Table of Contents

CRAB Client: Use, Deployment, Maintenance

Complete:	Go to SWGuideCrab

CRAB Client is now deployed by CMSSW automatic release build. A new stable release is made any time the CMSSW master release is updated (typically every 3-4 weeks) and includes whatever version of the client is indicated in the `spec` files for `prod`, `pre` and `dev` versions, which are independently controlled (by us).

The cmsbuild robot build updates those three crab versions every night in a "parallel universe", where things can be tested, but be aware that those are the same `spec` files as the actual release build, so be very careful before changing `crab-prod.spec`.

More on the proper procedure to care for the `spec` files is below

Using CRABClient

Using CRABClient last release in CMSSW distribution

1. get a proxy

```
voms-proxy-init -rfc -voms cms -valid 192:00
export X509_USER_PROXY=`voms-proxy-info -path`
```

2. setup CMSSW

```
cd CMSSW..
cmsenv
```

- note: if you do not have a CMSSW release at hand, just create any recent one, and it will do:

```
cmsrel CMSSW_10_4_0
cd CMSSW_10_4_0
cmsenv
```

3. point to your favourite version executing ONE of these

```
crab         # which is an alias for  crab-prod
crab-prod
crab-pre
crab-dev
```

Usually users should stick to `crab` i.e. `crab-prod` unless they want to test, or need to use, some new feature or bug fix only available in `pre` or `dev`.

Using CRABClient from nightly CMSSW Integration Build

There are two ways:

1. to simply use new code (but not all subtle dependencies like command autocompletion) do this at any time:

   ```
   export PATH=/cvmfs/cms-ib.cern.ch/latest/common:$PATH
   ```
2. to fully use the IB environment (including new CMSSW builds etc.) do this before `cmsenv` (or `cmsrel`):

   ```
   source /cvmfs/cms-ib.cern.ch/latest/cmsset_default.sh
   ```

Using CRABClient API

In order to use the API, besides setting up `cmsenv` as above, you need to source this script.

```
source /cvmfs/cms.cern.ch/common/crab-setup.sh
```

and make sure that your python script has this before any other `import` statement:

```
import CRABClient
```

That script can be safely sourced at all times, even if you do not plan to use the APO.

By default that will point you to the `prod` instance of the CRABClient. Otherwise the instance can be indicated as argument:

```
source /cvmfs/cms.cern.ch/common/crab-setup.sh prod
source /cvmfs/cms.cern.ch/common/crab-setup.sh pre
source /cvmfs/cms.cern.ch/common/crab-setup.sh dev
```

Or simply

```
source /cvmfs/cms.cern.ch/common/crab-setup.sh -h
```

If you are using the nightly IB, you should use instead

```
source /cvmfs/cms-ib.cern.ch/latest/common/crab-setup.sh
```

Examples

the following, usually working, scripts give concrete examples of how to setup things, I usuall execute one of this via a short alias every time I login for CRAB action. My alias also grabs a voms proxy.

```
/afs/cern.ch/user/b/belforte/public/CRAB/SETUP
/afs/cern.ch/user/b/belforte/public/CRAB/SETUP-IB
/afs/cern.ch/user/b/belforte/public/CRAB/SETUP-LOCAL
```

Using DBS Python API

DBS Python API is deployed by CMSSW as a CRABClient dependency. Therefore in order to use it in a standalone script, rather than inside CRABClient, you need to setup CMSSW via `cmsenv` command and execute the `crab-setup.sh` as indicated above. After that you can use the usual thing in your python scripts:

```
from dbs.apis.dbsClient import DbsApi
```

Develop for CRABClient

Using CRABClient directly from your GitHub repository

All binary dependency for CRABClient are now in CMSSW, therefore you only need to point to the python code and its python dependencies:

to do once:

1. go to your favorite development place

```
cd <some_dir>
GitDir=`pwd`
```

2. clone repositories and make sure to checkout proper versions (see example beflow for WMCore and DBS). Master HEAD may not always work, but if you want to develop code yoy may have to start there. In addition to CRABClient you will need DBS, WMCore and a couple files from CRABServer. If you want to eventually make git Pull Requests, you will fork first and clone from your fork the repository you want to work on. If that's the case you surely know how to modify following commands. If you simplt want to look around, debug things, or just use latest code, copy/pasting the following should do. But beware that WMCore versions change frequently. You can check `crab-*.spec` files in cmsdist to see which WMCore version is used with current CRABClient.

```
git clone https://github.com/dmwm/CRABClient
git clone https://github.com/dmwm/CRABServer
cp CRABServer/src/python/ServerUtilities.py CRABClient/src/python/
cp CRABServer/src/python/RESTInteractions.py CRABClient/src/python/
git clone https://github.com/dmwm/WMCore
cd WMCore; git checkout 1.3.3; cd ..
git clone https://github.com/dmwm/DBS
cd DBS; git checkout 3.10.0; cd ..
```

to do at each login

1. get a proxy

```
voms-proxy-init -rfc -voms cms -valid 192:00
export X509_USER_PROXY=`voms-proxy-info -path`
```

2. setup CMSSW

```
cd CMSSW..
cmsenv
```

3. execute these lines

```
GitDir=<the_directory_where_you_cloned>
MY_DBS=${GitDir}/DBS
MY_CRAB=${GitDir}/CRABClient
MY_WMCORE=${GitDir}/WMCore

export PYTHONPATH=${MY_DBS}/Client/src/python:${MY_DBS}/PycurlClient/src/python:$PYTHONPATH
export PYTHONPATH=${MY_WMCORE}/src/python:$PYTHONPATH
export PYTHONPATH=${MY_CRAB}/src/python:$PYTHONPATH

export PATH=${MY_CRAB}/bin:$PATH
source ${MY_CRAB}/etc/crab-bash-completion.sh
```

you can see a complete, usually working, example in
`/afs/cern.ch/user/b/belforte/public/CRAB/SETUP-LOCAL`

Managing Versions and dependencies via CMSDIST

Dependencies

CRAB Client needs WMCore, needs a couple files from CRABServer (Client and Server need to stay aligned on some defintions and settings) and carries a dependency from DBS for easy distribution of DBS Python API. The way to manage such dependencies is described in https://twiki.cern.ch/twiki/bin/view/CMSPublic/NotesAboutReleaseManagement#Cross_service_dependencies

Where and What

CRAB Client spec files are now in https://github.com/cms-sw/cmsdist

- You must use the master branch, **not** the `comp_*` branch !!! If in doubt check with CMSSW release experts. Note that in `cmsdist` repository there is no branch called simply `master`, and the master branch changes with time. It is usually the default one when you visit https://github.com/cms-sw/cmsdist and has a name like `IB/CMSSW_11_2_X/master`

- There are only 3 files which you can/should modify.
- You must change in your private fork and make a Pull Request
- At time of writing this twiki those are:
  - https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_2_X/master/crab-dev.spec
  - https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_2_X/master/crab-pre.spec
  - https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_2_X/master/crab-prod.spec

Those files all have a content like this (showing `crab-dev.spec` here):

```
############## IMPORTANT #################
#For new crabclient_version, set the version_suffix to 00
#For any other change, increment version_suffix
##########################################
%define version_suffix 01
%define crabclient_version v3.200531
### RPM cms crab-dev %crabclient_version.%{version_suffix}
%define wmcore_version     1.3.3
%define crabserver_version v3.200531
%define dbs_version        3.12.
```

Numbers there refer to tags in github. For CRAB the tag is always v3.YYMMDD. Usually you wll simply want to change the CRAB Client release number. Example:

```
from:
%define crabclient_version v3.200531
to:
%define crabclient_version v3.200620
```

- If needed, we can also modify the following two lines for the WMCore and CRABServer versions respectively:

```
%define wmcore_version 1.3,3
%define crabserver_version v3.200531
```

It is very rare that DBS Client version needs to change, but it is always better stay with a named version than with repo HEAD

- **IMPORTANT NOTE** (see also the comments at the beginning of the spec file) if you want to change **only** the WMCore or CRABServer version or anything in the build, but not CRABClient tag, then you MUST increment the `version_suffix`

When and How

You need to wait for the CMSSW/cmsdist admins (Shahzad) to trigger testing of the Pull Request via github comments which act as command for the build robot. Since CRAB changes can't bread CMSSW they usually approve everything unless the build fails, which you need to fix. Communications with CMSSW admins is via comments in the Pull Request. Once build is successful, admins will merge the Pull Request.

