

**ARDA Support for CMS Analysis Process (ASAP)
user guide**

Olga Kodolova

**Many thanks to Julia Andreeva and Craig Munro that supported ASAP
and to Alexander Lanev for careful testing and feedback**

The instructions from Alexander are at the end of the presentation

ASAP modules

- 1) Job creation and submission is located on User Interface (UI)
- 2) Server part is optional. Server part may take responsibility on user job submission/resubmission

User Guide is located:

<https://lxarda13.cern.ch/docs/>

Contact: arda-support-cms@cern.ch

Craig.Munro@cern.ch

Julia.Andreeva@cern.ch

User job definition

Two classes of jobs can be submitted with ASAP tools:

- data are registered in CMS DBS/DLS system (standard user job)**
- private user job: any job**
the limitation is that ASAP uses scramv1(or scram) environment

For both types of jobs one has to create directory, which is opened for r/w for everybody (tracking directory).

Example: mkdir path_to_directory/asap
chmod a+w path_to_directory/asap

Example: Create CMSSW project area

1) User introduces changes in the library (development)

```
scramv1 project -n cmssw103 CMSSW CMSSW_1_0_3
```

```
cd cmssw103/src
```

```
cvs co -r JetMETCorrections/JetPlusTrack
```

```
cd JetPlusTrack
```

```
Edit *.h, *.cc
```

```
scramv1 build
```

```
eval `scramv1 runtime -csh`
```

2) User uses default library

```
scramv1 project -n cmssw103 CMSSW CMSSW_1_0_3
```

```
cd cmssw103/src
```

```
eval `scramv1 runtime -csh`
```

Ready to start with ASAP submission

Standard user job

Requires two files:

- CMSSW cfg file (standard one, f.e. JetCorrectionTest.cfg or allReco.cfg)
- configuration file for asap – task.conf

Example of task.conf

specify directory to store tasks (tracking directory, see previous slide)

jobdir = /afs/cern.ch/user/k/kodolova/scratch0/asap

Store output at SE (otherwise output will go to SandBox-limited size)

store_output = 1

specify grid to submit to (lcg, my)

grid = lcg

specify dataset file

primary_dataset = CSA06-103-os-ExoticSoup0-0

tier = FEVT

processed_dataset = CMSSW_1_0_6-su_Exotics_ZprimeFilter-1161788458

Standard user job (continuation)

pset file (CMSSW configuration file)

pset_file = JetCorrectionTest.cfg

specify output sandbox, could be filename, a comma sep list of filename or *

output_files = CorrectedJets.root

total number of events you wish to process (if unspecified assume all)

events_required = 20

events per job

events_per_job = 10

specify minimum time requirements for the job

min_wall_clock_time = 100

min_cpu_time = 100

Additional GRID parameters:

good_se

bad_se

Private user job

Requires four files:

- **pset_file** - CMSSW cfg file (standard one, f.e. JetCorrectionTest.cfg or allReco.cfg)
- **configuration file for asap** – task_private.conf
- **dataset_file** - file with parameters, that has to be changed from job to job
- **template_file** – sh or csh script, that has to be run at working node (WN)

Private user job

template.sh file:

```
#!/bin/sh
```

```
# if one need to put some additional large file at working node, put this file  
to # any of Storage Elements (SE) in advance and use lcg_cp command  
# in template.sh
```

```
lcg-cp --vo cms lfn:/my_lfn file:///`pwd`/my_lfn
```

```
cmsRun --parameter-set `pwd`/JetCorrectionTest.cfg
```

```
# instead of cmsRun, one can use any other executable, that is transferred  
# to working node through SE.
```

Private user job

Examples of dataset_file:

Format: $\{\text{FILENAMES}\}$, $\{\text{SKIP_EVENTS}\}$, $\{\text{MAX_EVENTS}\}$,
ATTRIBUTE_1,...,ATTRIBUTE_n

1) there are root files (located in the area mounted to UI)

myroot_file_1.root 0 100 345 9329

myroot_file_2.root 100 100 1345 93290

myroot_file_3.root 200 100 2345 13290

3 tasks will be created

2) there are not root files

- 0 100 345 9329

- 100 100 1345 93290

- 200 100 2345 13290

- 300 100 1111 345

4 tasks will be created

Private user job

Usage of `#{FILENAMES}`, `ATTRIBUTE_1`,..., `ATTRIBUTE_n`

Part of `CMSSW.cfg` files

```
source = PoolSource {  
    untracked vstring fileNames = {#{FILENAMES}}  
}  
service = RandomNumberGeneratorService  
{  
    untracked uint32 sourceSeed = ATTRIBUTE_1  
    PSet moduleSeeds =  
    {  
        untracked uint32 VtxSmear = ATTRIBUTE_2  
    }  
}
```

