaming format. It is also expandable using Python and MATLAB based “gadgets”.

- The Berkeley Advanced Reconstruction Toolbox (BART) is a C-based open-source framework that contains many examples for MR image reconstruction and analysis . BART has both Python and

MATLAB interfaces and is supported in Open Recon with the Python server (https://github.com/kspaceKelvin/python-ismrmrd-server/blob/master/bartfire.py).

Getting started with Open Recon modules and python-ismrmrd-server

This repository contains a simple Getting Started description and two Open Recon Python examples utilizing the python-ismrmrd-server, i.e. `i2i.py` and `r2ci.py`, corresponding to the "image to image" and "raw to complex image" workflows respectively.

The python-ismrmrd-server repository is organized as a set of independent modules corresponding to the MRD config selected. In Open Recon, the config is always sent as `OpenRecon`. If a corresponding `OpenRecon.py` file exists, then this module will be executed when data is sent. If not, the Python server will call it's default config instead. The default config can be changed via command line when starting the server using the `--defaultConfig` option. When creating a Docker image for an Open Recon app, this can be configured in the `CMD` line of the Dockerfile.

Alternatively, the Python config can be overridden using an appropriately named parameter in the Open Recon UI. A string or choice field named `config` or `customConfig` will be parsed by the Python server and the appropriate module will be used during run-time. Two field names are provided: one (e.g. `config`) can be used as an enumerated list of valid configs, while `customConfig` can be used as an arbitrary string for additional user-defined overrides to the enumerated list.

Open Recon application specification

An Open Recon app is specified using JSON-formatted text stored in the Docker image's metadata. This config specifies how the app interfaces with the product reconstruction, the end user interface presented on the scanner, and general information about the app.

The Open Recon JSON config should be validated against OpenReconSchema_1.1.0.json prior to packaging for deployment using a tool like jsonschema. The JSON text is base64-encoded and stored as a label of the Docker image under `com.siemens-healthineers.magneticresonance.OpenRecon.metadata:1.1.0`.

Further details about the Open Recon JSON config can be found in OpenReconJsonConfig.md. The CreateORDockerImage.ipynb Python notebook contains example code for JSON validation and base64-encoding the text.

Packaging an Open Recon application

Open Recon apps are deployed to the scanner as a single .zip file that contains both the Docker image (.tar) and its documentation (.pdf). The .tar file of the Docker image must be created using `docker save`. Note that this is different than `docker export`, which exports only the file contents of the Docker image. The documentation must be in Adobe PDF format. **Note**: Please make sure that you are using a compatible Docker version - you can use 'checkDockerVersion.py' for checking.

Both .tar and .pdf files must have the same name, follow the format `OpenRecon_<Vendor>_<Name>_V<version>`, and be consistent with the information in the JSON config. The files should be archived into a single .zip file, also with the same base file name as above.

Note: The zip file must be created using the Deflate compression algorithm instead of Deflate64. If using Windows, right-clicking on files and selecting "Send to -> Compressed (zipped) folder" will use Deflate64 for a

total file size >2GB and will *thus be incompatible*. Creating a zip file with 7-Zip will use the Deflate algorithm for all file sizes.

The CreateORDockerImage.ipynb Python notebook also contains a script for naming the files based on the JSON config and creating a .zip file using 7-Zip.

