
TomoStream Documentation

Release 0.1

Argonne National Laboratory

Dec 21, 2022

CONTENTS

1	Content	1
	Bibliography	11

CONTENT

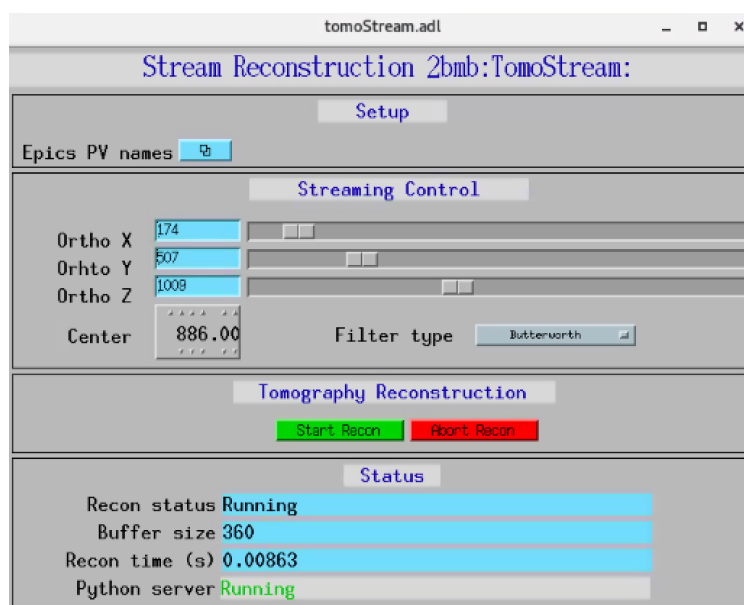
1.1 About

tomostream is Python module for supporting streaming analysis of tomographic data where all pre-processing and reconstruction procedures are performed in real time while images are collected and the rotary stage is moving. **tomostream** provides this main functionality:

- Streaming reconstruction of 3 X-Y-Z ortho-slices through the sample

The streaming reconstruction engine generates 3 selectable X-Y-Z orthogonal planes and makes them available as an EPICS PV viewable in ImageJ using the [EPICS_NTNDViewer](#) plug-in. Projection, dark and flat images used for the reconstruction are taken in real time from a set of PV access variables (pvapy) and stored in a synchronized queue. On each reconstruction call new data are taken from the queue, copied to a circular GPU buffer containing projections for a 180 degrees interval, and then reconstructed.

All **tomostream** functionalities can be controlled from the tomoStream user interface:

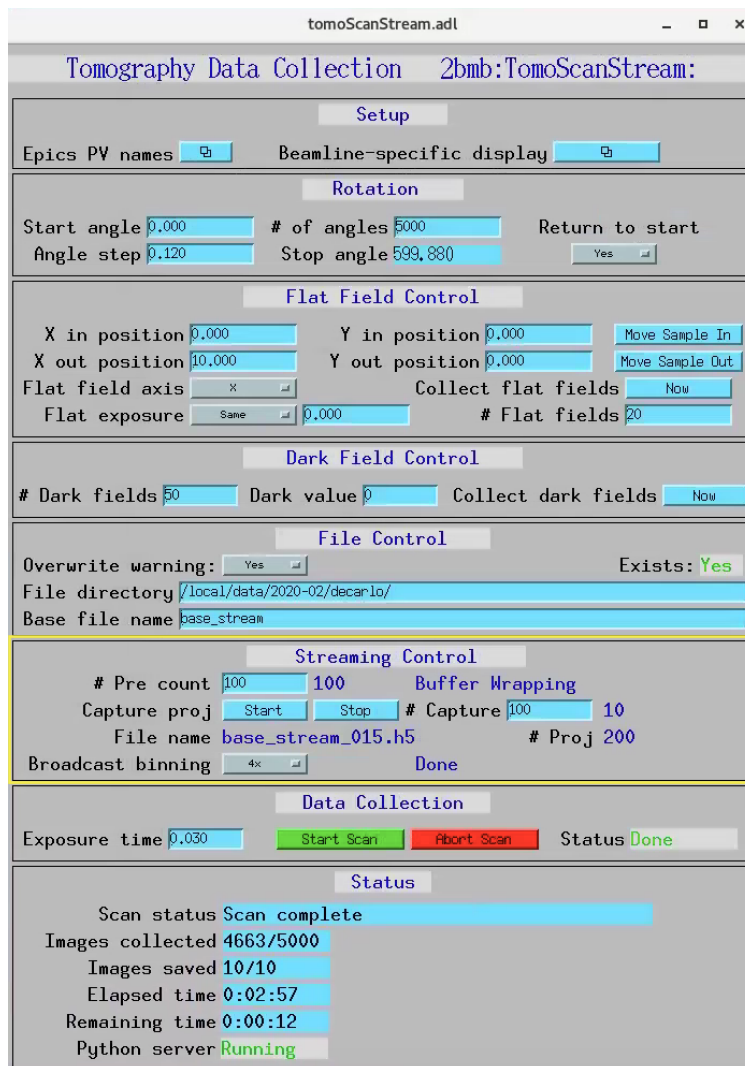


tomostream relies on [tomoscan_stream_2bm](#) (part of tomoScan) for:

- Tomography instrument control
- Projection, dark and flat image broadcast as PV access variables
- On-demand retake of dark-flat field images

- On-demand data capturing with saving in a standard hdf5 [DXfile](#) file
- Set a number of projections (“Pre count”) collected before a triggered data capturing event to be also saved in the same hdf5 file

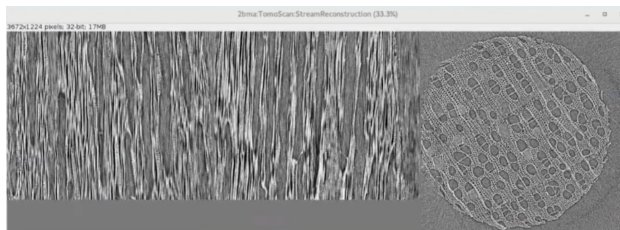
All `tomoscan_stream_2bm` functionalities supporting **tomostream** can be controlled from the tomoScanStream user interface marked in yellow:



The screenshot shows the Tomography Data Collection 2bmb:TomoScanStream: user interface. The interface is divided into several sections:

- Setup**: Epics PV names, Beamline-specific display.
- Rotation**: Start angle (0.000), # of angles (5000), Return to start, Angle step (0.120), Stop angle (599.880), Yes.
- Flat Field Control**: X in position (0.000), Y in position (0.000), Move Sample In, X out position (10.000), Y out position (0.000), Move Sample Out, Flat field axis (X), Collect flat fields (Now), Flat exposure (Save), 0.000, # Flat fields (20).
- Dark Field Control**: # Dark fields (50), Dark value (0), Collect dark fields (Now).
- File Control**: Overwrite warning: Yes, Exists: Yes, File directory (/local/data/2020-02/decarlo/), Base file name (base_stream).
- Streaming Control** (highlighted in yellow): # Pre count (100), 100, Buffer Wrapping, Capture proj (Start/Stop), # Capture (100), 10, File name (base_stream_015.h5), # Proj (200), Broadcast binning (4x), Done.
- Data Collection**: Exposure time (0.030), Start Scan, Abort Scan, Status Done.
- Status**: Scan status Scan complete, Images collected 4663/5000, Images saved 10/10, Elapsed time 0:02:57, Remaining time 0:00:12, Python server Running.

The output of **tomostream** is a live reconstruction displaying in ImageJ using the [EPICS_NTNDViewer](#) plug-in:



While the sample is rotating is possible to optimize instrument (alignment, focus, sample to detector distance etc.) and beamline (energy etc.) conditions and monitor the effect live on the 3 orthogonal slices. It is also possible to automatically trigger data capturing based on events occurring in the sample and its environment as a result of segmentation or machine learning.

1.2 Install directions

The computer performing the tomographic reconstruction must have CUDA/GPU installed. **tomostream** consists of two modules TomoScanApp and tomostream tools.

