Data recording at SOLEIL

Stéphane POIRIER

TangoMeeting
November 17-18, 2020

Agenda

Introduction

Brief status of data recording

The recording service

- TangoRecorder
- RecordingManager

Datastorage GUI

Dataflow usecase

Introduction

This talk is about recording configuration & metadata recording

We developed early (starting in 2005) a recording service as a set of Tango devices

This service has been rewritten in order to improve performance and simplify its configuration

It is more flexible, open, easy to adapt to various environment

- plug-and-play approach for low dependencies between the components

Data recording at SOLEIL

The NeXus/HDF5 file format is the « standard » on most of the beamlines since beginning of operations at SOLEIL (2006)

- > 10 millions files
- Almost all beamlines (24 out of 29) record data using NeXus/HDF5
  - in 15 beamlines: systematically for raw data
  - for 9: depending on the context (acquisition type, instruments, ...)
  - 5 beamlines use their own system to manage experimental data

The Recording service

The data recording is managed using a couple of Tango devices

- **RecordingManager**: front-end, manage files paths & names,
- **TangoRecorder**: collects and writes metadata

Additional Tango devices for specific purposes

- **ProjectManager**: user authentication, project selection, data path locations, files ownership & permissions
- **FileTransfer**: copy/move files from the beamlines local storage into the central storage

A C++ API (libNexusCPP) on top on HDF5 1.8.x lib

- Used by acquisition devices to record experimental data
- Used by the TangoRecorder

Metadata recording: TangoRecorder

Its role: collect metadata from Tango devices and write it into a location defined by an URI

Few list of commands & attributes

- destinationURI (RW attr)
- The command *RecordDevices* take a list of devices names

Metadata collection is threaded. Recording typically take less than 1 second, even with hundreds of devices (3 seconds in the worse case).

TangoRecorder

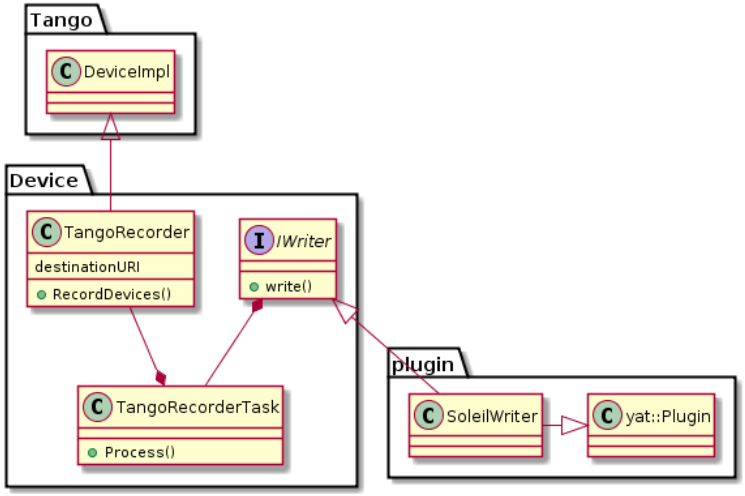
- Class Properties
- Device Properties
  - Symbols
  - ConfigPropName
  - Variables
  - Options
- Commands
  - State
  - Status
  - RecordDevices
  - Abort
  - IsDeviceRecordable
  - ClearLog
  - GetConfiguration
  - ErrorAck
- Scalar Attributes
  - destinationURI
  - configurationFilter
  - jobCount
  - isWriting
- Spectrum Attributes
  - configuration
  - deviceErrors
- Image Attributes
- States

TangoRecorder architecture

Modular conception to help deployement on another Tango environments

- metadata is written through a interface
- one concrete implementation: SoleilWriter
- plugin mechanism is on the way

```
class IWriter
{
public:

  //! virtual d-tor
  virtual ~IWriter() {}

  //! Ask the plugin to open an access to the given ressource
  //! \param uri: RFC 3986 complient Uniform Resource Identifier
  //! ex: file:///nfs/host/path/to/the/file.h5?uid=101,gid=10101
  virtual int open( const std::string& uri ) = 0;

  //! Ask the plugin to write the data collected from a device
  virtual int write( const DeviceRequest& device_request ) = 0;

  //! Ask the plugin to close the current ressource
  virtual int close() = 0;

  //! Get error text if needed
  virtual std::string get_last_error() = 0;
};
```



Metadata recording configuration

A property (default name is *Recording*) is set to each concerned class/device. It define the attributes to collect and their destinations inside the targeted container.

