

# **SAM User Guide**

**Robert A. Illingworth  
Marc W. Mengel  
Katherine Lato**

**December 1, 2014**

## **Abstract**

Sequential Access via Metadata (SAM) is a data handling system to store and retrieve files and associated metadata, including a complete record of the processing which has used or generated the files. The User Guide for SAM includes how to use the samweb interface, the File Transfer System, and monitoring of SAM jobs.

The location of copies of files changes over time. Files can be loaded from a tape to a fast-cache disk, or may move onto tape for long-term storage. At any given time, users don't need to know where their files are physically located. SAM handles the bookkeeping by using metadata, which is data about data. Users define what they want, and SAM finds and delivers the files, on whatever disk the file is needed.

For up-to-date documentation on using SAM, please see:

[https://cdcvs.fnal.gov/redmine/projects/sam/wiki/User\\_Guide\\_for\\_SAM](https://cdcvs.fnal.gov/redmine/projects/sam/wiki/User_Guide_for_SAM)

# Table of Contents

<b>1 Why use SAM?</b> .....	<b>3</b>
<b>1.1 Data File Location</b> .....	<b>3</b>
<b>2 Using SAM</b> .....	<b>4</b>
<b>2.1 Metadata</b> .....	<b>4</b>
<b>2.2 samweb Interface</b> .....	<b>5</b>
2.2.1 Setting up samweb Tools.....	6
2.2.2 samweb help.....	7
2.2.3 Getting Metadata.....	8
2.2.4 Getting ancestors/children.....	9
2.2.5 Listing Projects.....	9
2.2.6 Listing Definitions.....	10
2.2.7 Describing Definitions.....	10
2.2.8 Taking a Snapshot.....	10
2.2.9 Listing Files.....	11
2.2.10 Seeing File Location.....	12
<b>2.3 Activity Logging -- Where to Look When Things go Wrong</b> .....	<b>12</b>
2.3.1 Recover Dataset for a Project.....	13
<b>2.4 Constructing Datasets</b> .....	<b>13</b>
2.4.1 Datasets can be dynamic or Static.....	14
2.4.2 Definition Editor.....	14
<b>3 How SAM Works</b> .....	<b>15</b>
<b>3.1 What SAM Does</b> .....	<b>15</b>
<b>3.2 SAM Concepts</b> .....	<b>15</b>
3.2.1 SAM Datasets.....	16
3.2.2 Snapshot.....	16
3.2.3 Project.....	17
3.2.4 Station.....	17
<b>3.3 SAM Monitoring Example</b> .....	<b>17</b>
<b>3.4 Decoupling SAM Internals from Experiment's Framework</b> .....	<b>18</b>
<b>3.5 Location Catalog</b> .....	<b>19</b>
<b>3.6 Adding Files to the System (Cataloging files in SAM)</b> .....	<b>19</b>
<b>4 Processing Files</b> .....	<b>20</b>
<b>4.1 SAM Project Example</b> .....	<b>20</b>
4.1.1 Batch Project Example.....	21
4.1.2 Using SAM Interactively.....	21
<b>5 Integrating SAM with Experiments</b> .....	<b>22</b>
<b>5.1 art and SAM</b> .....	<b>23</b>
<b>6 SAM Reference Material</b> .....	<b>24</b>
<b>6.1 Glossary</b> .....	<b>24</b>
<b>6.2 References</b> .....	<b>25</b>

# 1 Why use SAM?

Sequential Access via Metadata (SAM) is a data handling system to store and retrieve files and associated metadata, including a complete record of the processing which has used the files. Experiments need to manage a large volume of data. Storing files in nested directories on an NFS fileserver isn't scalable, and makes it hard to find specific data, and it can fill up the allowed disk space. It can also be difficult to ensure that people are doing studies with the right versions of their data. SAM indexes all data according to metadata and delivers files in a storage location agnostic fashion.

SAM handles the file metadata and bookkeeping so the user doesn't need to know where the files are, or what the individual files are, to find data of interest. Bookkeeping information is kept in a database, which can be accessed by a series of commands described later.

The dataflow for analysis isn't as simple as taking raw data, reconstructing it, analyzing the reconstructed data and publishing paper(s). Even a small experiment has millions of data files to keep track of, and a complicated process to ensure the correct ones are given to the next stage of processing. SAM helps with this.

SAM is used to:

- get information such as finding out the data and time of a particular subrun.
- get a list of data files that match particular criteria, for example all raw data files from a specific time period where equipment was in a particular mode.
- manage data storage to disk or tape without the user needing to know where the data is currently stored.

## 1.1 Data File Location

As experiments take data, it's not practical to store every single data file (raw and offline processed) on the Fermilab central disk services. Files are typically stored on a combination of:

### \* Hierarchical storage

- dCache/Enstore at Fermilab
- dCache/HPSS at BNL
- CASTOR at CERN
- GPFS/TSM at INFN
- etc.

### \* NFS Filers

- BlueArc at Fermilab
- BNL Filer at CERN

#### \* Distributed Filesystems

- LUSTRE
- CEPH
- AFS
- etc.

Except for special applications (with very small files), central disks shouldn't be used. The central disk should be avoided for batch jobs. There is also the local disk, i.e. the disk of the node where the job runs, but the files are only there temporarily while being used for a job.

The location of a file changes over time as it migrates between different locations. It can be loaded from a tape to a fast cache disk, or may move off of BlueArc and onto tape for long term storage. At any given time, users don't know where their files are physically located. SAM handles the bookkeeping. Users define what they want, and SAM finds and delivers the files, on whatever disk the file is needed.

## 2 Using SAM

### 2.1 Metadata

---

Metadata is data about data. The main purpose is to allow files to be queried. Metadata includes physical data such as file size or checksum. It also includes the physics metadata such as: run number, detector configuration, and simulation parameters. Raw and processed data files have a large amount of metadata that describe the files themselves, how they were generated, and other auxiliary information useful in understanding what is in a file or how it should be grouped with other files.

SAM contains a metadata catalog to store information about experiment files. Most of this data is experiment-defined. It is used to specify the properties of files the user wants to look at. For example:

- `data_tier = reconstructed`  
User only wants reconstructed files
- `nova.detectorid = fd`  
User only wants far detector files

Every file stored in SAM must have metadata associated with it, although some types of files need very little. Metadata is normally uploaded when the file is first added to the database. Since metadata is used to catalog the data, it is best to add it early.

