Table of Contents

A tool for One-to-One comparison of EDM objects

This is a tool for event-by-event, particle-by-particle, variable-by-variable comparison.

Requirements

This needs:

Package	Tag	Integrated in Release
`Validation/Tools`	`V00-05-16`	`None yet`

or newer.

Make sure you `scram b` after updating to the correct packages. If you use *(t)csh*, you'll want to type `rehash` to pick up the latest path changes.

Important: All scripts mention here have a `--help` option that shows all available options.

Uses

This tool is designed to be used anytime we want to check that a given input file processed by two different methods gives similar results. For example, we can use this to compare:

- Comparing a PATtuple to the RECO/AOD file that was used to create it
- Comparing RECO Scientific Linux 4 to RECO Scientific Linux 5, both made from same RAW file
- Comparing a PATtuple 2.0 file to a PATtuple 1.0, both made from same RECO file

Note that in the last case, since Root can't load two versions of the same library (*e.g.,* PAT V1 and PAT V2), you will have to save one of them as a `GenObject` root file and then use that to compare to the other.

The first part of the documentation will assume that you are comparing two files of the same type. Note that before we can compare different variables on objects, we must first find the equivalent objects in both files. This matching is usually either done using matching (when available) or simply `index` (*e.g.,* assuming the objects are in the same order in both files).

Automatic Comparison

As you can see below, the main machinery needs to be told which variables to compare and how to access them. Since we are now planning on comparing two files with the same content, we will have these descriptions created for us automatically.

`runEdmFileComparison.py`

The general idea is to tell the master script *here are the two files to compare*. This command will create a lot of files, so I recommend you make an empty directory somewhere (inside of your release) that has soft links of the two files available. Here is a command that compares two ttbar files:

```
cplager@cmslpc12> runEdmFileComparison.py --prefix=ttbar_345_432 --verbose\
slc4_ia32_gcc345_ttbar_reco.root slc4_ia32_gcc432_ttbar_reco.root \
```

I tell the script to use the `prefix` of `ttbar_345_432` (so that I can run several comparisons of different files, each with a different prefix). `--verbose` tells the script to let me know at every stage what it's doing.

`runEdmFileComparison.py` Options

For a typical reco file, this command will lauch about 180 separate processes. The total time can take about 30 minutes or so. If you have access to a queuing system, you can tell this script to run the jobs in a local queue:

```
cplager@cmslpc12> runEdmFileComparison.py --prefix=ttbar_345_432  --verbose\
slc4_ia32_gcc345_ttbar_reco.root slc4_ia32_gcc432_ttbar_reco.root \
"--queueCommand=/uscms/home/cplager/bin/clpQueue.pl addjob" --verbose
```

where `/uscms/home/cplager/bin/clpQueue.pl addjob someCommand.bash` is how I tell my queuing system to queue the job `someCommand.bash`.

You can also:

- limit the branches that are compared by using `--regex=` XXX option.
- can print out the summary counts directly using `--summary`
- make diff trees using `--compRoot`

For example, the command below will:

- only run on CaloJets
- print out a summary
- make the diff trees

```
cplager@cmslpc12> runEdmFileComparison.py --prefix=ttbarv1_345_432 \
     slc4_ia32_gcc345_ttbar_reco.root slc4_ia32_gcc432_ttbar_reco.root \
     --regex=calojet --summary --compRoot --verbose
files slc4_ia32_gcc345_ttbar_reco.root slc4_ia32_gcc432_ttbar_reco.root
Getting edmDumpEventContent output
recoCaloJet.txt exists.  Skipping
comparing branch reco::CaloJet ak5CMS.CaloJets,,RECO
comparing branch reco::CaloJet ak7CMS.CaloJets,,RECO
comparing branch reco::CaloJet iterativeCone5CMS.CaloJets,,RECO
comparing branch reco::CaloJet kt4CMS.CaloJets,,RECO
comparing branch reco::CaloJet kt6CMS.CaloJets,,RECO
comparing branch reco::CaloJet sisCone5CMS.CaloJets,,RECO
comparing branch reco::CaloJet sisCone7CMS.CaloJets,,RECO
total:      7
success:    0
weird:      0
Problem types:
  mismatch       :    7
  -------------  : ----
  total          :    7 + 0 = 7
```

Note: When you use `--compRoot`, you will need to look at the resulting log file to get the name of the library you need to load. See below for more details.