Private user job

Example of task_private.conf

specify directory to store tasks

jobdir = /afs/cern.ch/user/k/kodolova/scratch0/asap

Store output at SE

store_output = 1

good_se = grid100.kfki.hu

specify grid to submit to (lcg, my)

grid = lcg

specify dataset file

dataset_file = SingleJets_GEN_SIM_DIGI_asap.dataset

pset file (CMSSW configuration file)

pset_file = SingleJets_GEN_SIM_DIGI_asap.cfg

template file (shell script to run at WN)

template_file = template.sh

Private user job

specify output sandbox, could be filename, a comma sep list of filename or *

output_files = *.root

specify minimum time requirements for the job

min_wall_clock_time = 100

min_cpu_time = 100

Job submission

1. Setup environment

```
source /afs/cern.ch/sw/arda/install/CMS/asap3/setup.{sh,csh}
```

2. Create voms proxy

```
voms-proxy-init -voms cms
```

3 Prepare job – see previous slide. Preparation depends whether the job uses DBS/DLS or not.

4. cd project_area -> your test area created with scramv1 (see previous slides)

5. eval `scramv1 runtime -csh`

6. asap –config task.cong create

- asap package users project area (libraries, executables to transfer to GRID)
- check DBS/DLS if it is required
- provide TaskID -> all other actions has to be performed to TrackID

Job submission

What you see on the screen:

```
$ asap --config task.conf --create  
* starting asap3, fasten your seatbelts  
* using SCRAM environment  
* application: CMSSW_1_0_0  
* created 10 jobs with task id 21767
```

7. Submit manually

```
$ asap --taskid TASKID --submit
```

Job submission

8. Monitor status

```
$ asap --taskid TASKID --update --list
```

```
$ asap --taskid 21767 --list
```

* starting asap3, fasten your seatbelts

Job	ASAP Status	GRID Status	GRID Reason
-----	-------------	-------------	-------------

1	SUBMITTED	Scheduled	Job successfully submitted to Globus
2	SUBMITTED	Scheduled	Job successfully submitted to Globus
3	SUBMITTED	Scheduled	Job successfully submitted to Globus
4	SUBMITTED	Scheduled	Job successfully submitted to Globus
5	SUBMITTED	Scheduled	Job successfully submitted to Globus
6	SUBMITTED	Scheduled	Job successfully submitted to Globus
7	SUBMITTED	Scheduled	Job successfully submitted to Globus
8	SUBMITTED	Scheduled	Job successfully submitted to Globus
9	SUBMITTED	Scheduled	Job successfully submitted to Globus
10	SUBMITTED	Scheduled	Job successfully submitted to Globus

Job submission

Kill Jobs. If you discover a problem with your jobs you can cancel them.

```
$ asap --taskid TASKID --kill
```

Fetch output

```
$ asap --taskid TASKID --fetch
```

```
$ asap --taskid TASKID --fetch --output_dir MY_NEW_OUTPUT_DIR
```

```
$ asap --taskid TASKID --fetch-logs
```

```
$ asap --taskid TASKID --fetch-data
```

Resubmit

```
$ asap --taskid TASKID --submit
```

Job submission with monitor

Follow instructions up to job submission. Then instead for job submission:

1. Delegate your proxy

```
$ asap-user-register
```

2. Register jobs with Monitor

```
$ asap --taskid TASKID --register
```

ASAP will submit and resubmit jobs to GRID upon completion the output

3. Before retrieving the output one need to unregister job

```
$ asap --taskid TASKID --unregister [--output_dir a_new_output_dir]
```

Monitor status with cmd line

```
$ asap --taskid TASKID --list
```

```
$ asap --taskid 21767 --list
```

* starting asap3, fasten your seatbelts

Job	ASAP Status	GRID Status	GRID Reason
-----	-------------	-------------	-------------

```
-----  
1  FETCHED *  Done (Success)  Job terminated successfully  
2  FETCHED *  Done (Success)  Job terminated successfully  
3  FAILED *   Done (Failed)   Aborted...  
4  DONE *     Done (Success)  Job terminated successfully  
5  FETCHED *  Done (Success)  Job terminated successfully  
6  FETCHED *  Done (Success)  Job terminated successfully  
7  FETCHED *  Done (Success)  Job terminated successfully  
8  FETCHED *  Done (Success)  Job terminated successfully  
9  FETCHED *  Done (Success)  Job terminated successfully  
10 FETCHED *  Done (Success)  Job terminated successfully  
-----
```