Provenance, also known as pedigree or lineage, refers to the complete history of a file. Provenance data facilitates reproduction and enables validation of results. The metadata in SAM stores the parent files from which it was derived, and the application and version used to create it.

The metadata fields mostly consist of key-value pairs which is a set of two linked data items: a key, the unique identifier for some item of data, and the value, which is either the data that is identified or a pointer to the location of that data. For example:

```
"file_type": "importedDetector"

"file_format": "raw"

"application": {
  "family": "online",
  "name": "datalogger",
  "version": "33"
}
```

Experiments can define their own fields, called parameters, as well as use the predefined metadata fields at: [https://cdcvns.fnal.gov/redmine/projects/sam-web/wiki/Metadata\\_format](https://cdcvns.fnal.gov/redmine/projects/sam-web/wiki/Metadata_format). Values may be anything, and can be used to store any data relevant to the file. Parameters must be defined in advance of use.

Most experiments have a person, or a small group of people, who define experiment-specific parameters. The average user just needs to know where they can be found for their experiment. As an example, Minerva use the following parameters to state whether a data file has passed a quality check.

```
Quality.MINERvA: Good
Quality.overall: good
Quality.preproc: good
Quality.tar-checked: good
```

All the metadata is stored in a database by SAM, but users don't interact with the database directly. Instead, client programs send requests to an http(s) server called a samweb server.

## 2.2 samweb Interface

---

Most users need to know how to list files, count files, and examine information about files. A few people in each experiment need to know how to define a dataset, how to run a project, and how to create and store files into SAM, as well as other administrative tasks.

The samweb interface is not a website despite the name. It's generally used via a command line application or a python or C++ API.

All SAM clients send requests to the samweb http server (<http://samweb.fnal.gov:8480/sam/uboone/api>)

Ways to interface with SAM include:

- samweb - setup sam\_web\_client
  - line mode - samweb -e <experiment-name> <subcommand> ...

- python - import samweb\_cli
- ifdh client tools - setup ifdhc
  - line mode - ifdh <subcomand>
  - python - import ifdh
  - C++ client - class ifdh, not art-specific
- art client - setup ifdh\_art
  - wraps ifdhc c++ sam client as art service (IFDH service) and provides sam-capable instances of file delivery and file transfer services.

Note, ifdhc has a limited samweb client interface. The full interface is available with sam\_web\_client. The ifdhc commands provide a way for experiments to run batch jobs OSG-wide via ifdh cp instead of cp or cpn commands, and to declare metadata for their files into SAM. For more about ifdhc, see:

[https://cdcvs.fnal.gov/redmine/projects/ifdhc/wiki/Ifdh\\_commands](https://cdcvs.fnal.gov/redmine/projects/ifdhc/wiki/Ifdh_commands)

### 2.2.1 Setting up samweb Tools

---

The samweb user tools are provided as a UPS product. The name of the package is "sam\_web\_client" and it is available from the common products area at FNAL. To set it up, set up the ups command (unless you've already done that for another product), add the common products area to your UPS products path, then do a setup and authentication:

```
% source /grid/fermiapp/products/common/etc/setups.sh
% export PRODUCTS=/grid/fermiapp/products/common/db/:$PRODUCTS
% setup sam_web_client
% kx509
```

This will give access to the "samweb" tool. This tool can be run anywhere and communicates with SAM through the SAM web server (http access). By default the authentication mechanism used to communicate with the server is a kx509 certificate based authentication. To protect against unauthorized access to Fermilab computers, the Computing Division has implemented Kerberos to provide what is known as strong authentication over the network. Access to Grid resources (jobs, FTP transfers) is via X509 SSL certificates, which may be generated from the kerberos ticket. For job submission, this is handled by the kproxy script. For file access, users may need to run the 'get-cert' command. For more information on certification, see:

<https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Authentication>

Often, users will need to set up experiment-specific information. Minimally, set the experiment with an environment variable to save typing.

```
export EXPERIMENT=<experiment-name>
export SAM_EXPERIMENT=<experiment-name>
```

For example:  

```
export EXPERIMENT=nova
```

```
export SAM_EXPERIMENT=nova
```

This setup information can be automated by putting the information in your login files, i.e. `bash_profile` or `.bashrc`. Remember that your home area is shared when you log into any of the experiment gpvm machines. This is important to keep in mind if you are on multiple experiments.

If you want to want to do something with a different experiment, you can type `"-e <experiment>"` to override the environment variable.

### 2.2.2 samweb help

---

Show available commands:

```
%samweb --help-commands
```

The commands are divided into administration, definition, data file, project and utility.

For all available commands, please see: [https://cdevs.fnal.gov/redmine/projects/sam-main/wiki/Sam\\_web\\_client\\_Command\\_Reference](https://cdevs.fnal.gov/redmine/projects/sam-main/wiki/Sam_web_client_Command_Reference)

If a command is mistyped, the general usage statement for samweb will appear.

The base options for each command are the same, and many commands have command-specific options as well. These can be seen by typing:

```
% samweb <command> -h
```

This gives a usage statement, a short description of the command, the base options, and command-specific options, if any.

For example:

```
samweb get-metadata -h
usage: samweb [base options] get-metadata [command options]
<file name>
```

Get metadata for a file

options:

```
-h, --help          show this help message and exit
--help-commands    list available commands
```

Base options:

```
-e EXPERIMENT, --experiment=EXPERIMENT
    use this experiment server.
    If not set, defaults to
    $SAM_EXPERIMENT.
--dev          use development server
-s, --secure   always use secure (SSL) mode
--cert=CERT    x509 certificate for authentication.
                If not specified,
                use $X509_USER_PROXY, $X509_USER_CERT/
                $X509_USER_KEY or standard grid proxy
```

```
location
--key=KEY      x509 key for authentication (defaults
               to same as certificate)
-v, --verbose  Verbose mode
```

```
get-metadata options:
--json
```

In this example the 'get-metadata' specific option is to specify [--json](#)

For these examples, assume that the experiment was set to nova, as follows.

```
export EXPERIMENT=nova
export SAM_EXPERIMENT=nova
```