Deployment on the scanner

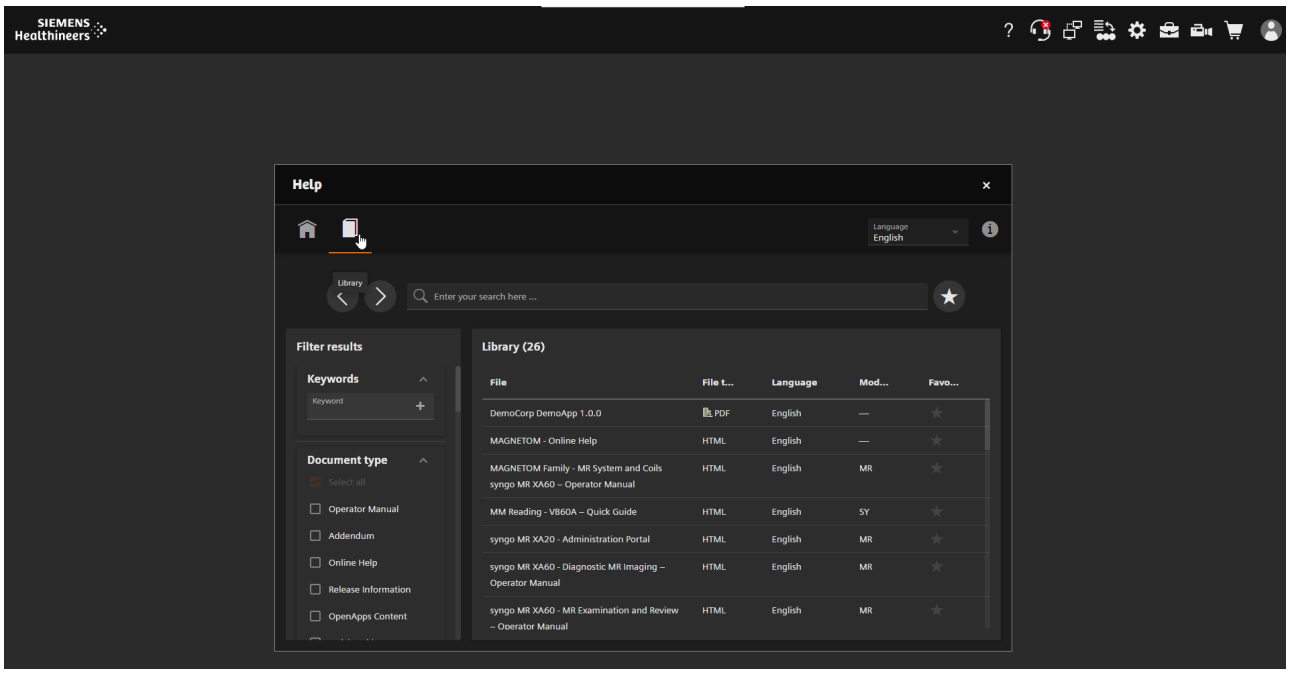
1. On the scanner console, exit kiosk mode by pressing `Tab + Del + NumPadPlus` and enter the required credentials.

2. Copy the OpenRecon.zip file into the folder `C:\Program Files\Siemens\Numaris\OperationalManagement\FileTransfer\incoming`.
   Note: Copying to this folder requires escalated privileges (same as for kiosk mode). It may be necessary to copy to another folder (e.g. Downloads) first and then move into the `incoming` folder.

3. Wait for 2-3 minutes for the file to be processed. The .zip file will disappear if it is successfully imported.

   - If the .zip file remains after 5 minutes, look for messages in the Message Viewer. Click on "gear" icon on the top right of the screen, then select "Administration Portal" and then "Message Viewer". Filter the message text for "OpenRecon".
   - A log file is created in `C:\ProgramData\Siemens\Numaris\log\syngo.MR.HostInfra.OpenRecon.Watcher\` for each installed app.

4. The PDF documentation from Open Recon Apps can be accessed by clicking on the "?" icon on the top right of the screen and selecting "Help". Then click on the book icon to access the Library. Double-click on the name of the App in the File column to open the PDF.



5. Open the myExam Cockpit and open a program in edit mode. Go to the Inline -> Open Recon card and select the Open Recon app from the drop down menu.

    - If the Open Recon app does not appear in the menu, open the file `C:\ProgramData\Siemens\Numaris\log\syngo.MR.Exam.Appl.utr` in Logviewer and look for errors.

6. Runtime messages including std::cout messages from the Docker container are logged to `C:\ProgramData\Siemens\Numaris\log\OpenRecon.utr`, which can be parsed by Logviewer.