* - job is registered with the monitor

Monitor status with web page

Before looking at the task one need to insert the GRID certificate in the Browser, following instructions

<http://lcg.web.cern.ch/LCG/users/registration/load-cert.html>

To follow jobs use Dashboard web page:

<http://arda-dashboard.cern.ch/cms/>

Использование ASAP для запуска задач на GRID

Александр Ланев, ЛФЧ ОИЯИ

18.01.2007

1. ASAP (Arda Support for CMS Analysis Processing) - это одна из систем запуска задач на GRIDe для пользователей CMS. Более распространенной является система CRAB
<http://cmsdoc.cern.ch/cms/ccs/wm/www/Crab/>
(Имеется tutorial <http://indico.cern.ch/conferenceDisplay.py?confId=8814#17>)

Описание ASAP можно найти на

/afs/cern.ch/sw/arda/install/CMS/asap3/docs/html/user_guide/index.html

или на <https://lxarda13.cern.ch/docs/index.html>

Разработчики ASAP, которым можно задавать вопросы если что-то не работает:

Craig Munro <craig.munro@cern.ch>

Julia Andreeva <andreeva@mail.cern.ch>

2. Установка на Ixplus

Предполагается, что уже установлена CMSSW

(например, версия CMSSW_1_2_0 находится в директории ~/scratch0/CMSSW_1_2_0).

Специальной установки ASAP не требуется, просто загрузите определения:

```
cd ~/scratch0/CMSSW_1_2_0/src
```

```
eval `scramv1 ru -csh`
```

```
source /afs/cern.ch/sw/arda/install/CMS/asap3/setup.csh
```

и создайте директорию:

```
mkdir ~/scratch0/asap
```

```
cd ~/scratch0/asap
```

Для работе в GRID необходимо зарегистрироваться

в виртуальной организации CMS и создать проху сертификат

(лучше указать достаточное время, например, 200 часов):

```
voms-proxy-init -voms cms -hours 200
```

3. Запуск задачи на GRIDe

Задайте файл конфигурации задачи `task.conf` и создайте задачу:

```
asap --config task.conf --create
```

Пример задачи переменных в файле `task.conf`

```
jobdir = /afs/cern.ch/user/l/lanyov/scratch0/asap
```

```
primary_dataset = CSA06-103-os-EWKSoup0-0
```

```
tier = FEVT
```

```
processed_dataset = CMSSW_1_0_4-su_BSM_mc2e_Filter-1161045561
```

```
pset_file = cmssw_configuration_file.cfg
```

```
output_files = histograms.root
```

```
events_per_job = 500
```

Команда

```
asap --list
```

позволяет узнать какие ваши задачи известны ASAP.

В первой колонке она выдает `TaskID`, который будет использоваться для дальнейшей

работы и который удобно задать в качестве переменной окружения:

```
set task=2318 (в csh/tcsh)
```

```
task=2318 (в sh/zsh/bash)
```

Для реального запуска job'ов необходимо выдать:

```
asap --taskid $task --submit
```

Узнать статус job'ов можно командой

```
asap --taskid $task --update --list
```

После того как все или хотя бы часть job'ов закончатся, необходимо забрать результаты и листинги работы:

```
asap --taskid $task --fetch
```

Они будут копироваться в директорию, указанную как jobdir в конфигурационном файле.

Ненужные задачи можно удалить командой

```
asap --taskid $task --delete
```

(при этом сотрутся и все файлы из директории jobdir для этой задачи).

Интерактивный мониторинг запуска задач удобно делать с помощью CMS dashboard

(автор - Julia Andreeva): <http://arda-dashboard.cern.ch/cms/>

- кликните "Interactive view" и выберете свои задачи в меню "any user".

4. Резюме по опыту использования ASAP

На основании опыта запусков ASAP в декабре 2006 г. можно сказать, что успехом завершаются около 60% попыток запуска.

В остальных случаях выдаются ошибки traceback из программы Python.

Иногда это было связано с ошибками в ASAP, которые оперативно исправлялись

его разработчиками (см. e-mail в начале). Иногда проблемы возникали из-за того,

что не работали какие-то сервера, необходимые для запуска задач в GRID (в этом случае запуск в CRAB тоже не работал и тоже с таинственной ошибкой).

Как правило, это возникало в выходные и праздники.