### 2.2.3 Getting Metadata

---

Can list the Metadata associated with a file via the 'get-metadata' command. Since all files in SAM are unique, the file name doesn't have a path, just the base name.

```
samweb [base options] get-metadata [command options] <file name>
```

```
%samweb get-metadata a_specific_file_name
```

Specific example:

```
%samweb get-metadata fardet_r00013114_s20_t00.raw
```

```
File Name: fardet_r00013114_s20_t00.raw
File Id: 4877797
File Type: importedDetector
File Format: raw
File Size: 6908296
        Crc: 74650857 (adler 32 crc type)
Content Status: good
        Group: nova
        Data Tier: raw
Application: online datalogger 33
Event Count: 110
First Event: 171026
Last Event: 179507
Start Time: 2014-02-14T01:34:14
End Time: 2014-02-14T01:37:43
Data Stream: 0
Online.ConfigIDX: 0
Online.DataLoggerID: 1
Online.DataLoggerVersion: 33
Online.Detector: fardet
Online.DetectorID: 2
Online.Partition: 1
Online.RunControlID: 0
Online.RunControlVersion: 0
Online.RunEndTime: 1392341863
Online.RunNumber: 13114
Online.RunSize: 1727074
```

```

        Online.RunStartTime: 1392337488
            Online.RunType: 0
                Online.Stream: 0
        Online.SubRunEndTime: 1392341863
    Online.SubRunStartTime: 1392341654
        Online.Subrun: 20
            Online.TotalEvents: 110
                Online.TriggerCtrlID: 0
                    Online.TriggerListIDX: 0
    Online.TriggerPrescaleListIDX: 0
        Online.TriggerVersion: 0
    Online.ValidTriggerTypesHigh: 0
    Online.ValidTriggerTypesHigh2: 0
        Online.ValidTriggerTypesLow: 0
            Runs: 13114.0020 (online)
                File Partition: 20

```

## 2.2.4 Getting ancestors/children

---

Can get the lineage of a file. These can be children/descendants or parents/ancestors. Children are files derived directly from the input file.

```

samweb [base options] file-lineage [command options] \
<parents|children|ancestors|descendants|rawancestors> \
<file name>

```

Specific examples:

```
% samweb file-lineage children fardet_r00013096_s14_t00.raw
```

```
% samweb file-lineage parents
fardet_r00013096_s14_t00_numi_S14-01-20_v1_data.daq.root
```

## 2.2.5 Listing Projects

---

```
samweb [base options] list-projects [command options]
```

list-projects options:

```
--name=NAME
```

```
--user=USER
```

```
--group=GROUP
```

```
--defname=DEFNAME
```

```
--snapshot_id=SNAPSHOT_ID
```

```
--started_before=STARTED_BEFORE
```

```
--started_after=STARTED_AFTER
```

```
--ended_before=ENDED_BEFORE
--ended_after=ENDED_AFTER
```

To see the projects for user mengel:

```
%samweb list-projects --user=mengel
mwmTestNov9_a
mengel-art_sam_wrap.sh_20121108_235445_32228
mengel-art_sam_wrap.sh_20121108_235957_32485
(many more)
```

### 2.2.6 Listing Definitions

---

List existing dataset definitions.

```
samweb [base options] list-definitions [command options]
```

list-definitions options:

```
--defname=DEFNAME
--user=USER
--group=GROUP
--after=AFTER
--before=BEFORE
```

```
%samweb -e nova list-definitions --after=2014-04-26
fardet_data_raw2root_keepup_limited.04-25-2014
fardet_data_raw2root_keepup_limited.04-24-2014
...
tute-prodarddaq_S14-02-05CryFD_r1000001
```

### 2.2.7 Describing Definitions

---

Describe an existing dataset definition

```
samweb [base options] describe-definition [command options] <dataset definition>
```

```
%samweb describe-definition tute-prodarddaq_S14-02-
05CryFD_r1000001
Definition Name: tute-prodarddaq_S14-02-05CryFD_r1000001
Definition Id: 10417
Creation Date: 2014-04-27T23:39:16
Username: gsdavies
Group: nova
Dimensions: data_tier artdaq and nova.detectorid fd and ...
```

Be sure to check that the files are what you expect, as well as determine how many of them there are before running on the grid.

### 2.2.8 Taking a Snapshot

---

Users can create a snapshot of an existing dataset definition by:

```
samweb [base options] take-snapshot [command options] <dataset definition>
```

take-snapshot options:

```
--group=GROUP
```

```
% samweb take-snapshot mwm_test_9  
13991
```

If the dataset is large, this can take a while as SAM builds a list of all the files satisfying the query. Users can then use the id from the take-snapshot command to list the files.

## 2.2.9 Listing Files

---

List files by dimensions query

```
samweb [base options] list-files [command options] <dimensions query>
```

list-files options:

```
--parse-only Return parser output for these dimensions instead of evaluating them
```

```
--fileinfo Return additional information for each file
```

```
--summary Return a summary of the results instead of the full list
```

```
--help-dimensions Return information on the available dimensions
```

Users can list files from specific dimensions such as a snapshot, data tier and a detector, from a run, from a time period, and many other specifics.

### Listing files by Snapshot

```
%samweb list-files "snapshot_id 13991"  
sim_cosmics_nd_10000_r1_2311_S12.06.17_20120703_153830_daq.root
```

This returns the list of files in the snapshot. By taking the snapshot, and then using the snapshot id, users are assured the list of files has not changed even if someone added new files to SAM.