`summarizeEdmComparisonLogfiles.py`

After the above job has finished, let's look at the results:

```
cplager@cmslpc12> summarizeEdmComparisonLogfiles.py ttbar_345_432_%_ logfiles --counts
total:      146
success:    43
weird:      0
Problem types:
  cppException   :    4
  labelDict      :    5
  mismatch       :   80
```

```
missingCfg      :    2
missingLabel    :    2
noEdmWrapper    :    9
uint32          :    1
-------------   : ----
total           :  103 + 43 = 146
```

A few comments about this command:

- If you give `runEdmFileComparison.py` a prefix, the first argument needs to be the prefix with `_%_` added on. In this example, I used a prefix of `ttbar_345_432`, so my first argument was `ttbar_345_432_%_`.
- If you do not give `runEdmFileComparison.py` a prefix, the first argument needs to be `%_`.
- I am assuming you are running this command in the same directory where you ran `runEdmFileComparison.py`. `runEdmFileComparison.py` makes a directory `logfiles` where it stores all logfiles. The second (optional) argument given this script is the directory location of log files.
- `--counts` gives you only the counts of what happened, but not the detailed summary.
  - `success:` means that the branches compared successfully (for the objects and precisions used).
  - `mismatch` means that there are disagreements among the variables compared.
  - `other` means that there is a new problem. Either one has made a mistake or has found a new problem. In this case, you should contact me.
  - `cppException` means there was a c++ exception generated when the tool is reading the data file
  - All other categories means that there is a known issue. In this case, we do not know whether this branch compares successfully or not.

```
cplager@cmslpc12> summarizeEdmComparisonLogfiles.py ttbar4_345_432_%_ logfiles >& summary.txt
```

(See the output file `summary.txt`)

Let's take a look at one of these:

```
ttbar4_345_432_recoCaloCluster_pfElectronTranslator_pf_RECO.log:
 {   'count_recoCaloCluster': 40,
  'eventsCompared': 10,
  'recoCaloCluster': {'_missing': {(0, 1): 2}, '_var': {'index': 1}}}
```

This tells us

- 10 events were compared.
- In these 10 events, 40 `recoCaloCluster` were successfully matched.
- `'_missing': {(0, 1): 2}` means that there were `2` events where the first file had `0` `recoCaloCluster` s and the second file had `1` `recoCaloCluster` s that were not matched to each other.
- `'_var': {'index': 2}` means that the variable `index` was not matched for 2 `recoCaloCluster` s out of the 40

```
ttbar2_345_432_recoSuperCluster_correctedHybridSuperClusters__RECO.log:
{'eventsCompared': 10, 'count_recoSuperCluster': 102}
```

shows that 102 `recoSuperCluster` s were compared in 10 events and that there were no problems.

diffTreeTool.py

This assumes you have already run the steps above and now want more details about the differences you are seeing.

```
cplager@cmslpc12> edmOneToOneComparison.py recoCaloJet.txt slc4_ia32_gcc345_ttbar_reco.root slc4_
   --compare \
   --tuple=reco \
   --label=reco^recoCaloJet^sisCone5CMS.CaloJets,,RECO \
   --compRoot=345_432_calojet.root
TClass::TClass:0: RuntimeWarning: no dictionary for class type_info is available
Loading GO Root Library
loading genobjectrootlibs/GenObject_ca1d1_C
(stuff deleted)
```

- The first line give the script the configuration file and the two root files.
- `--compare` tells it to compare the two files
- **`--label=reco^recoCaloJet^sisCone5CMS.CaloJets,,RECO`** tells it three things, seperated by **`^`** (carot)
  - For a **`reco`** file (all files right now are considered `reco`)
  - For the **`recoCaloJet`** object (same name as configuration file).
  - Using label **`sisCone5CMS.CaloJets,,RECO`**
- `--compRoot=345_432_calojet.root` tell is to write a diff tree `345_432_calojet.root`.
- You need to find the line `loading genobjectrootlibs/GenObject_ca1d1_C` as this tells you what library the next problem will load.