Changes to the `crab-*.spec` files are propagated nightly to the Integration Build and can the tested using the procedure indicated at the top Instead deployment to CVMFS common directory, for defaut use is only done when a new release is build for the CMSSW version indicated in the name of the master branch, i.e. the most recent release, which happens about every 3 or 4 weeks.

You can consult the calendar for scheduled CMSSW releases to know the planned dates.

Corollary:

- the CRAB Client release for users can only be updated about once/month
- the CRAB Client release for users must be well tested (in `dev` and `prod`) because there is no way to roll back (well.. if a major disaster strick Shahzad and Bockjoo can hack things by hand, but let's avoid it).

The ideal procedure is

1. develop and test locally as indicated above
2. when confident, make a new release in github and update the crab-dev spec
3. the day after you can start checking that build went well and that everything works in IB
4. when confident, port the change to crab-pre.spec
5. it there's no urgency, wait for this to be deployed on "lxplus" for more extensive checking, e.g. involving users
6. finally update crab-prod-spec
    - ♦ at same time it may be a good idea to set crab-pre.spec to same old version as crab-prod was, so if there is a disaster users can roll back by replacing crab with crab-pre command (`pre` could mean either `preprod` or `previous` !)

NEW Do it yourself CRABClient deployment:

It is now possible for us to deploy a CRABClient update on CVMFS at any time ! While it will still happen that when the top CMSSW series is updated, CRABClient is also deployed, since it is now a dependency in CMSSW.

how to do it

```
note: slithgly edited by stefano
On 04/05/2021 18:38, Malik Shahzad Muzaffar wrote:

    https://cmssdt.cern.ch/jenkins/job/cms-build-package/  jenkins job should allow you to build/
    Daina and Stefano are amdinistrator for this job and can run it. All you need is to

 - select "crab" as "CMS_PACKAGE"
 - select "UPLOAD" if you want to upload the newly built package otherwise job will only build
 - select "DEPLOY_ON_CVMFS" if you want to deploy it on cvmfs ("UPLOAD" should be selected too)
```

```
Note that for crab client you need to build "crab" which should build its deps (crab-pre, crab-d
For example if we have merged changes for crab-dev.spec only then the job will build crab-dev an

I have tested it to build/deploy SCRAMV1 and it works as expected.

Cheers, --Shahzad
```

Important additions as of June 2021

CAVEAT

Please remember, this jobs builds crab RPM and upload it to cms repository. CMSSW release build job also builds and uploads the same crab RPM. So if you run this job while we are building a CMSSW release then this can cause issues and trigger a re-build of CMSSW release. So always be carefull when you run this job. One way to check if there is any cmssw release build going on is to look for https://cmssdt.cern.ch/jenkins/job/build-release/ job. As this is deploying stuff on production CVMFS repository, so please avoid running it too often. Try to do the testing in IBs env first.

If we screw up

```
Date: Mon, 31 May 2021 14:54:59 +0200
From: Malik Shahzad Muzaffar
```

we cab also roll back users by making `crab` point to `crab-pre` instead of `crab-prod` via:

```
https://cmssdt.cern.ch/jenkins/view/CVMFS-CMS/job/cvmfs-cms-crab-symlink/ job should allow you to

Note that this only changes the /cvmfs/cms.cern.ch/common/crab symlink. For crab python api, one
```

Tricky technical points

All CRAB Clients instances use a common build file https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_1_X/master/crab-build.file

If it is changed via a Pull Request, it will affect all instances (`prod, pre, dev`) at same time. The way to test changes is to make those changes in the `crab-dev.spec` only.

see https://github.com/cms-sw/cmsdist/pull/5561#issuecomment-587293336 and https://github.com/cms-sw/cmsdist/pull/5561#issuecomment-587483867

Refs.

