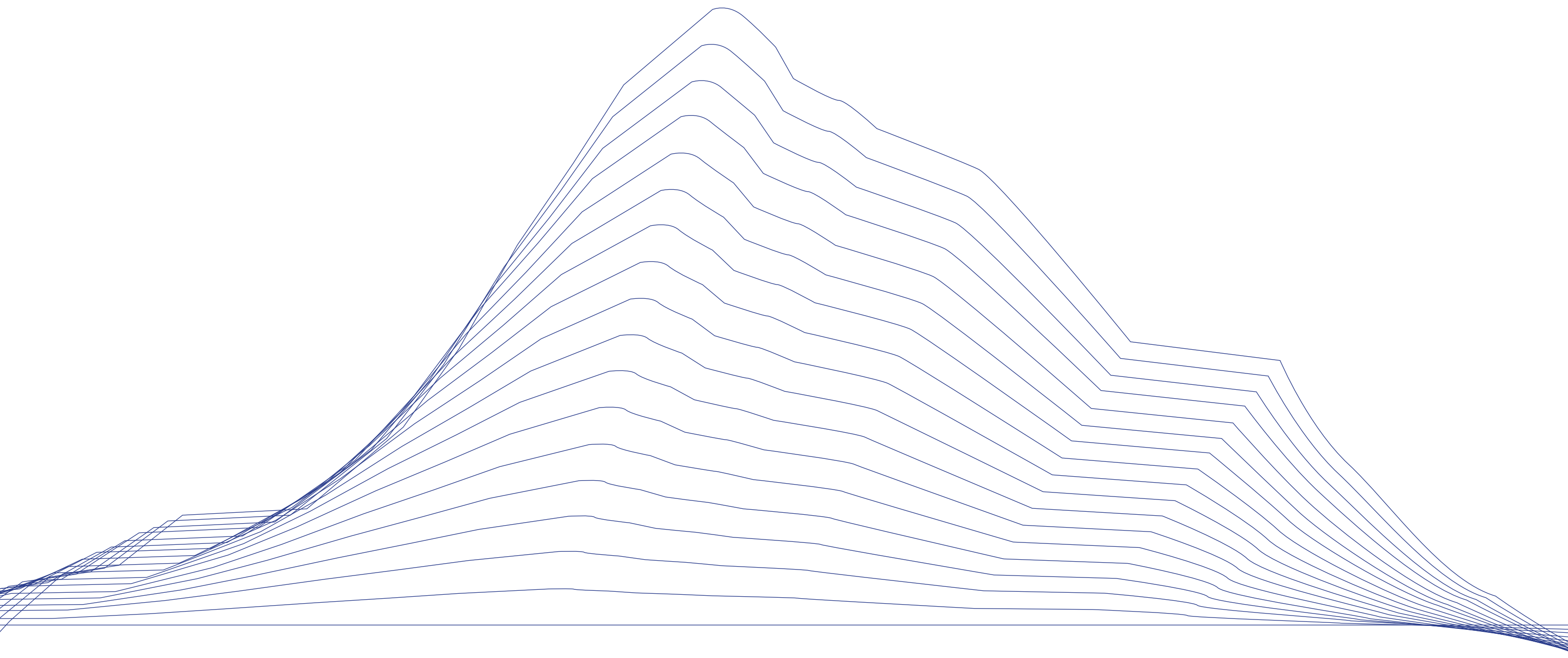

User Manual

EPU

Document Version 2.0

DECTRIS Ltd.

5405 Baden-Daettwil, Switzerland

© Copyright 2026

www.dectris.com

TABLE OF CONTENTS

1. GENERAL INFORMATION

1.1. Contact and Support

Address	**DECTRIS Ltd** Taefernweg 1 5405 Baden-Daettwil Switzerland	**DECTRIS USA Inc.** 1500 Walnut Street, Suite 1630 Philadelphia, PA 19102 USA	**DECTRIS Japan K.K.** Nakagawa building 801 3-37 Higashi-Nobusue Himeji-shi Hyogo 670-0965 Japan
Phone	+41 56 500 21 02	+1 267 924 5179	+81 79 280 9585
Website	www.dectris.com		
Email	support@dectris.com		

1.2. Warranty Information

CAUTION
Do not ship the system back before you receive the necessary transport and shipping information.

1.3. Disclaimer

DECTRIS® has carefully compiled the contents of this manual according to the current state of knowledge. Damage and warranty claims arising from missing or incorrect data are excluded.

DECTRIS® bears no responsibility or liability for damage of any kind, also for indirect or consequential damage resulting from the use of this system.

DECTRIS® is the sole owner of all user rights related to the contents of the manual (in particular information, images or materials), unless otherwise indicated. Without the written permission of DECTRIS® it is prohibited to integrate the protected contents in this publication into other programs or other websites or to use them by any other means.

