Electron Validation User's Guide

Most of files involved in electron validation stay in CMSSW/RecoEgamma/Examples/test (a migration to CMSSW/Validation/RecoEgamma is under work).
Therefore, any validation session should start like this :

```
> scram project CMSSW CMSSW_<release>
> cd CMSSW_<release>/src
> cmsenv
> cvs co RecoEgamma/Examples # generally best to take the very last version
> scram b
> cd RecoEgamma/Examples/test
```

The cmsRun configuration files

As compared to usual files, they have been modified so to automatically catch the list of input files from the DBS discovery web services (thanks to the script dbs_discovery.py described alter on), and they store their generated histograms into the ROOT file whose name is defined by the unix variable **$TEST_OUTPUT_FILE**. As a consequence, one cannot directly types "cmsRun <my_config_cfg.py>". One must first define the variable $TEST_OUTPUT_FILE, and the variables which are tuning dbs_discovery.py.

The cmsRun configuration files currently used are :

- GsfElectronMCAnalyzer_cfg.py: takes into account the MC data.
- GsfElectronFakeAnalyzer_cfg.py: ?.

The local script dbs_discovery.py

This python script is querying https://cmsweb.cern.ch/dbs_discovery/ so to get the list of input files.
It is configured thanks to three shell environment variables:

- **$DBS_RELEASE**: the same as the "software release" menu in the web page, for example "CMSSW_2_2_0_pre1".
- **$DBS_SAMPLE**: the same as the "primary dataset" menu in the web page, for example "RelValSingleElectronPt35".
- **$DBS_LIKE**: this shell-like regular expression is match against the full names of datasets selected by the two variables above. This typically help to filter the required datatier and relevant global tag. An example of used value is "*MC_31X_V2-v1*-RECO".

Worth to note, dbs_discovery.py can be both directly invoked from a shell command line, or

imported and called within another python script (typically a cmsRun configuration file).

Histograms post-processing

The ROOT script newvalidation.C is able to prepare plots comparing two sets of histograms. You should never need to call it directly.

The shell script **newvalidation.csh** is meant to run the ROOT script above, insert the plots in a web page, and put the web page in the relevant afs area so that they are available on the CERN web server ("/afs/cern.ch/cms/Physics/egamma/www/validation"). It is tuned by few shell environment variables and command-line arguments:

- $1 : eventual first command-line argument, immediatly duplicated into $VAL_ENV, it is a user-defined simplified name for the current sample, used to build some default value for other variables.
- $2 : eventual second command-line argument, immediatly duplicated into $VAL_OUTPUT_FILE, is the default default base name of the files containing the histograms ; it is also used to build some default value for other variables.
- $3 : eventual third command-line argument, immediatly duplicated into $VAL_WEB_SUB_DIR, it is the name of the web subdirectory. Default is "${DBS_SAMPLE}Ideal".
- **$VAL_ANALYZER** : name of the analyzer used to do the histograms ; it is used to know which histograms must be searched for. The value must be one of GsfElectronMCAnalyzer, GsfElectronDataAnalyzer, GsfElectronFakeAnalyzer or SimplePhotonAnalyzer.
- **$VAL_NEW_RELEASE** : chosen name for the new release to validate ; used in web pages and used to build the path where the web pages will be stored.
- **$VAL_REF_RELEASE** : chosen name of the old release to compare with ; used in web pages and used to build the path where the ref file will be searched for.
- $VAL_NEW_FILE : complete path of the file containing the new histograms, defaults to "./cmsRun.$VAL_ENV.olog.$VAL_OUTPUT_FILE".
- $VAL_REF_FILE : complete path of the file containing the old histograms to compare with, defaults to "/afs/cern.ch/cms/Physics/egamma/www/validation/ $VAL_REF_RELEASE/data/cmsRun.$VAL_ENV.olog.$VAL_OUTPUT_FILE".
- $DBS_SAMPLE : short chosen name for the current dataset ; used in web pages and used to build the path where the web pages will be stored.

So to see the result, go to the web page http://cmsdoc.cern.ch/cms/Physics/egamma/www/ validation/$VAL_NEW_RELEASE/vs$VAL_REF_RELEASE/$3.

Important note: each time newvalidation.csh is executed, it is copying $VAL_NEW_FILE into /afs/cern.ch/cms/Physics/egamma/www/validation/$VAL_NEW_RELEASE/data/. Later on, when $VAL_NEW_RELEASE becomes $VAL_REF_RELEASE, the default value of $VAL_REF_FILE should point to the relevant file. So the histograms are currently archived in the afs web space. Perhaps we will change this when migrating to the DQM machinery.

Trick: if you do not want to compare to anything, use the same value for

$VAL_NEW_RELEASE and $VAL_REF_RELEASE.