1.2.1 TomoScanApp

Provides all the EPICS PVs needed by **tomostream**. To install TomoScanApp follow these steps:

Build a minimal synApps

To build a minimal synApp:

```
$ mkdir ~/epics
$ cd epics
```

- Download in ~/epics [assemble_synApps.sh](#)
- **Edit the assemble_synApps.sh script as follows:**
 - Set FULL_CLONE=True
 - Set EPICS_BASE to point to the location of EPICS base. This could be on APSshare (the default), or a local version you built.
 - For tomostream you only need BUSY and AUTOSAVE. You can comment out all of the other modules (ALLENBRADLEY, ALIVE, etc.)

- Run:

```
$ assemble_synApps.sh
```

- This will create a synApps/ directory:

```
$ cd synApps/support/
```

- Edit busy-R1-7-2/configure/RELEASE to comment out this line:

```
ASYN=$(SUPPORT)/asyn-4-32) .
```

- Clone the tomostream module into synApps/support:

```
$ git clone https://github.com/tomography/tomostream.git
```

- Edit tomostream/configure/RELEASE to comment out this line:

```
ASYN=$(SUPPORT)/asyn-4-38
```

- Edit tomostream/tomoStreamApp/src/Makefile to comment out this line:

```
tomoStreamApp_LIBS += asyn
```

- Edit configure/RELEASE add this line to the end:

```
TOMOSTREAM=$(SUPPORT)/tomostream
```

- Edit Makefile add this line to the end of the MODULE_LIST:

```
MODULE_LIST += TOMOSTREAM
```

- Run the following commands:

```
$ make release
$ make -sj
```

Testing the installation

- Edit `/epics/synApps/support/tomostream/configure`
 - Set EPICS_BASE to point to the location of EPICS base:
 - EPICS_BASE=/APSshare/epics/base-3.15.6
- Start the epics ioc and associated medm screen with:

```
$ cd ~/epics/synApps/support/tomostream/iocBoot/iocTomoStream
$ start_IOC
$ start_medm
```

1.2.2 tomostream python tools

```
$ cd ~/epics/synApps/support/tomostream/
$ python setup.py install
```

Testing the installation

```
$ cd ~/epics/synApps/support/tomostream/iocBoot/iocTomoStream
$ python -i start_tomostream.py
```

1.3 Usage

1.3.1 Using the tomoStream

Pre-requisites

Before running **tomostream** you need to install and run `tomoscan_stream_2bm` (see `tomoScan` for details) to provide:

- Tomography instrument control
- Projection, dark and flat image broadcast as PV access variables
- On-demand retake of dark-flat field images
- On-demand data capturing

Once `tomoScan` is installed on the computer connected to the detector:

- start area detector, e.g.:


```
user2bmb@lyra$ 2bmbPG1 start
```

- start tomoScan IOC, e.g.:

```
user2bmb@lyra$ cd /local/user2bmb/epics/synApps/support/tomoscan/iocBoot/
↪ iocTomoScan_2BM/
user2bmb@lyra$ ./start_IOC
```

- start the instance of tomoscan.py supporting tomostream tasks at your beamline, e.g.:

```
user2bmb@lyra$ cd /local/user2bmb/epics/synApps/support/tomoscan/iocBoot/
↪ iocTomoScan_2BM/
user2bmb@lyra$ python -i start_tomoscan_stream.py
```

- start tomoScan user interface, e.g.:

```
user2bmb@lyra$ cd /local/tomo/epics/synApps/support/tomostream/iocBoot/
↪ iocTomoStream/
user2bmb@lyra$ ./start_medm
```

The screenshot shows the 'tomoScanStream.adl' window with the title 'Tomography Data Collection 2bmb:TomoScanStream:'. The interface is divided into several sections:

- Setup:** Includes 'Epics PV names' and 'Beamline-specific display' buttons.
- Rotation:** Contains input fields for 'Start angle' (0.000), '# of angles' (5000), 'Return to start' (Yes), 'Angle step' (0.120), and 'Stop angle' (599.880).
- Flat Field Control:** Includes 'X in position' (0.000), 'Y in position' (0.000), 'X out position' (10.000), 'Y out position' (0.000), 'Flat field axis' (X), 'Flat exposure' (Same), 'Collect flat fields' (Now), and '# Flat fields' (20).
- Dark Field Control:** Includes '# Dark fields' (50), 'Dark value' (0), and 'Collect dark fields' (Now).
- File Control:** Includes 'Overwrite warning' (Yes), 'Exists' (Yes), 'File directory' (/local/data/2020-02/decanlo/), and 'Base file name' (base_stream).
- Streaming Control:** Includes '# Pre count' (100), 'Buffer Wrapping', 'Capture proj' (Start/Stop), '# Capture' (100/10), 'File name' (base_stream_015.h5), '# Proj' (200), and 'Broadcast binning' (4x).
- Data Collection:** Includes 'Exposure time' (0.030), 'Start Scan' (green button), 'Abort Scan' (red button), and 'Status' (Done).
- Status:** Displays 'Scan status Scan complete', 'Images collected 4663/5000', 'Images saved 10/10', 'Elapsed time 0:02:57', 'Remaining time 0:00:12', and 'Python server Running'.

All `tomoscan_stream_2bm` functionalities supporting **tomostream** can be controlled from the `tomoScanStream` user interface marked in yellow.

Run tomoStream

- start tomoStream IOC, e.g.:

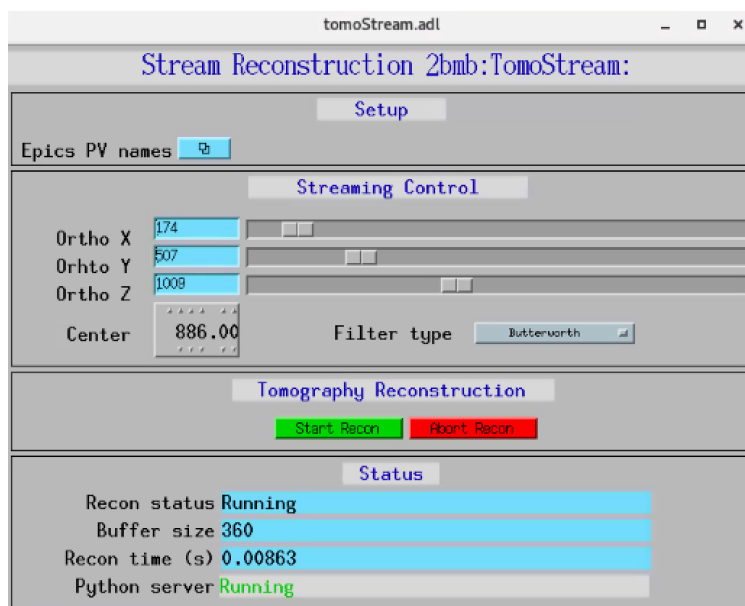
```
tomo@handyn$ cd /local/tomo/epics/synApps/support/tomostream/iocBoot/iocTomoStream/
tomo@handyn$ ./start_IOC
```

- start the `tomostream.py` supporting streaming reconstruction, e.g.:

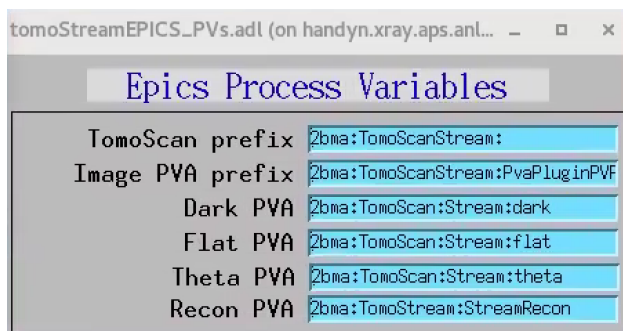
```
tomo@handyn$ cd /local/tomo/epics/synApps/support/tomostream/iocBoot/iocTomoStream/
tomo@handyn$ python -i start_tomostream.py
```