### Listing Files by Data Tier and Detector

Data tier and detector example:

```
%samweb list-files "data_tier raw AND online.detector fardet"
```

### Listing Files using a Variable to Build the Query

A variable such as BASE\_QUERY can be used.

```
%export BASE_QUERY="data_tier raw AND online.detector fardet"  
%samweb list-files -e nova $BASE_QUERY
```

Then it can be expanded:

```
%samweb list-files "$BASE_QUERY and run_number 13114"  
fardet_r00013114_s01_t00.raw  
fardet_r00013114_s04_t04.raw
```

```
fardet_r00013114_s05_unknown.raw
fardet_r00013114_s10_t00.raw
(and many more)
```

```
Files created between two times:
%samweb list-files "$BASE_QUERY and start_time > \
'2014-01-30T23:29:00'and start_time < '2014-01-31T00:30:00'"
fardet_r00012787_s07_t00.raw
fardet_r00012787_s07_unknown.raw
fardet_r00012787_s12_t00.raw
fardet_r00012787_s15_unknown.raw
(and many more)
```

### **Listing Files that match a Pattern**

Can match part of a file name

```
%samweb list-files -e nova \
"file_name like fardet_r00011414%s60%raw%"
fardet_r00011414_s60.raw
fardet_r00011414_s60_t00.raw
fardet_r00011414_s60_t02.raw
fardet_r00011414_s60_t05.raw
```

#### **2.2.10 Seeing File Location**

---

```
samweb [base options] locate-file [command options] <file name>
%samweb locate-file -e nova fardet_r00011414_s60_t05.raw
novadata:/nova/data/rawdata/FarDet/000114/11414/05
enstore:/pnfs/nova/rawdata/FarDet/000114/11414(66@vpm007)
```

### **2.3 Activity Logging -- Where to Look When Things go Wrong**

---

All SAM file activity is logged in a database. This keeps a complete record of which files were given to which jobs, and which files and jobs were reported as having completed successfully.

The tracking of all file activity allows for simple recovery for files that were not correctly processed. This can happen for a variety of reasons:

- the storage system may not have been able to provide some of the dataset;
- the job may not have completed processing due to preemption or crashes;
- output may have been lost due to some failure of the copy back procedure.

SAM can automatically create a recovery dataset for any project. This consists of all the files that were not successfully processed in the first pass.

The following shows an error display, just as an example:

<b>Process id</b>	37327944
<b>Node name</b>	fcdfnx5.fnal.gov
<b>Status</b>	error
<b>Start time</b>	2014-04-23 06:54:44
<b>Files seen</b>	10
<b>Last activity</b>	2014-04-23 07:09:44 (process ended - error)

### 2.3.1 Recover Dataset for a Project

---

To display the dimensions for the recovery dataset for a project

```
samweb [base options] project-recovery [command options] <project name>
```

To make a recovery dataset:

```
setup sam_web_client
samweb project-recovery -e $EXPERIMENT $project_name
```

This gives either a definition or an empty string. (An empty string means nothing needs doing.)

Specific example:

```
%samweb project-recovery mengel-
xx_art_sam_wrap.sh_20140522_163831_23529
(snapshot_id 14404 minus (project_name mengel-
xx_art_sam_wrap.sh_20140522_163831_23529 and
consumed_status consumed)) or consumer_process_id in
(692938,692939,692940)
```

From this, users can define a dataset and run a job to gather more information, or try the dataset again.

## 2.4 Constructing Datasets

---

Note: in most experiments, only a small group of people create new datasets, but it's useful to know what is going on.

When working with a dataset definition, users start by selecting the type of data files, also called specifying file dimensions. Essentially, specifying a file dimension means searching for specific values of metadata parameters. For example, users can select a date range, multiple date ranges, a run number, or subrun numbers. Each time a dimension is added, users include a logical operation that specifies how to select it (i.e. AND or OR).

Users can select other criteria including the trigger stream, number of events in a file, the actual file size or any other metadata that is recorded for the class of files of interest.

To see the possibilities, consult: [https://cdcvs.fnal.gov/redmine/projects/sam-web/wiki/Dimension\\_Syntax](https://cdcvs.fnal.gov/redmine/projects/sam-web/wiki/Dimension_Syntax)

The dataset definition can be tested by submitting the query.

For example, you can query to find all the raw files that belong to a particular run as follows:

```
$ samweb -e samdev list-files "data_tier raw and run_number 798"
MN_00000798_0004_numib_v04_0911090240_RawEvents.root
MN_00000798_0004_numib_v04_0911090239_RawEvents.root
MN_00000798_0005_numib_v04_0911090240_RawEvents.root
MN_00000798_0006_numib_v04_0911090240_RawEvents.root
```

Note, users typically run a one-time query, as shown above, and then when satisfied that the query syntax is correct, they save that query as a dataset definition.

```
samweb [base options] create-definition [command options] \  
<new definition name> <dimensions>
```

```
$ samweb create-definition raw_numib_v04_09_rawevents
```

#### 2.4.1 Datasets can be dynamic or Static

---

Since the database query is run whenever the user requests the dataset, datasets can dynamically change based on the query. For example, if the dataset is all the files since May 1, 2014, that dataset will grow, while one that is defined as the data from April 1 2014 to May 1 2014 may be static. But, if files are backdated later, and fall within that time period, then the dataset will change. The only thing really static is the snapshot.

#### 2.4.2 Definition Editor

---

The following definition editors have been defined. These can be useful to define datasets.

These web forms let you build up a 'dimension' query by choosing options from menus rather than having to remember the syntax.

[http://samweb.fnal.gov:8480/sam/coupp/definition\\_editor/](http://samweb.fnal.gov:8480/sam/coupp/definition_editor/)

[http://samweb.fnal.gov:8480/sam/gm2/definition\\_editor/](http://samweb.fnal.gov:8480/sam/gm2/definition_editor/)

[http://samweb.fnal.gov:8480/sam/minerva/definition\\_editor/](http://samweb.fnal.gov:8480/sam/minerva/definition_editor/)

[http://samweb.fnal.gov:8480/sam/mu2e/definition\\_editor/](http://samweb.fnal.gov:8480/sam/mu2e/definition_editor/)

[http://samweb.fnal.gov:8480/sam/nova/definition\\_editor/](http://samweb.fnal.gov:8480/sam/nova/definition_editor/)

[http://samweb.fnal.gov:8480/sam/uboone/definition\\_editor/](http://samweb.fnal.gov:8480/sam/uboone/definition_editor/)

Nova has them defined for: Generic editor, Raw data editor, MC editor and Reco editor

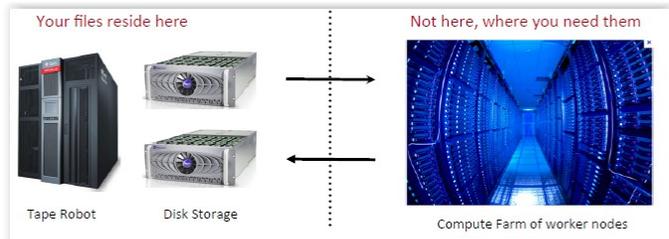
## 3 How SAM Works

### 3.1 What SAM Does

---

As much as possible, SAM frees experiments from dealing with the specific details of their data so they can focus on the content. SAM provides:

- metadata catalog — What's in the data files?
- location catalog - Where are the files?
- data delivery service - Access to the files.