DECTRIS® reserves the right, at its own discretion and without liability or prior notice, to modify and/or discontinue this publication in whole or in part at any time, and is not obliged to update the contents of the manual.

2. EIGER PROCESSING UNIT (EPU)

The EIGER Processing Unit (EPU) efficiently complements EIGER detector systems at high demand experimental stations. It consists of a high-end server providing substantial processing power and dedicated software packages eliminating bottlenecks in many of today's experiment installations.

EPU key advantages are:

- Highly reliable and performant data transfer.
- Fast local storage for several weeks of regular user operation.
- Fast data processing using XDS third-party-software

2.1. Pre-installed Software Packages

The EPU comes with Rocky 9.x pre-installed.

The default password for the normal user epu is set to EIGER_PU

The default password for the superuser root is set to !<ServiceTag>!

- The service tag is also the server name

In addition to the operating system, the following software is preinstalled:

- GRIMSEL2
  - copies data from the ramdisk of the detector server to the ramdisk of the EPU
  - copies data from the ramdisk to disk
- XDS (crystallographic data reduction package
  - http://xds.mpimf-heidelberg.mpg.de
- Neggia
- ALBULA viewer

2.2. Third-Party Software

The third-party software XDS is proposed for crystallographic data processing. The architecture of the EPU allows taking advantage of the multi-threading capabilities of XDS. Keeping all data and computations in ramdisk eliminates the bottleneck of disk I/O and results in superior data processing performance.

XDS comes pre-installed and ready to use. The XDS binaries are located in the directory /var/lib/xds

For full XDS documentation and literature please refer to http://xds.mpimf-heidelberg.mpg.de/

Please note: XDS is free of charge for non-commercial applications. For industrial usage of XDS a license is required (e-mail enquiry: Wolfgang.Kabsch@mpimf-heidelberg.mpg.de).

2.3. Data Processing with the EPU

To process with XDS, create XDS.INP with the metadata stored in the master file.

If you do not have a mechanism to extract the metadata at your beamline, use one of the tools suggested in the XDS wiki:

http://strucbio.biologie.uni-konstanz.de/xdswiki/index.php/Eiger

For fastest processing, use the Neggia plugin and enable parallel execution of XDS:

https://github.com/dectris/neggia

Adjust XDS.INP as follows:

- Point LIB= to where the Neggia plugin is saved. For example, LIB= /usr/local/lib64/dectris-neggia.so
- Set MAXIMUM_NUMBER_OF_JOBS= and MAXIMUM_NUMBER_OF_PROCESSORS= to values similar to each other, with MAXIMUM_NUMBER_OF_PROCESSORS= larger and the product of the two numbers slightly smaller than the total number of threads you want to use.

We recommend MAXIMUM_NUMBER_OF_JOBS= 8 and MAXIMUM_NUMBER_OF_PROCESSORS= 18 if only one XDS job is run and data are transferred at the same time. This will use 144 threads for processing, leaving 16 threads for copying and the system.

Space available on the ramdisk of the DCU should be monitored using the following command:

```
curl http://<IP of DCU>/filewriter/api/1.8.0/status/buffer_free
```

3. GRIMSEL2

GRIMSEL2 is capable of synchronizing data between several storages. The data flow is illustrated in Figure 1.

GRIMSEL2 copies data from the detector server to the EPU ramdisk and copies the same data from the ramdisk to the EPU storage (internal storage) and optionally to an external storage.

A detailed description of GRIMSEL2 is provided in the following sections.