How to process a single sample

1. Set the DBS_* variables accordingly, and directly call dbs_discovery.py, so to check that it is returning the relevant list of input files. For example:

```
> setenv DBS_RELEASE "CMSSW_3_1_1"
> setenv DBS_LIKE "*MC_31X_V2-v1*-RECO"
> setenv DBS_SAMPLE "RelValSingleElectronPt35"
> ./dbs_discovery.py
```

2. Set $TEST_OUTPUT_FILE to any value (for example "gsfElectronHistos.root"), and call cmsRun with one of the config files modified for dbs_discovery.py, for example "cmsRun GsfElectronMCAnalyzer_cfg.py". If you want to process several samples and keep all the histograms files for some time in your current directory, you should change the value $TEST_OUTPUT_FILE before each cmsRun, or rename it afterwards. Actually, it is recommended to choose a short name for the sample (let's call it <short-sample-name>) , and call your histograms file "cmsRun.<short-sample-name>.olog.gsfElectronHistos.root". For example:

```
> setenv SAMPLE "Pt35"
> setenv ANALYZER "GsfElectronMCAnalyzer"
> setenv HISTOS "gsfElectronHistos"
> setenv TEST_OUTPUT_FILE
"cmsRun.$SAMPLE.olog.$HISTOS.root"
> cmsRun ${ANALYZER}_cfg.py
```

3. The final step is to set the variables for newvalidation.csh and run it. For example:

```
> setenv $VAL_ANALYZER $ANALYZER
> setenv $VAL_NEW_RELEASE "311"
> setenv $VAL_REF_RELEASE "310"
> ./newvalidation.csh $SAMPLE $HISTOS.root
```

How to launch the whole validation suite

The validation suite can be fully processed by the oval tool (http://oval.in2p3.fr/), as configured by the local OvalFile. The user's guide (http://oval.in2p3.fr/release/user.html) is worth spending 15 minutes so to get familiar with the tool concepts. There are also many

comments in the local OvalFile. At last, we can give here few hints for the very impatient user :

- The command "oval run cmsRun.Pt35" is first searching in OvalFile the environment defined with the tag <environment name="pt35">, then in this environment it is searching for the instructions defined with the tag <executable name="cmsRun" args="...">. Before running cmsRun with the given instructions, it is setting all the shell variables defined in the environment "Pt35" and in all the parent environments (including the top global level).
- The character "%" is the Oval wildcard. The command "oval run cmsRun.%" will search the tag <executable name="cmsRun"...> in all the OvalFile environments, and run them one after the other.

The local OvalFile has been designed so to make the validation session as simple and automatic as possible.
What we have called a <short-sample-name> in the previous section corresponds here to the OvalFile environment names.
Concretely, the operator of the validation will generally issue three commands, as described below

oval run dbs%

This run dbs_discovery.py for all the data samples to be analyzed. Then the operator can check if the list of input files seems correct. Quite often, some of the sets are empty and one must correct in OvalFile the gloval value of DBS_LIKE.
If you want to run dbs_discovery.py for a single sample defined in the OvalFile, check the corresponding environment name in OvalFile (lets' call it <env-name>), and type "oval run dbs_discovery.py.<env-name>".

oval run cmsRun%

This run "cmsRun" for all the data samples. OvalFile is written so that each cmsRun execution generate a file called "gsfElectronHistos.root", which is immediatly renamed "cmsRun.<env-name>.olog.gsfElectronHistos.root" .
If you want to run cmsRun for a single sample, type "oval run cmsRun.<env-name>".

Worth to note, Oval will automatically compare what appears on the output channel with some reference files, and eventually raise a flag if something differs very much, so that the user is aware of some eventual problem. The OvalFile enables to precisely define what lines should be compared, and what tolerance is accepted for the important quantities. Yet, the sample we generally analyze are not very stable, and the tolerance have not yet been accurately tuned, so do not worry too much if Oval is warning you about something which has changed in the output.

oval run newvalidation%

This run newvalidation.csh for all the data samples. Then the web pages are ready.
If you want to run it for a single sample, type "oval run newvalidation.csh.<env-name>".

How to process other input files

The validation suite is designed to process RelVal samples, but you can rather easily process any other files.
Just edit the usual cmsRun configuration files, such as GsfElectronMCAnalyzer_cfg.py, and replace the call to dbs_discovery.search() with your own list of files.
You must also either edit the value of "process.gsfElectronAnalysis.outputFile" in the cmsRun config file, or set TEST_OUTPUT_FILE, or even ask oval to run cmsRun for the more relevant environment.
If you want to pursue with a call to newvalidation.csh, ensure that the output file will be called something like "cmsRun.<env-name>.olog.gsfElectronHistos.root", or oval will not help and you will have to set yourself all the variables needed by newvalidation.csh .

How to process other histograms

If you want to process histograms prepared elsewhere, typically on the grid or a batch system, the problematic is rather the same as above. Either you rename your histos files accordingly to what Oval expect, and then you can call "oval run newvalidation.csh.<env-name>", or you have to set all the variables needed by newvalidation.csh and you call it directly.