- start tomoStream user interface, e.g.:

```
tomo@handyn$ cd /local/tomo/epics/synApps/support/tomostream/iocBoot/iocTomoStream/
tomo@handyn$ ./start_medm
```



Open the EPICS PV names configuration screen:



to set the `TomoScan` prefix and the PVAccess names provided by `tomoScan` for projection (Image), dark and flat image broadcast. Here also set the Recon PVAccess name where the streaming reconstruction will served. Use the Recon

tomoStream PVA Names

Record name	Description
\$(P)\$(R)\$(I)\$(G)\$(C)\$(P)\$(V)\$(A)\$(N)\$(a)\$(m)\$(e)	name of the TomoScan PV storing the PV prefix of the images streamed by the detector
\$(P)\$(R)\$(I)\$(D)\$(C)\$(P)\$(V)\$(A)\$(N)\$(a)\$(m)\$(e)	the name of the TomoScan PVA where the dark images are stored
\$(P)\$(R)\$(I)\$(F)\$(C)\$(P)\$(V)\$(A)\$(N)\$(a)\$(m)\$(e)	the name of the TomoScan PVA where the flat images are stored
\$(P)\$(R)\$(I)\$(R)\$(C)\$(P)\$(V)\$(A)\$(N)\$(a)\$(m)\$(e)	the name of the TomoScan PVA where the rotation angle positions are stored
\$(P)\$(R)\$(I)\$(R)\$(C)\$(P)\$(V)\$(A)\$(N)\$(a)\$(m)\$(e)	the name of the TomoStream PVA where the the selected 3 orthogonal slices are stored

Streaming analysis control

Record name	Description
\$ (P) \$(R) \$(G) Camera ID Prefix	Prefix for the camera, e.g. 13BMDPG1:
\$ (P) \$(R) Status	Flag storing the streaming status. Choices are ‘Off’ and ‘On’. When ‘On’ the streaming reconstruction is enabled
\$ (P) \$(R) Buffer Size	Buffer size
\$ (P) \$(R) Rotation Center	Rotation center for streaming reconstruction
\$ (P) \$(R) Filter Type	Filter type for streaming reconstruction, ‘Parzen’, ‘Shepp-logan’, ‘Ramp’, ‘Butterworth’
\$ (P) \$(R) Order X	No slice in the X direction for streaming reconstruction
\$ (P) \$(R) Order Y	No slice in the Y direction for streaming reconstruction
\$ (P) \$(R) Order Z	No slice in the Z direction for streaming reconstruction

Stream status via Channel Access

Record name	Record type	Description
\$ (P)	\$ (R) Reconstruction form	This record will be updated with the stream reconstruction status while scanning.
\$ (P)	\$ (R) Reconstruction time	This record will update with the time to reconstruct the selected 3 orthogonal slices.
\$ (P)	\$ (R) Server Running	This record will be Running if the Python server is running and Stopped if not. It is controlled by a watchdog timer, and will change from Running to Stopped within 5 seconds if the Python server exits.

tomoStream_settings.req

This is the autosave request file for tomoStream.template tomoStream_settings.req.

It has the same usage and type of content as tomoStream_settings.req described above, except that it contains the PVs for the derived class TomoStream.