While people commonly think of files as being identified by name and directory, they can be identified by a unique name so they do not require knowing the path, which is location-specific.

SAM is a highly-automated, hands-off system requiring minimal routine intervention. Experiments don't need to provide dedicated expertise for data management. Data staging from tape and from storage element to storage element is automatic. The philosophy of SAM is, 'Bring the data to the jobs.'

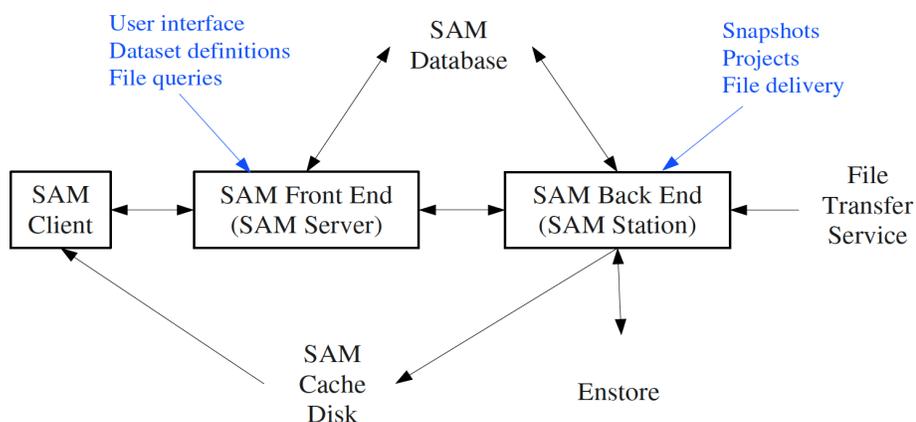
SAM was developed and used on the Run II of the Tevatron. It evolved and was improved and modernized for use by multiple current and future Fermilab experiments.

### 3.2 SAM Concepts

---

SAM is organized around the concepts of datasets, snapshots, projects and stations. A station manages a collection of hardware resources. Projects are attached to stations. SAM tracks data from a variety of experiments, and understanding how SAM works can help to effectively use SAM within an experiment.

# Data Handling Overview



## 3.2.1 SAM Datasets

Users often want to analyze a group of files that share characteristics. This could be the set of files that are from a certain run, files from a given time range, files that have events from a specific trigger, or other common characteristics. This collection of files is called a dataset. It is convenient to give datasets symbolic and descriptive names that can be used by others when performing similar analysis.

Example:

The user wants to analyze the set of all raw files which meet the following criteria:

1. Are from May 1 to June 1 of 2014
2. Contain trigger events of type trigger1
3. Are from runs with more than 10,000 events

Using a symbolic name avoids having to remember exactly which files were in this set. Something like:

```
trigger1_01MAY14-01June14_greater10k
```

There can be hundreds, thousands, or hundreds of thousands of files that match a given selection criteria. The total size can be hundreds of gigabytes and the event count can be in the millions.

Often one person or a small group of people sets up the datasets for each experiment. Most people only need to know how to access datasets, not how they are created.

## 3.2.2 Snapshot

Once a dataset definition is created, it is possible to search for the relevant files and store the result as a snapshot. A SAM snapshot is the actual list of files that satisfy the metadata

query at a particular point in time, i.e. when a particular analysis was run. When this dataset definition is required in the future, the snapshot can be referred to instead of carrying out the search a second time, which could lead to a different set of files. Using the same snapshot makes it possible to compare a current analysis with an earlier analysis and be assured of having the same files. Also, using snapshots can reduce the load on the central database, and increase the proportion of operations which can be supported by a distributed hash table, which improves the reliability and scalability.

### 3.2.3 Project

---

A SAM project represents the pool of files available for delivery, and the collection of processes pulling files from that pool. The project is not just the dataset, it is also the processes that work on the files. The project is typically started by specifying the dataset definition name, and the SAM snapshot is created at that time. The name of the project is also specified. Internal to SAM, a project is the unix process called *pmaster* running on the station node. *pmaster* responds to requests from processes and coordinates the delivery of files particular to the project.

### 3.2.4 Station

---

Since SAM monitors based on stations, users should be aware of what they do although the internal details may not be important. A SAM station is an application that coordinates all communication between projects, processes and the database, as well as all file delivery activities. The stations request and log file delivery to user projects, recording which files are stored on which disks, and managing storage space. Projects are associated with a particular station.

## 3.3 SAM Monitoring Example

---

SAM provides tables and graphs to allow users to monitor the activity on their data runs.

On the afternoon of May 6th, 2014, the following information was available on D0 from [http://samweb.fnal.gov:8480/station\\_monitor/](http://samweb.fnal.gov:8480/station_monitor/).

## D0 SAM station monitoring

Station	Projects	Last updated	Last activity
<a href="#">clued0</a>	0	2014-05-06 14:30:00	project ended at 2014-05-06 13:47:13 (42 minutes ago)
<a href="#">fnal-cabsrv1</a>	28	2014-05-06 14:30:09	file delivered at 2014-05-06 14:48:48 (<1 minute ago)
<a href="#">osg-ouhep</a>	0	2014-05-06 14:30:00	No activity in last 6 hours

Clicking on one of the stations gives how many projects are running, with how many processes, and a number of graphs and links to details about specific projects. A process represents a job that demands files from SAM.