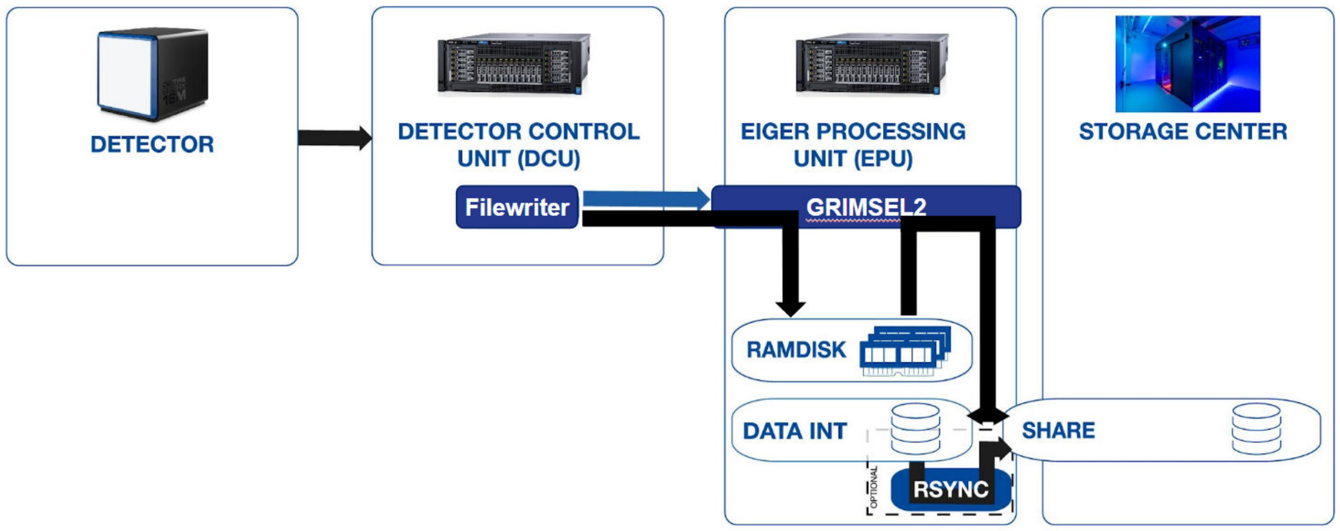

Figure 1. Data flow of GRIMSEL2

3.1. Running GRIMSEL2

GRIMSEL2 permanently copies image data written by the DCU and writes them into the target directory (in the following called /mnt/data_buffer), to the local EPU disk and optionally to external storages. Note, any additional data created on /mnt/data_buffer is only copied if it matches the filter in /etc/GRIMSEL2.toml

The uid, gid, and timestamp of the copied files are preserved.

GRIMSEL2 removes files from /mnt/data_buffer whenever a user change takes place, i.e. whenever the GRIMSEL2 service is restarted.

GRIMSEL2 will delete all files belonging to the old user as soon as they have been copied to the local disk on the EPU. The idea behind that is, whenever a user change takes place, the data of the former user is no longer needed on /mnt/data_buffer. This particular behavior of GRIMSEL2 makes it necessary to regularly restart GRIMSEL2 after each user change to prevent /mnt/data_buffer from overflowing. When GRIMSEL2 notices /mnt/data_buffer (or the directory, to which the variable ramdisk_mountpoint in the configuration file is set to) is likely to overflow, it converts files on /mnt/data_buffer into symbolic links to the respective files on the disk.

It is strongly recommended to restart GRIMSEL2 after every user change or at least after each beamline shift. This can be done either explicitly by the command `sudo systemctl restart grimsel2` or by using a different uid/gid. To change the uid/gid, the configuration file (/etc/grimsel2.toml) needs to be adapted and the service restarted.

Exactly one internal storage device must be configured to run GRIMSEL2. By default, the local EPU disk is used as internal storage. In addition, at most one external storage and several optional storages may be set up in the configuration file (see Section 3.3).

If an external storage is defined, the internal EPU storage is cleared after a user change, and files on the internal storage are, in case of overflow, substituted for symbolic links to the external storage. Without an external storage, it is the beamline administrator's responsibility to keep the EPU storage clean.

All storages must be mounted locally on the EPU. GRIMSEL2 assumes that the storages are always available. This might be an issue when working with removable USB drives.

If multiple storages are defined, the slower storage system will limit the speed with which GRIMSEL2 copies data from the RAM disk. Therefore, if the storages have significantly different write speeds, it is advisable to define the faster storage as internal storage and sync data from this faster storage to all slower storages.

Rsync provides a simple solution to sync the data from one storage to another:

```
rsync –avz source_storage dest_storage
```

3.2. Start and stop GRIMSEL2

Control GRIMSEL2 via the command line (administrator privileges needed):

```
systemctl start|stop|restart|status grimsel2.service
```

3.3. GRIMSEL2 Configuration

The following parameters can be set globally in the configuration file of GRIMSEL2 /etc/grimsel2.toml:

- source_host: webdav server hostname and optionally port, e.g. dcu16:1234
- source_path: webdav server path to copy from
- staging_dir: staging storage (ramdisk) mount point, mandatory
- ram_only: allows for files to only be copied from the DCU to the EPU RAM. Files are not removed from *staging_dir*
- internal_dir: internal storage mount point, mandatory
- external_dir: external storage mount point, optional
- log_level: log level debug, info, warn or error
- check_mountpoints: if true, check that directories are on separate mounts and not on the root filesystem
- delete: delete files from source after copy. Do not change the default except for debugging!
- filter: regexp to filter files ((?i) makes it case insensitive)
- uid: uid to set on copied files, use 0 or undefined to keep original
- gid: gid to set on copied files, use 0 or undefined to keep original
- num_procs_copy: number of concurrent processes to use for copy
- num_procs_archive: number of concurrent processes to use for archive
- staging_limit: link files to internal dir when staging dir is filled to this fraction
- staging_limit_trim: trim files down to this fraction of the staging_limit
- internal_limit: move files to external until internal dir is filled to this fraction
- internal_limit_trim: trim files down to this fraction of the internal_limit

3.4. Logs

The log files are handled by journalctl and can be viewed likes this (simple example):

- `journalctl -u grimsel2.service -f`
  - –u: service name / user
  - –f: follow

For more information please refer to the journalctl Linux manual page