- The Recording property's content in device instance override the content of Recording property content defined at class level
- It use substitution variables to offer a flexible way to built datasets paths
  - variables may be define through in a TangoRecorder property (next slide)
  - automatic variables : 'device', 'domain', 'family', 'member'
  - some variable can be set in the query part of the URI string (e.g. 'entry')
  - environment variables can be used too (BEAMLINE is define at system level)
- The syntax is: '**`data item:target path inside destination`**'
  - data items are : “strings”, attributes names, properties name
  - The target path is decoded and interpreted by the writer object

Recording	"$(device)" : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/controller_record exposureTime : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/exposure_time nbFrames : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/nb_frames binningH : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/binning_x binningV : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/binning_y latencyTime : $(entry)(entry)/$(BEAMLINE)(instrument)/$(member)(detector)/gap_time

Metadata recording configuration

One can define as many variables as needed to shorten destination paths

```
# General destination paths inside the Nexus tree
entry_grp = /$(entry){entry}
instrument = $(entry_grp)/$(BEAMLINE){instrument}

# Final group name definition
device_names = $(domain)-$(family)-$(member)
device_name_or_alias = $(alias|device_names)
device_group_name = $(specific_group_name|device_name_or_alias)

# Final default group path for a device
device_instr_grp = $(instrument)/$(device_group_name)

# Paths related to device classes to put in the NXinstrument group
detector = $(device_instr_grp){detector}
bender = $(device_instr_grp){bender}
beam = $(device_instr_grp){beam}
diffractometer = $(device_instr_grp){diffractometer}
...
```

$(a|b) means : if 'a' is defined then substitute the value of 'a', otherwise substitute 'b'

Metadata recording configuration

Using the additional substitution variables the previous recording configuration for LimaDetector device class is:

Recording	"$(device)" : $(detector)/controller_record exposureTime : $(detector)/exposure_time nbFrames : $(detector)/nb_frames binningH : $(detector)/binning_x binningV : $(detector)/binning_y latencyTime : $(detector)/gap_time

Each time the TangoRecorder is asked to record metadata for a device, it read the 'Recording' property attached to both the device and its class

RecordingManager

Manage the data recording (experimental data & metadata). It is the entry point.

- High level commands

Dynamic interface

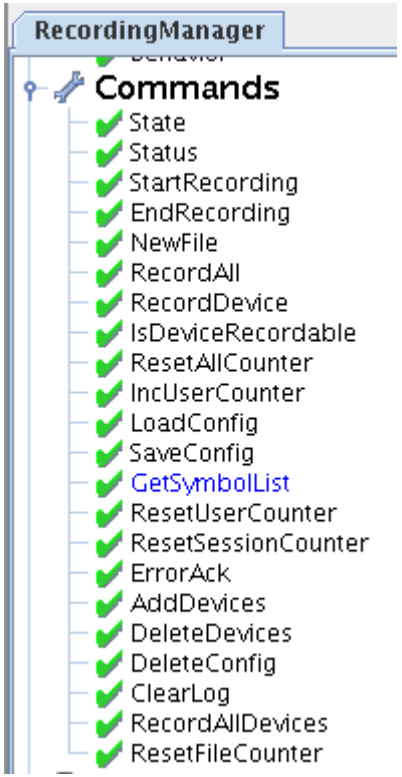
- Attributes for file name, data path, entry name (if using NeXus), intermediate path (if using a temporary storage),...

Automatic values

- Numeric counters, date/time values, attributes values of other devices can be used to build paths strings

Hook mechanism

- Allow to trigger actions on other devices at some key moments (start, end, new file)



Dynamic interface definition ('Variables' property)

- Each line defines a attribute
  - a short name that can be used in another dynamic field and to define the 'destinationURI' attribute value
  - the attribute name and label
  - some properties (RO/RW, operator/expert level, lock at start)
  - optionaly a data source (a tango attribute value of another device)
  - optionaly a default value

```
# user defined sub directory, file name and nxentry name
sdir@dataSubDirectory:Data subdirectory path:(O,L)
fn@fileName:File name:(O,L)
nx@rootNode:Root node:(O,L)

# Values available from the ProjectManager device (read-only)
rp@rootPath:Root path:storage/management/projectmanager.1/rootPath:(O)
uid@projectUid:Project UID:storage/management/projectmanager.1/projectUID:(O)
gid@projectGid:Project GID:storage/management/projectmanager.1/projectGID:(O)
pc@projectCode:Project code:storage/management/projectmanager.1/currentProject:(O)

# spool paths definitions
spp@spoolProjectPath:Spool project path:(O,R):/nfs/srv5/spool1/tangorecorder/[pc]

# Final destination in the storage facility
stp@storageDestPath:Storage destination path:(O,R):[rp]/[pc]/[sdir]
```