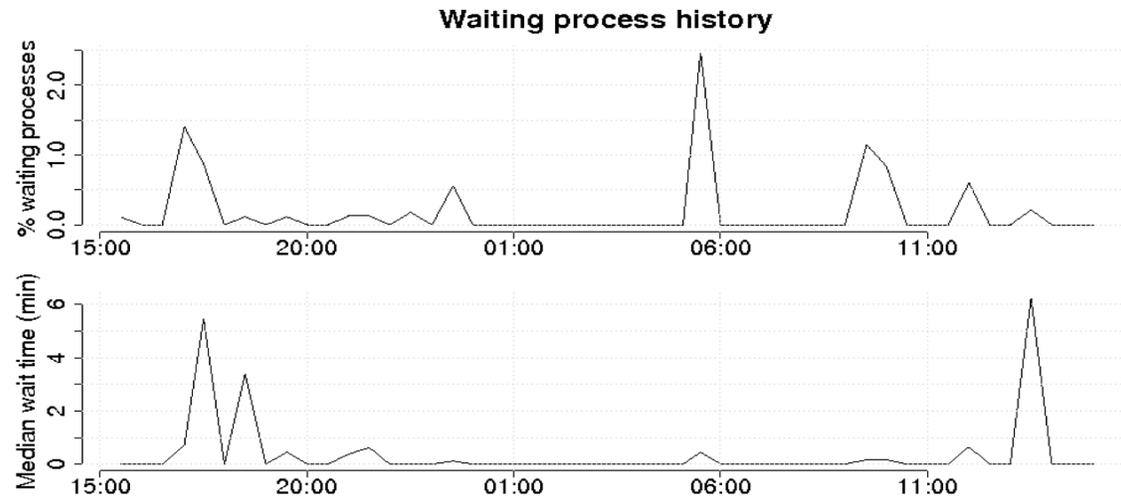
Clicking on the station that has active projects from the above picture, showed 32 running projects with 1232 processes.

### **fnal-cabsrv1 station**

Generated at 2014-05-06 15:00:11

**There are 32 running projects**

<b>Active processes</b>	1232
<b>Waiting processes</b>	0 ( 0.00% )



There is more information about the station, with links to more details. To explore, click on:

[http://samweb.fnal.gov:8480/station\\_monitor/](http://samweb.fnal.gov:8480/station_monitor/)

### **3.4 Decoupling SAM Internals from Experiment's Framework**

With samweb, experiments do not need to integrate a SAM API piece into the experiment framework to communicate with SAM. Instead, communication is performed via an http protocol. The samweb server converts the http requests into SAM requests. Advantages to this method include:

- No SAM specific code in the framework, and the experiment's framework is not directly coupled to SAM (e.g. it loads no SAM library). SAM could be replaced with some other data management system and the experiment framework and data handling code need not change.
- Intimate knowledge of SAM isn't needed to write the data handling code. The code need only know how to make certain http requests and deal with responses.
- SAM is experiment agnostic. No knowledge of the experiment framework is necessary.
- If the samweb server needs to be scaled up, more instances of it can be generated.

- SAM client can be updated without touching the experiment code since the SAM client is only deployed with the samweb server, not to the experiment.
- Deploying to remote sites is easy as the SAMWeb servers remain at Fermilab. There is nothing to deploy at remote sites.

### **3.5 Location Catalog**

---

SAM has a location catalog which stores information about where the data files can be found. The system is designed to work with a variety of storage systems including dCache or Storage Resource Management (SRM.) dCache is disk caching software used for high performance volatile storage, or in conjunction with Enstore as the high speed front end of a tape backed hierarchical storage system.

Users do not need to know file location at job submission time. They only need to provide the dataset name. The SAM portion of a processing task, called a project, is responsible for delivering all the files in the input dataset. A single SAM project can be associated with more than one batch job. When there are multiple jobs in a project, the available files are distributed dynamically across all the jobs, so that each file is given to one job only.

The file locations stored in the location catalog are mapped into concrete access URLs which are handed out to worker jobs. Individual jobs have no direct control over the order the files are delivered. Instead they request a new file and are handed a URL with which to access it. The job is responsible for the access method for the 'last-mile' access of the data files from the storage system to the worker node.

Because SAM works with datasets rather than individual files, it can command pre-staging of files from tape, or transfers from storage system to storage system, before the end user job needs to access the file. This enables more efficient file access than a purely access driven system while not requiring the user to manually determine the order of the files in the dataset. SAM handles that.

### **3.6 Adding Files to the System (Cataloging files in SAM)**

---

The File Transfer System (FTS) is an independent subsystem that handles cataloging, replicating, archiving and deleting files. The FTS interfaces with the SAM metadata catalog and data delivery systems and supports interactions with the Enstore tape archive facilities as well as Bluearc central storage.

While most users won't need to do this, it's useful to know that FTS watches specified directories for new files. When it finds new files, it catalogs them, and archives them on tape according to predefined rules. Users should follow their experiments' procedures for doing this. It is also possible to manually add files to SAM, but FTS is a better way of managing the file storage and metadata handling, although if users require the file to be in SAM immediately, they can declare the metadata and the location without having to wait for FTS to discover the file(s).

The FTS is distributed as a ups product. The latest version of FTS should be installed unless it is known that an older version is needed for compatibility reasons. The versions of FTS available can be seen with upd:

```
$ upd list -aK+ FileTransferService
```

More information is available at:

[File Transfer Service Information](#)

It is also possible to add files to SAM by hand outside of FTS with the [declare-file](#) command. But then the user must come up with the metadata, and has to tell the database where the file lives in Enstore or dCache with the [add-file-location](#) command. (FTS does this automatically, so is the recommended way.)

## 4 Processing Files

### 4.1 SAM Project Example

---

A project manages delivery of a dataset to one or more jobs. Each individual job independently requests files from the project, continuing until there are none left. To generate the unique project name, usually some combination of username and time is used.

1. Generate unique project name.
2. Start project.
3. Start consumer process
4. File loop
  1. Get location (uri) of next file.
  2. Copy file to scratch disk.
  3. Process file.
  4. Release file.
  5. Delete file from scratch disk.
5. Stop consumer process.
6. Stop project.

Files requested by a project are delivered from (a subset of) the locations known to the SAM catalog, as specified by the routing protocol for that station, to the station cache, which may be a set of physical disks mounted on one machine, or a distributed cache consisting of disks on a set of nodes. Files delivered to station cache are temporarily protected from deletion until the consumer which needs them has issued a signal to release them. No files are replaced in the station cache until a new project request does not have space, and then files are deleted according to a programmable policy (currently, least recently used). The information about which files have been successfully delivered to the project is reported back to the catalog and stored. Files can be pinned in a station

cache; that is, marked as unavailable for deletion until an administrator of the system issues a command to unpin them.