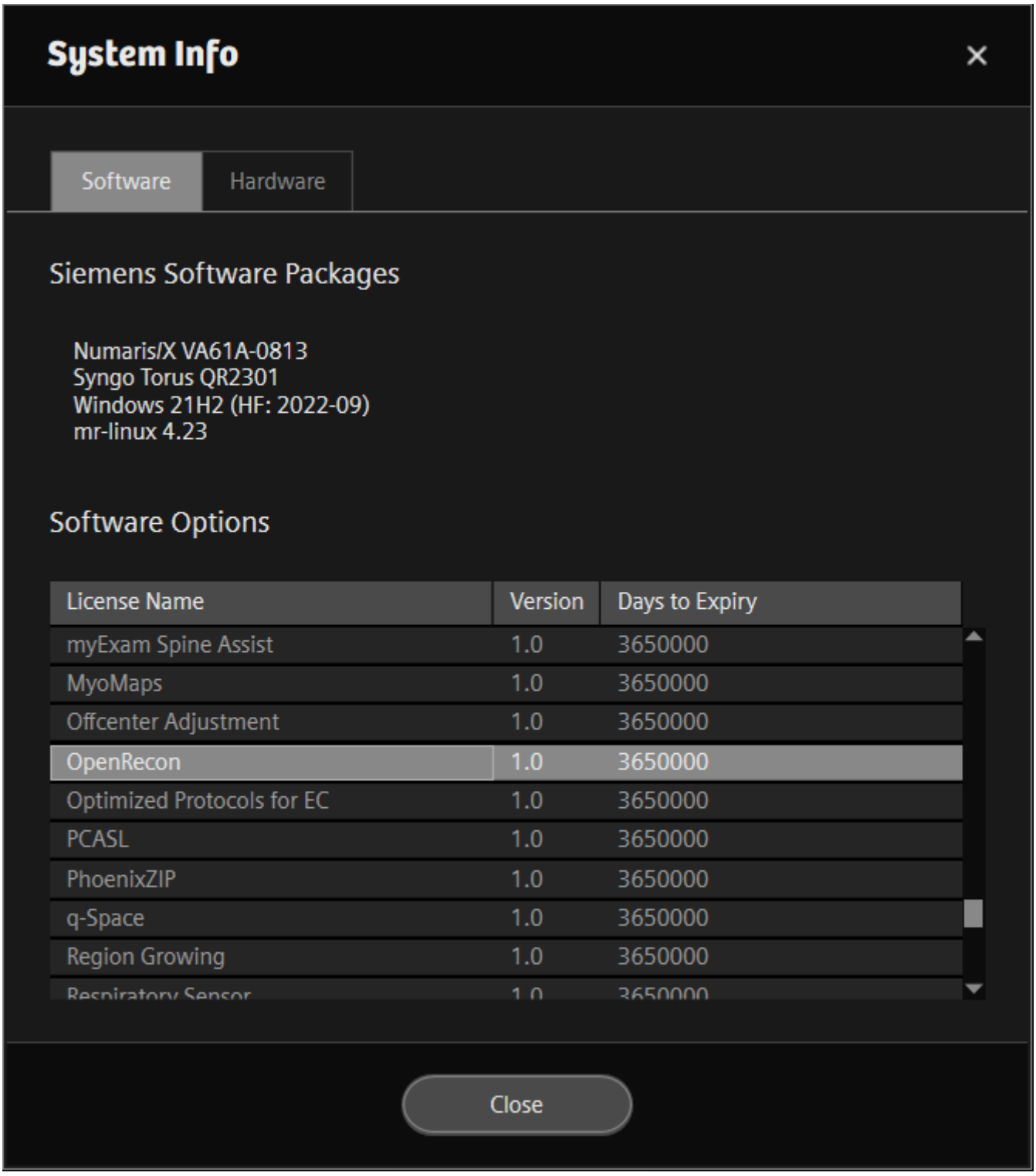
Reset all installed Open Recon apps on the scanner

Open Recon apps cannot be individually uninstalled. However, all Open Recon apps installed on a scanner can be removed using the RemoveInstalledOpenReconApps.bat script. This must be run with administrative privileges and a "Restart Workspace" or scanner reboot must be performed afterward.