Destination paths and names is defined using a URI ('destinationURI' RO attribute) according to a specific pattern (defined by a property named 'URIpattern').

```
file:[spp]/[sdir]/[fn].nxs?entry=[nx],uid=[uid],gid=[gid],
file_mode=640,dir_mode=755,dev_config=[_dc_],dest_path=[stp]
```

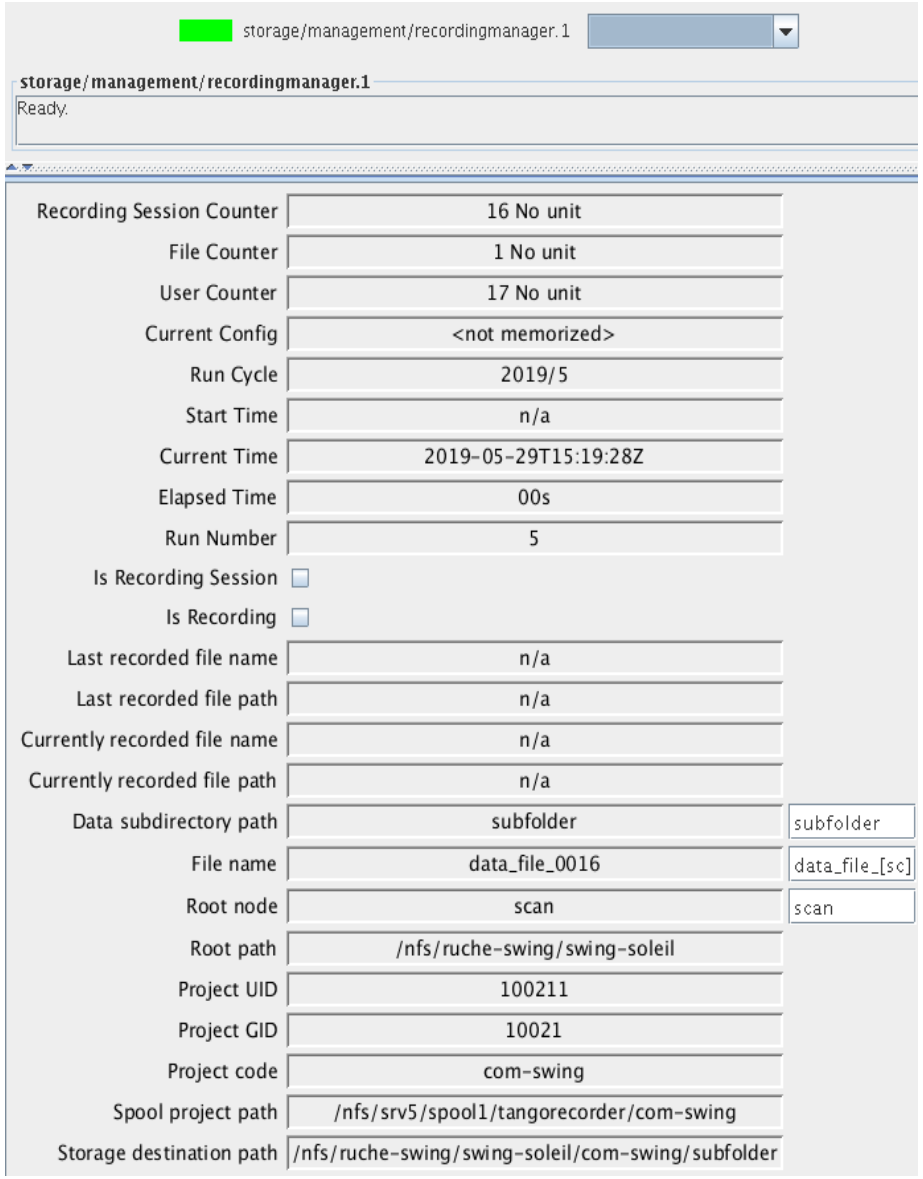
```
# user defined sub directory, file name and nxentry name
sdir@dataSubDirectory:Data subdirectory path:(O,L)
fn@fileName:File name:(O,L)
nx@rootNode:Root node:(O,L)

# Values available from the ProjectManager device (read-only)
rp@rootPath:Root path:storage/management/projectmanager.1/rootPath:(O)
uid@projectUid:Project UID:storage/management/projectmanager.1/projectUID:(O)
gid@projectGid:Project GID:storage/management/projectmanager.1/projectGID:(O)
pc@projectCode:Project code:storage/management/projectmanager.1/currentProject:(O)

# spool paths definitions
spp@spoolProjectPath:Spool project path:(O,R):/nfs/srv5/spool1/tangorecorder/[pc]

# Final destination in the storage facility
stp@storageDestPath:Storage destination path:(O,R):[rp]/[pc]/[sdir]
```