The ifdh commands in bash or python are available at:

<https://cdcv.s.fnal.gov/redmine/projects/ifdhc/repository/revisions/master/entry/demo.sh>

<https://cdcv.s.fnal.gov/redmine/projects/ifdhc/repository/revisions/master/entry/demo.py>

#### 4.1.1 Batch Project Example

---

Can use batch jobs to process a dataset of files by using the Directed Acyclic Graph (DAG) feature of Condor/jobsub to serialize start project, worker, and stop project batch jobs. This assumes you understand how to use [jobsub](#).

Users can write scripts to process a dataset of files to start and stop a SAM project with leader/trailer jobs, and a variable number of worker jobs.

The specifics are experiment-dependent, but each worker job should do something like the following:

1. find ifdhc, and use it to...
  1. find its project url
  2. establish itself as a consumer
  3. getNextFile in a loop
    - 3.1. fetch the file locally with fetchInput
    - 3.2. say if the fetch went okay
    - 3.3. run whatever command is wanted (experiment-specific) on the file
      - 3.3.1. if it succeeds, set it as consumed, and add any output files
      - 3.3.2. if it fails, mark it skipped, and set overall status to bad
  4. go back around for the next file
2. when out of files, copy back the output, and end the consumer process.

Note: It's a good idea to report status as "ok" or "bad" before exiting, otherwise the system will time out and declare the process as bad, which factors into recovery datasets. By reporting success or failure, that status can be used to make a recovery dataset if needed.

If an experiment is using the art framework, there is a 'art\_sam\_wrap.sh' script in ifdh\_art that does all of this and more.

#### 4.1.2 Using SAM Interactively

---

Users can start a project and run a project as described above to do local analysis.

```
%ifdh startProject projectname station-name dataset-name user group
(where station-name and group are experiment name)
```

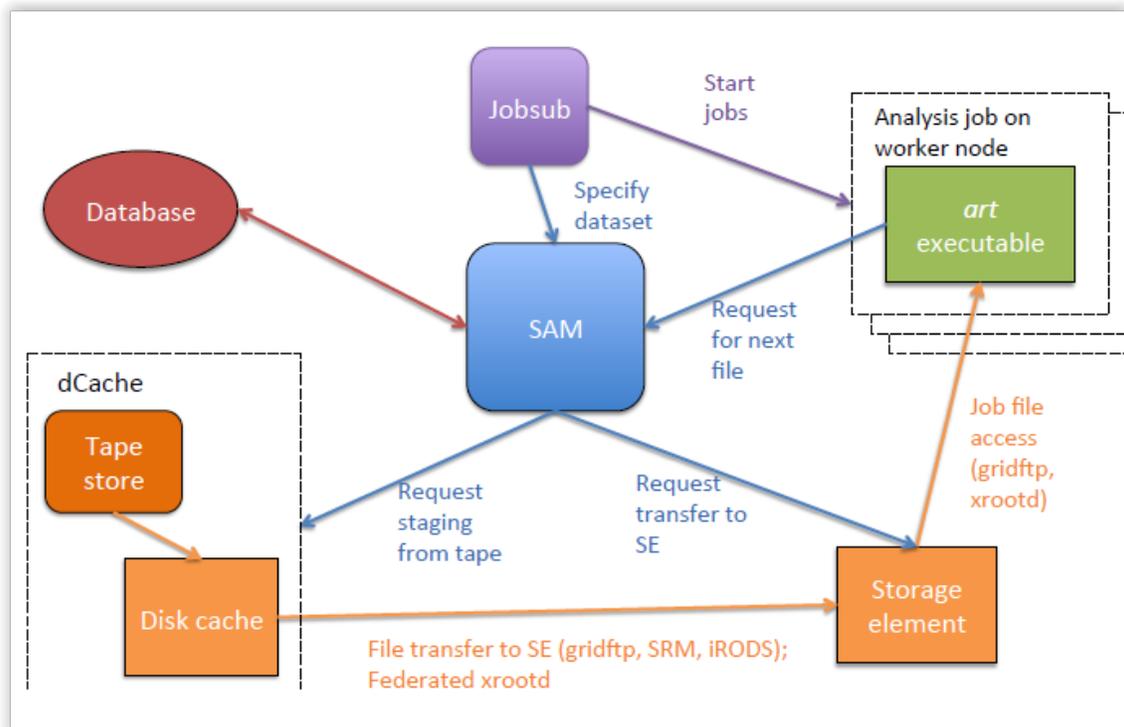
Do the same stuff as described above, and listed in [sample samjob.sh script](#) to do a local analysis.

End with

```
%ifdh endProject url
(where the url is the one you received from startProject)
```

Note that the 'scratch disk' directory used by ifdh is defined by the environment variable TMPDIR. For interactive use, define this by hand to avoid filling up /var/tmp on the gpvm nodes.

## 5 Integrating SAM with Experiments



Note: the box labeled 'art' in the picture could be an experiment-specific framework instead of art.

## 5.1 *art* and SAM

---

The `ifdh_art` package provides *art* service access to the libraries from the `ifdhc` package, which is the Intensity Frontier data handling client code. This enables users to specify SAM metadata configuration in `.fcl` files and submit jobs via `jobsub`.

For example:

```
if using art with SAM, you need these entries,
OR the --sam-* command-line options
user.services.IFDH: {IFDH_BASE_URI:
"http://samweb.fnal.gov:8480/sam/minerva/api"}
user.services.CatalogInterface: {
  service_provider: "IFCatalogInterface" }
user.services.FileTransfer: {
  service_provider: "IFFileTransfer" }
outputs.out1.dataTier: "raw"
process_name: "whatever"
services.FileCatalogMetadata: { applicationFamily: "demo"
                                applicationVersion: "1"
                                fileType: "importedDetector"
                                }
# if you use either of IFBeam or nucondb services, include the
# respective entry below.
user.services.IFBeam: {}
user.services.nucondb: {}
```

There are other ways to set the `.fcl` files, but assuming you've done that, then you can use `jobsub` as follows

```
1 jobsub \
2   -g \
3   -r S12-11-16 \
4   -N 6 \
5   --dataset_definition=my_dataset_name \
6   $IFDH_ART_DIR/bin/art_sam_wrap.sh \
7   -X nova \
8   -c /path/to/my/config.fcl \
9   --rename uniq \
10  --dest /grid/data/$USER
```

This

- 1) submits a DAG of jobs which
- 2) are grid jobs, and
- 3) use release S12-11-16,
- 4) has 6 nodes running to analyze the files in the
- 5) `my_dataset_name` dataset using
- 6) the `art_sam_wrap.sh` wrapper to run the art
- 7) `nova` executable with a
- 8) specified `fcl` file and asks

- 9) that the output file be renamed to be unique, and
- 10) copied back to the users nova data area.