Once having created the diff tree, you can now print out information from it. We now run the `diffTreeTool.py`

```
cplager@cmslpc12> diffTreeTool.py \
   345_432_calojet.root \
   genobjectrootlibs/GenObject_ca1d1_C \
   eta phi  > diff.txt
```

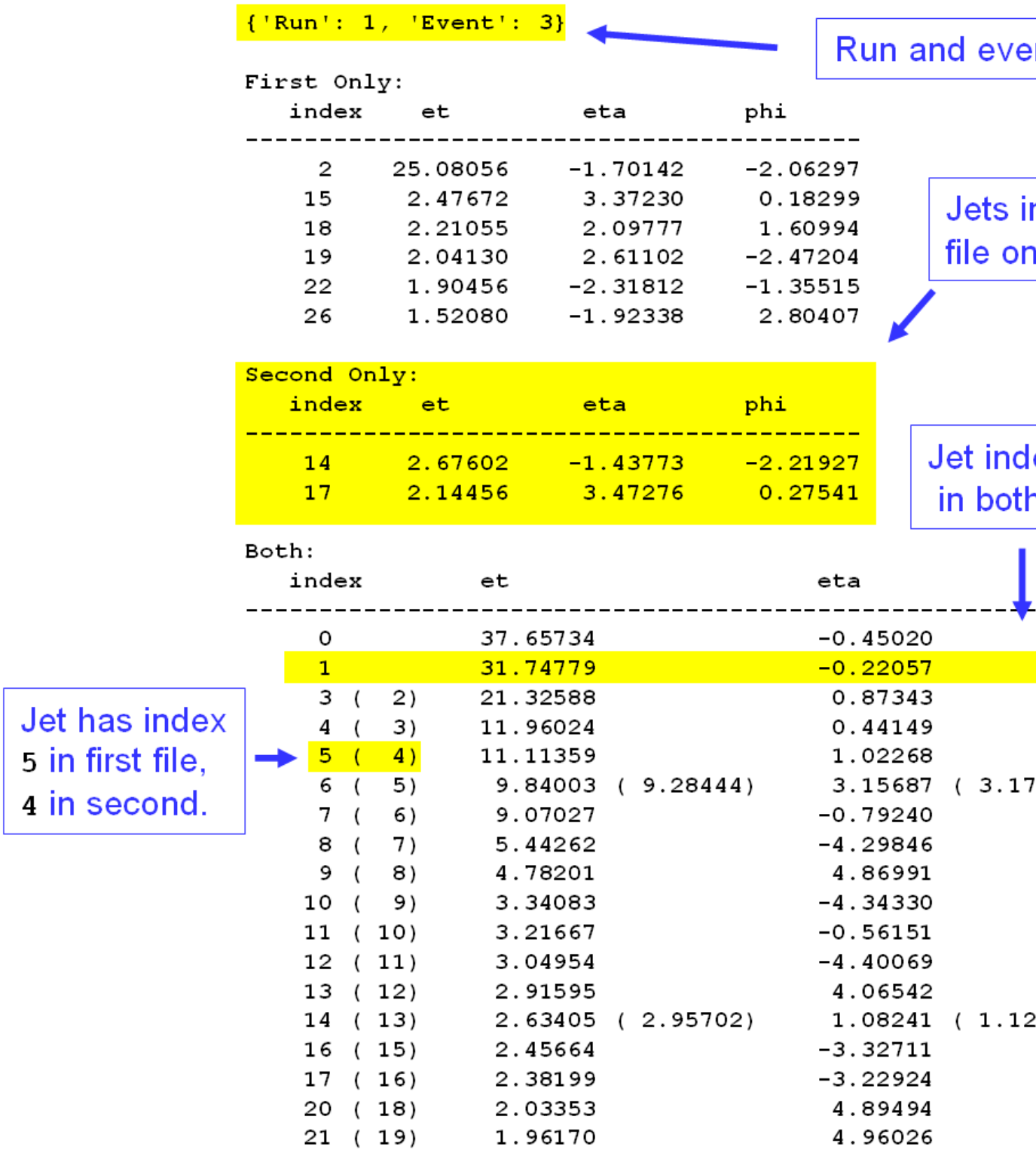
where we pass in

- `345_432_calojet.root` - diff tree name
- `genobjectrootlibs/GenObject_ca1d1_C` - shared object library created by `edmOneToOneComparison.py`
- `eta phi` - variables we want printed

(See the output file `diff.txt`)

Note that if we want to show the *deltas* instead of the value for the second file when there is a disagreement, you and use the `--delta` option.