Troubleshooting

Open Recon does not appear in the Inline card or the Inline card is missing

The Inline card is hidden if there are no sub-cards (e.g. Open Recon) activated. The Open Recon card requires that the Open Recon product license be installed on the system. The Open Recon license is not included by default and must be separately requested for each system. A list of installed product licenses can be found by clicking on the "?" icon at the top right, selecting "About", then "System Info":



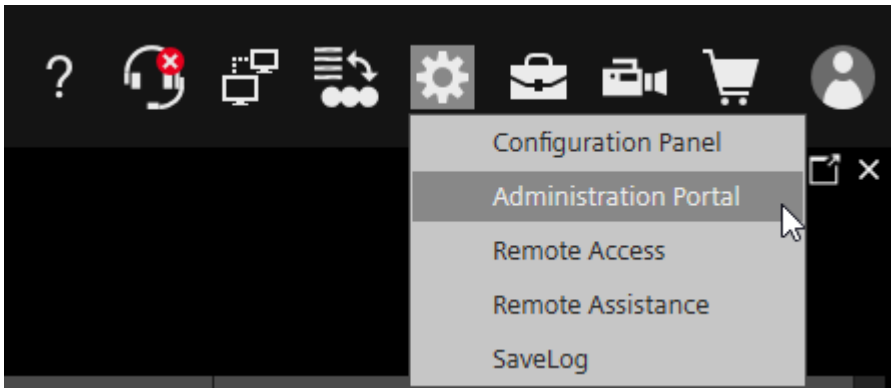
Open Recon zip file remains in `incoming` folder

The `C:\Program Files\Siemens\Numaris\OperationalManagement\FileTransfer\incoming` folder is continually monitored for .zip files and these files are automatically removed upon successful installation. If a .zip file remains in this folder for >3 minutes after it is copied there, then the app may have been rejected. Common failures are:

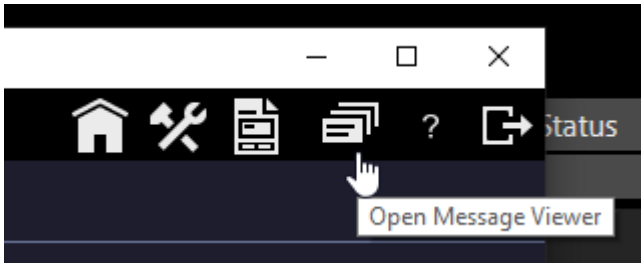
- Incorrect file naming. Files must be named exactly as `OpenRecon_<Vendor>_<Name>_V<version>`, e.g. `OpenRecon_DemoCorp_DemoApp_V1.0.0`, for the .zip file, as well as the .tar and .pdf files within.
- Incorrect zip format. The Deflate compression algorithm instead of Deflate64 must be used. Retry .zip creation with 7-Zip.

Logging messages for this stage are found in the Message Viewer:

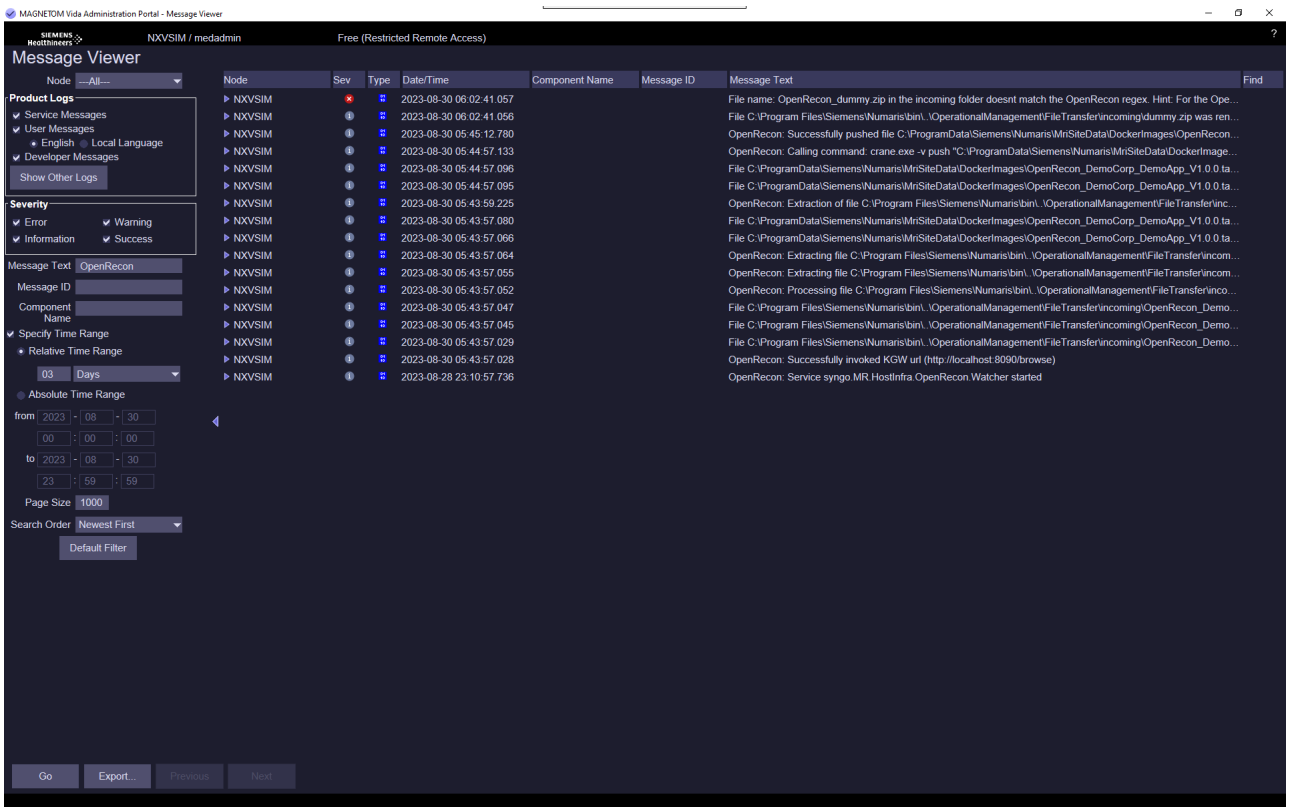
1. Click the gear icon on the top right of the screen, then Administrator Portal.



2. Log in with the medadmin account, and select Message Viewer.



3. Enter the keyword "Open Recon" in the "Message Text" section on the left and click on "Go" on the bottom left.



Open Recon app does not appear in Open Recon "Algorithm" drop-down menu

The Open Recon JSON config is parsed from the Docker image during installation. A parsed copy of all installed apps on the scanner is stored in `C:\ProgramData\Siemens\Numaris\MriSiteData\Scanner\OpenReconStore.json`. This file can be edited directly and changes will take effect after the parameters card for a sequence is closed and reopened. Note that `OpenReconStore.json` is regenerated every time an app is installed, so editing this file is should be considered temporary for debugging purposes.

If the config does not conform to OpenReconSchema_1.1.0.json, then the app will not appear in the UI. Logging information for parsing of the JSON config can be found in `C:\ProgramData\Siemens\Numaris\log\syngo.MR.Exam.Appl.utr`, which can be parsed by Logviewer.

Important files/folders

- `C:\Program Files\Siemens\Numaris\OperationalManagement\FileTransfer\incoming\`
  - Installation folder for Open Recon app .zip files
- `C:\ProgramData\Siemens\Numaris\log\syngo.MR.HostInfra.OpenRecon.Watcher\`
  - Installation logs from parsing Open Recon app .zip files (one per app)
- `C:\ProgramData\Siemens\Numaris\MriSiteData\Scanner\OpenReconStore.json`
  - Cache file for configurations of all installed Open Recon apps
- `C:\ProgramData\Siemens\Numaris\log\syngo.MR.Exam.Appl.utr`
  - Log file for parsing OpenReconStore.json into the UI
- `C:\ProgramData\Siemens\Numaris\log\OpenRecon.utr`
  - Main log file for Open Recon, including output from the Docker container itself
- `C:\ProgramData\Siemens\Numaris\MriSiteData\DockerImages\`
  - Location of extracted/processed Docker images. Note: These files should not be directly manipulated!
- `C:\ProgramData\Siemens\Numaris\MriSiteData\DockerRegistry\`
  - Location of actual Docker registry used by crane. Note: These files should not be directly manipulated!

Disclaimer

Open Recon is to add clinical reconstructions to the system, if signed and released for clinical use by Siemens Healthineers. Any other recon used e.g., by researchers is automatically labelled not for diagnostic use, which may require observation of national regulations.