RecordingManager

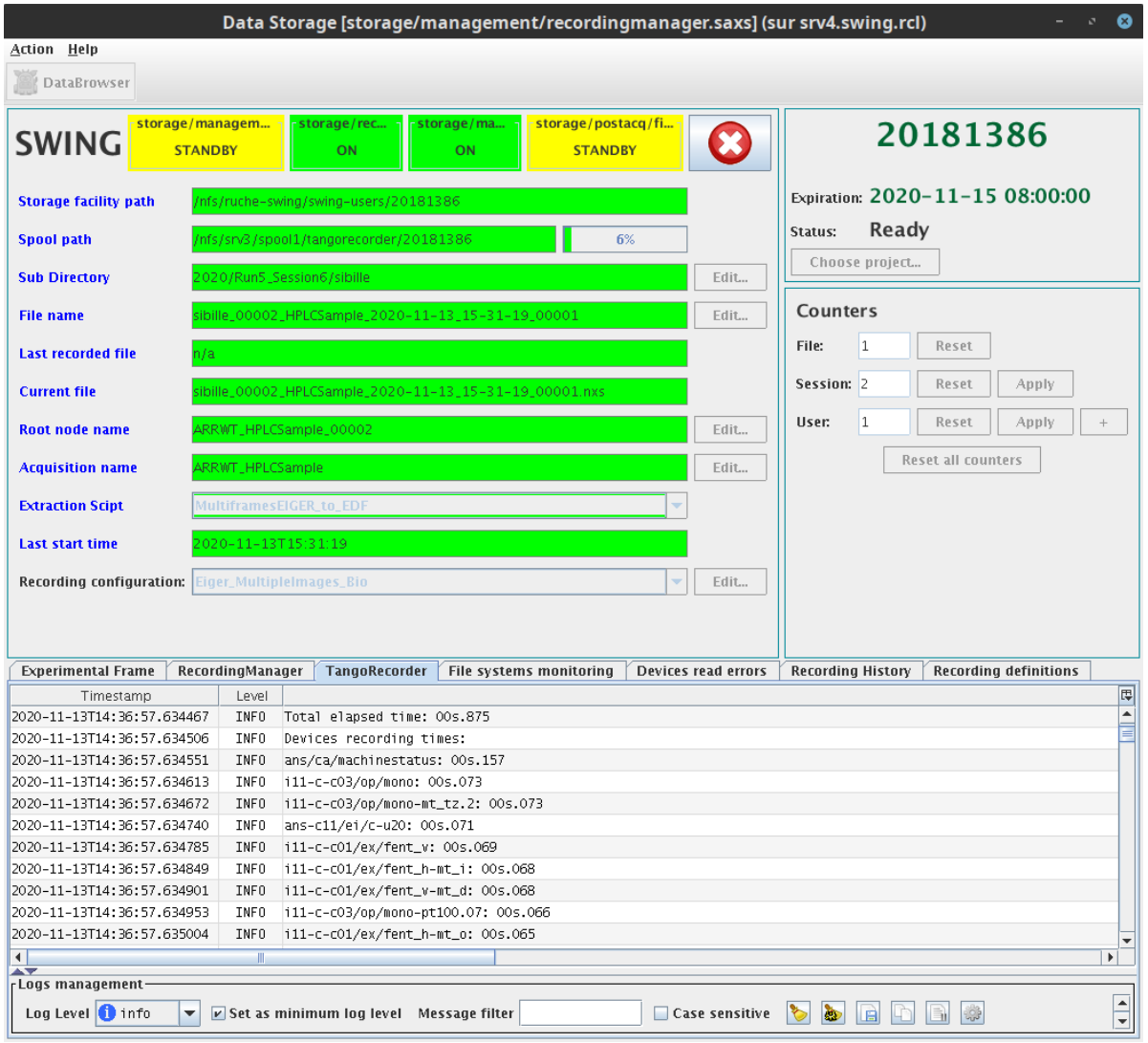
According to the previously defined dynamic interface...