Some discussions about client deployment:

- https://github.com/dmwm/CRABClient/issues/4844
- https://hypernews.cern.ch/HyperNews/CMS/get/crabDevelopment/2459.html
- https://hypernews.cern.ch/HyperNews/CMS/get/webInterfaces/1379/1/2/1.html

Old ways... deprecated, obsoleted, broken....

If you only want to develop some code and do not care about installing the whole client then you should consider sourcing the production client (`source /cvmfs/cms.cern.ch/crab3/crab.sh`), cloning the client from the GitHub repository (https://github.com/dmwm/CRABClient) and then:

```
export CLIENT_HOME=/your-local-github-repository/CRABClient
export WMCORE_REPO=/your-local-github-repository/
export PYTHONPATH=$CLIENT_HOME/src/python/:~$WMCORE_REPO/WMCore/src/python/:$PYTHONPATH
export PATH=$CLIENT_HOME/bin:$PATH
```

You may also test not only your own code like this but also the PRs from other people: For how to checkout locally the PRs from the upstream repository see here: https://gist.github.com/piscisaureus/3342247

```
$ vi .git/config

[remote "upstream"]
   url = https://github.com/dmwm/CRABClient.git
   fetch = +refs/heads/*:refs/remotes/upstream/*
   fetch = +refs/pull/*/head:refs/remotes/origin/pr/*

$ git fetch upstream
$ git checkout pr/999
```

When you finish testing the PR in question go back to the original branch

```
$ git checkout master
```

Deployment using RPM

The CRAB3 client can be installed via RPM using the following instructions:

Please check http://cmsrep.cern.ch/cmssw/comp/RPMS/slc6_amd64_gcc493/ and select newest or needed version for you. If you are developing, you should be taking always the newest version.

Follow the instructions at point **4)** in NotesAboutReleaseManagement#Releasing_a_CRABClient_release_c

Deployment of CRAB client on CVMFS (production and pre-production)

The procedure is very simple thanks to the automated Bockjoo's build/install scripts. This script constantly checks for new rpm of the client, and then installs them on cvmfs. This is how you can **trigger the automatic creation** of a RPM:

- Modify the crabclient.spec file to contain the needed (newest) version. The .spec file to use is the one in the comp_gcc630 branch (used to be comp_gcc493 until September 2019) of the cms-sw/cmsdist github repo. Lets say we're ready to deploy in production the April CRABClient (3.3.1604): we change the first line of the .spec file containing the crabclient version to 3.3.1604, which can be accomplished also using the GitHub edit function in the browser. Otherwise, use the listed git commands to push the changes to your fork of cmsdist and them make a PR.

```
git clone <your-cmsdist-fork> && cd cmsdist && git checkout comp_gcc493
vim crabclient.spec
git add crabclient.spec
```

```
git commit -m "Message about commit"
git push origin
```

Changes to the crabclient.spec file:

```
-### RPM cms crabclient 3.3.1604.rc2
+### RPM cms crabclient 3.3.1604
```

- If needed, we can also modify the following two lines for the WMCore and CRABServer versions respectively:

```
%define wmcver 1.0.14_crab_4
...
%define crabserver 3.3.1604.rc1
```

- - ♦ If the specified CRABClient version is of the `3.3.1604` or `3.3.1604.patch##` format, it means this is a production version of the client, which will be deployed after the merge and point the `crab.sh` and `crab.csh` to the `3.3.1604` or `3.3.1604.patch##` client.
  - ♦ If the specified CRABClient version ends with a release candidate ("=rc=") extension, `3.3.1604.rc##`, it means this is a pre-production version, which will be deployed after the merge and point the `crab_pre.sh` and `crab_pre.csh` to the `3.3.1604.rc##` client.

- Wait for the automatic build to run and the corresponding comment to appear on the PR. If no error occurred, the message will show a "+1", the architecture version and the link to the build log.

- Add a comment with the "merge" text to merge the PR.

- Follow the instructions at point **5)** in NotesAboutReleaseManagement#Releasing_a_CRABClient_release_c

---