medm files

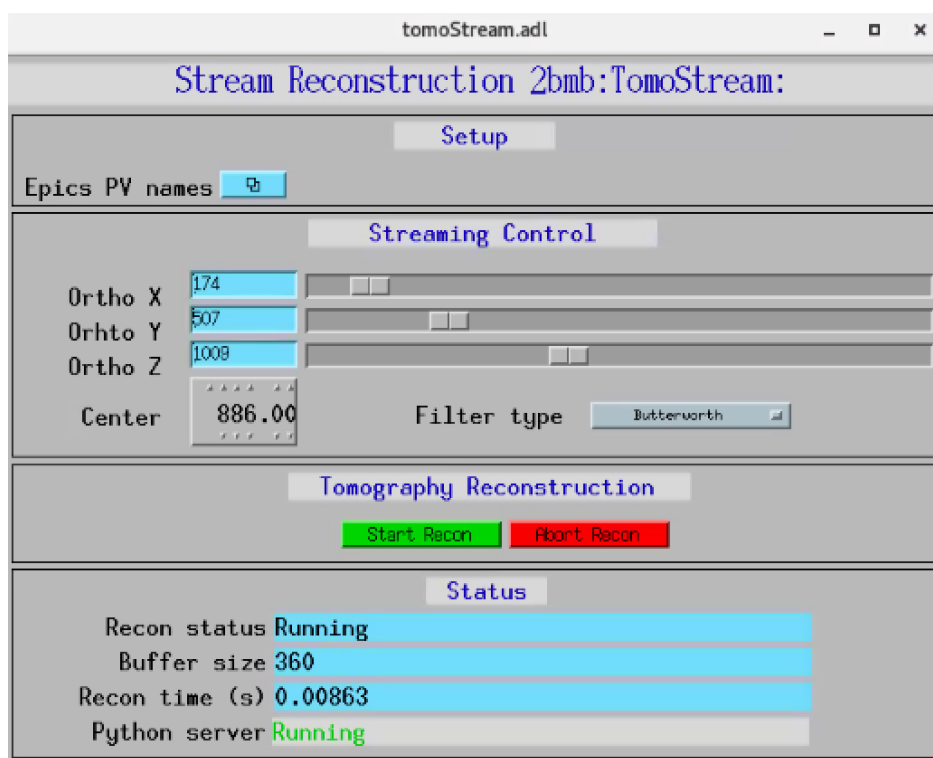
To start the tomostream medm screen:

```
$ cd /local/USERNAME/epics/synApps/support/tomostream/iocBoot/iocTomoStream
$ start_medm
```

where USERNAME is the username under which the tomoStreamApp is installed.

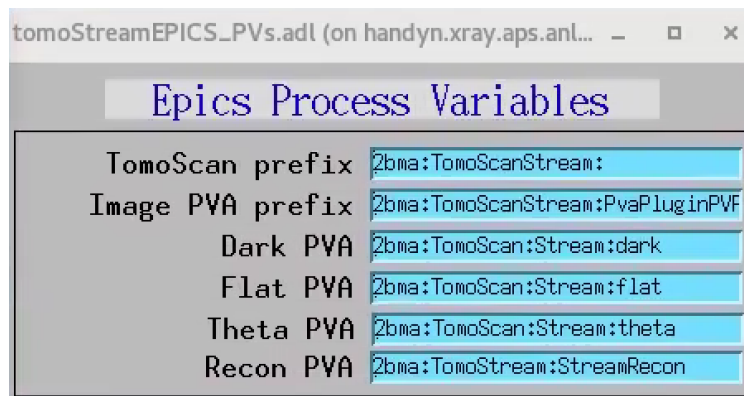
tomoStream.adl

The following is the MEDM screen tomoStream.adl. This screen contains the PVs to control tomoStream.



tomoStreamEPICS_PVs.adl

The EPICS PV names screen is below:



1.5 API reference

tomostream Modules:

1.5.1 tomostream.kernels

1.5.2 tomostream.tomostream

1.5.3 tomostream.solver

1.6 Credits

1.6.1 Citations

1.6.2 References

BIBLIOGRAPHY

- [A1] Viktor Nikitin, Aniket Tekawade, Anton Duchkov, Pavel Shevchenko, and Francesco De Carlo. Real-time streaming tomographic reconstruction with on-demand data capturing and 3D zooming to regions of interest. *Journal of Synchrotron Radiation*, 29(3):, May 2022. URL: <https://doi.org/10.1107/S1600577522003095>, doi:10.1107/S1600577522003095.
- [B1] Viktor Nikitin. Tomocupy: efficient gpu-based tomographic reconstruction with conveyor data processing. 2022. URL: <https://arxiv.org/abs/2209.08450>, doi:10.48550/ARXIV.2209.08450.