The 'destinationURI' attribute (expert level) therefore is:

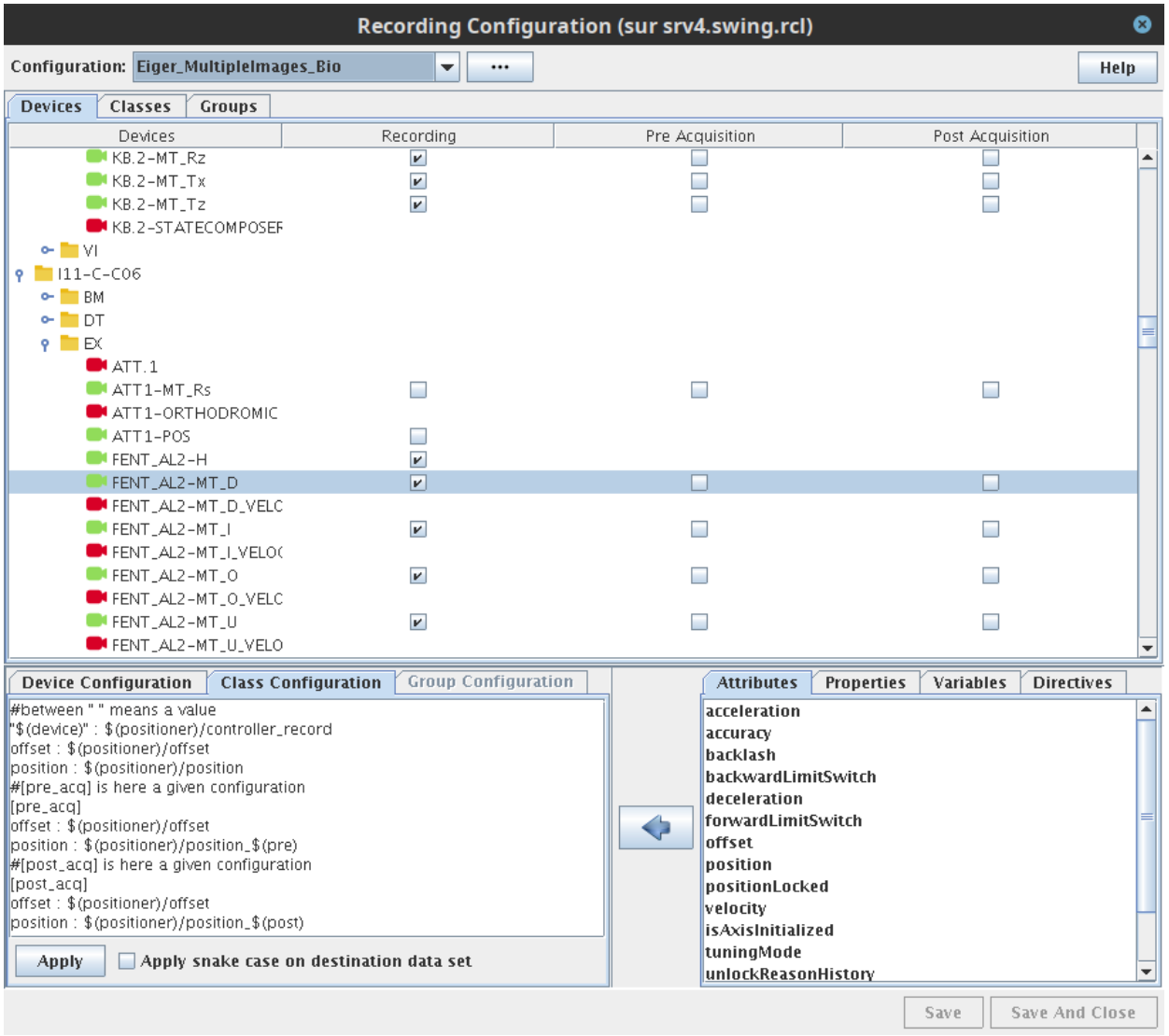
```
file:/nfs/srv5/spool1/tangorecorder/com-swing/subfolder/data_file_0016.nxs?
entry=scan,uid=100211,gid=10021,file_mode=640,dir_mode=755,dev_config=,
dest_path=/nfs/ruche-swing/swing-soleil/com-swing/subfolder
```