This is normally done in concert with:

```
jobsub --dataset_definition=$d -N $n --project_name=$p ...
```

It does a sam\_begin job that does #2, \$n worker jobs that should do #3..5, and then a trailer job that does #6.

If the samweb command-line options are used, many of the entries specified in .fcl files will be added automatically. The art\_sam\_wrapper.sh script will similarly add options to the configuration before running the program.

For more information, see: <https://cdcvs.fnal.gov/redmine/projects/ifdh-art/wiki>

## 6 SAM Reference Material

---

### 6.1 Glossary

Bluearc - central storage

DAG - Direct Acyclic Graph is a Condor feature that allows for dependencies in job scheduling. For example, with a DAG one can ensure that job A runs to completion before jobs B, C, and D start and then job E starts only when the previous jobs all complete.

Dataset - a metadata query that resolves to a desired set of input files for processing. Dataset definitions may be created by SAM client tools on the command line or with experiment-specific web-based tools. Users build a list of metadata parameters to define a collection of like files, the dataset.

dCache - dCache is disk caching software developed jointly by DESY, Fermilab and NGDF. dCache can be used independently for high performance volatile storage, or in conjunction with Enstore as the high speed front end of a tape backed hierarchical storage system.

Enstore - Enstore is the mass storage system developed and implemented at Fermilab as the primary data store for scientific data sets. It provides access to data on tape to/from a user's machine on-site over the local area network, or over the wide area network through the dCache disk caching system.

FTS- The File Transfer System (FTS) is an independent subsystem that handles cataloging, replicating, archiving and deleting files. The FTS interfaces with the SAM metadata catalog and data delivery systems and supports interactions with the Enstore tape archive facilities as well as Bluearc central storage.

ifdh - Intensity Frontier Data Handling (ifdh) is a suite of tools for data movement tasks for Fermilab experiments. It encompasses moving input data from caches or storage

elements to compute nodes (the "last mile" of data movement) and moving output data potentially to those caches as part of the journey back to the user. IFDH is called in job scripts (e.g. "ifdh cp"), hiding the low-level data movement.

kx509 - By default the authentication mechanism that is used to communicate with the server is a kx509 certificate based authentication. To protect against unauthorized access to Fermilab computers, the Computing Division has implemented Kerberos to provide what is known as strong authentication over the network.

Metadata - data about the data, used to specify the properties of the files. For example: `data_tier = reconstructed` (only want reconstructed files), or `nova.detectorid = fd` (only want far detector files).

Playlist - some projects refer to a set of data files that an analysis might want to analyze as one entity as a playlist.

Process - represents a job that demands files from SAM. The job starts the SAM process by specifying the parent SAM Project name. Using this mechanism, a collection of jobs can work in parallel and pull files from the Project because SAM ensures that a particular file goes to one and only one process. A job may need to pull files from more than one pool (e.g. different types of overlay events). In this case, it is reasonable for a job to start multiple processes under different projects. Projects, in this example, would represent pools of different types of overlay events.

Project - represents the pool of files available for delivery and the collection of Processes pulling files from that pool. The Project is typically started by specifying the dataset definition name, and the SAM snapshot is created at that time. The name of the SAM Project is also specified. Internal to SAM, a Project is a unix process called *pmaster* running on the SAM station node. *pmaster* responds to requests from SAM processes and coordinates the delivery of files particular to the Project.

SAM - Sequential Access via Metadata (SAM) is a data handling system to store and retrieve files and associated metadata, including a complete record of the processing which has used the files.

Snapshot - a list of files that satisfy a SAM dataset definition at a particular point time.

Station - is an application that coordinates all communication between projects, processes and the database as well as all file delivery activities.

## 6.2 References

1. The wiki for the SAM User's Guide at: [https://cdcvs.fnal.gov/redmine/projects/sam/wiki/User\\_Guide\\_for\\_SAM](https://cdcvs.fnal.gov/redmine/projects/sam/wiki/User_Guide_for_SAM)
2. The main SAM wiki at: <https://cdcvs.fnal.gov/redmine/projects/sam/wiki>
3. SAMweb Command Reference at: [https://cdcvs.fnal.gov/redmine/projects/sam-main/wiki/Sam\\_web\\_client\\_Command\\_Reference](https://cdcvs.fnal.gov/redmine/projects/sam-main/wiki/Sam_web_client_Command_Reference)
4. "A Data Handling System for Modern and Future Fermilab Experiments" by Robert Illingworth at:

- <https://indico.cern.ch/event/214784/session/5/contribution/440/material/slides/0.pdf>
5. "SAM at the Intensity Frontier" at: [http://uscms-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=4325;filename=samweb\\_samfs.pdf;version=1](http://uscms-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=4325;filename=samweb_samfs.pdf;version=1)
  6. Intensity Frontier Computing Model and GRID Tools Requirements and Design Documents at: <http://uscms-docdb.fnal.gov/cgi-bin/ShowDocument?docid=4082>
  7. "SAMGrid Integration of SRMs" at: <http://www.dcache.org/manuals/chep04/chep04.robert.paper.pdf>
  8. "SAM and art integration" at: <http://uscms-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=4684;filename=SAM%20and%20ART%20integration.pdf;version=3>
  9. SAM Web Cookbook (Nova version) at: [https://cdcvs.fnal.gov/redmine/projects/nova\\_sam/wiki/SAM\\_web\\_cookbook](https://cdcvs.fnal.gov/redmine/projects/nova_sam/wiki/SAM_web_cookbook)
  10. SAMweb Interface definitions at: [https://cdcvs.fnal.gov/redmine/projects/sam-web/wiki/Interface\\_definitions](https://cdcvs.fnal.gov/redmine/projects/sam-web/wiki/Interface_definitions)
  11. SAM project meeting information at: <https://sharepoint.fnal.gov/org/scd/fife/samProjectMeeting/default.aspx>