DataStorage GUI

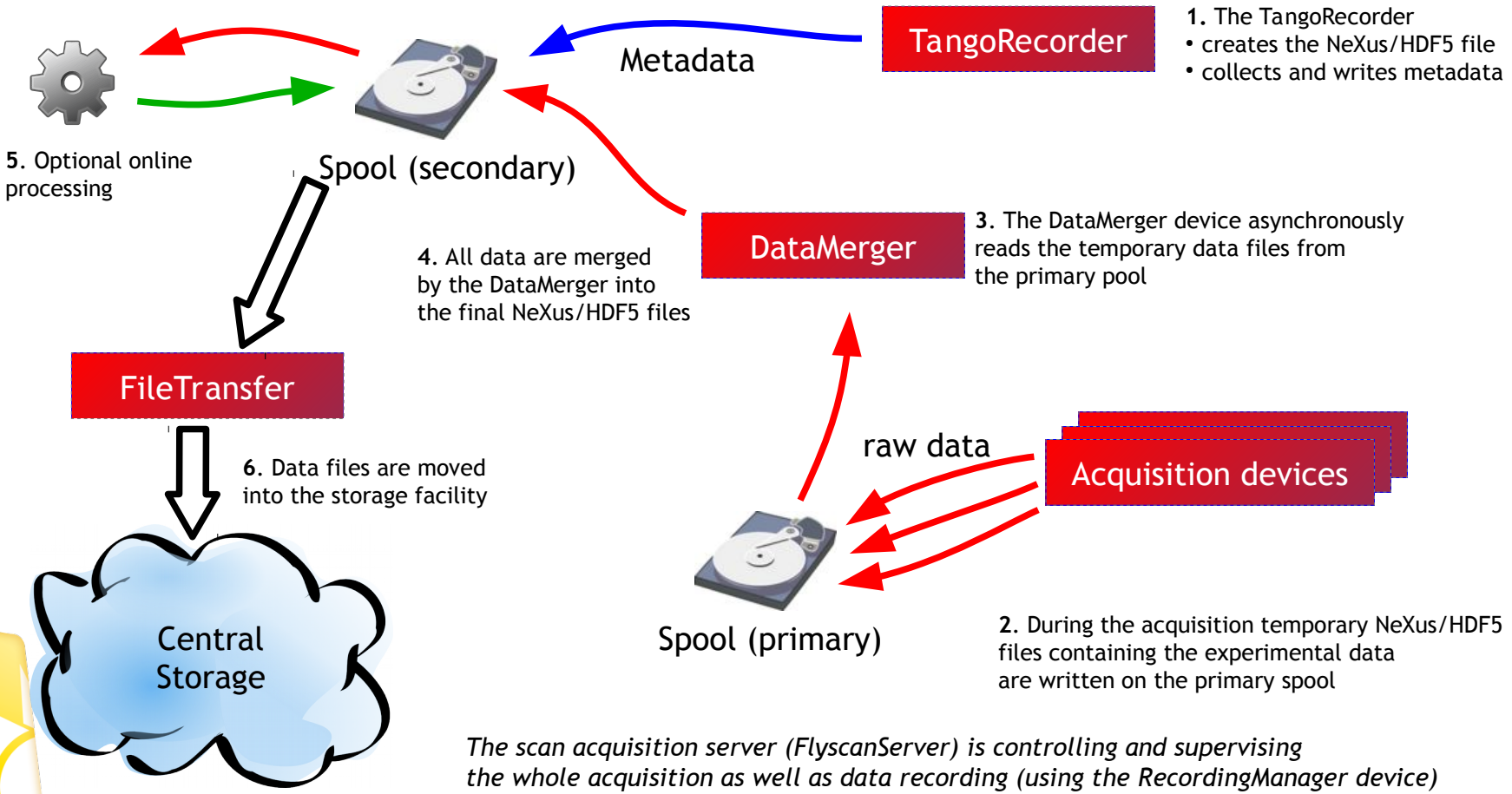


DataStorage GUI



Data flow

Asynchronous process to get better performance and scalability



The scan acquisition server (FlyscanServer) is controlling and supervising the whole acquisition as well as data recording (using the RecordingManager device)

Conclusion

This highly versatile service is now deployed on 18 beamlines and will be in production on 24 out of 29 beamlines during next spring

Complete the writer plugin mechanism

If another facility/institute is interested on using it, please contact us :)

Data recording at SOLEIL