Here is an detailed look at understanding the output.



Describing the variables that need to be compared

Everything needed to describe the ntuple files is in `config.txt`. This file is divided into two parts. First, for each object (e.g., a muon), we will define a `GenObject` which variables are going to be present and compared. Second, we will explain how to fill a `GenObject` from different files (e.g., a PATtuple).

`GenObject` Description

These scripts do not assume that the objects will be in the same order in both files. Therefore, for every `GenObject`, we need to first define "equivalence." This is the criteria, for example, of matching the muons in your first file for a given event for the muons in the second file. In the example below, we use `eta,0.1` and `phi,0.1` which mean that in order to match, both `eta` and `phi` must be smaller than the limits provided. In the case of multiple possibilities, the match will be which minimizes the difference in the variables listed.

We then list the variables that we will be matching. For each variable, we list an absolute precision (e.g., `prec=0.1` means 0.1 units) and a format for text printing (`form=%%7.2f`)

```
# GenObject 'muon' definition
[muon]
-equiv: eta,0.1 phi,0.1
px:   prec=0.1  form=%%7.2f
py:   prec=0.1  form=%%7.2f
pz:   prec=0.1  form=%%7.2f
eta:  prec=0.1  form=%%7.2f
phi:  prec=0.1  form=%%7.2f
```

Note: If your object is not a singleton (*e.g.,* structure holding run and event numbers), the `index` of the object will automatically be stored and available.

Describing How To Extract Information From Different ntuples.

Next we tell the code how to fill the GenObject from a given ntuple. In the case of a PATtuple, we tell the code to look for a tree name Events

```
#ntuple definition
[pat:Events]
```

Now we need to tell it how to get a `muon`. In `[]=s, we give it a =:` separated list of three items:

- `GenObject` name (*e.g.,* `muon`)
- ntuple name (*e.g.,* as defined just above `pat`)
- What we want to call these (*e.g.* `patMuons`)

We also want to define the complete alias (equivalent to `TTree::SetAlias()`).

```
# 'pat'-tuple 'muon' 'tofill' information
[muon:pat:patMuons alias=patMuons_selectedLayer1Muons__StarterKit.obj]
```

And finally tell it how to fill each of the muon's variables:

```
px:  px()
py:  py()
pz:  pz()
eta: eta()
phi: phi()
```

`jetpt_config.txt` is a complete, working example used to compare RECO to PATtuple jets.

Everything Below Here May Be Out Of Date

Running The Script

This script has the standard --help option:

```
cplager@cmslpc11> edmOneToOneComparison.py --help
usage: edmOneToOneComparison.py [options]
Visit https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePhysicsToolsEdmOneToOneComparison
for full documentation.

options:
  -h, --help            show this help message and exit

  Mode Conrols:
    --compare           Compare tuple1 to tuple2
    --saveAs=SAVEAS     Save tuple1 as GO Root file
    --printTuple        Print out all events in tuple1
    --interactive       Loads files and prepares "event" for interactive mode

  Tuple Controls:
    --tuple1=TUPLE1     Tuple type of 1st tuple
    --tuple2=TUPLE2     Tuple type of 2nd tuple
    --file1=FILE1       1st tuple file
    --file2=FILE2       2nd tuple file
    --numEvents1=NUMEVENTS1
                        number of events
    --alias=ALIAS       Change alias ('tuple:object:alias')
    --changeVariable=CHANGEVAR
                        Change variable filling ('tuple:objName:varName:def')

  Options:
    --config=CONFIG     Configuration file (default: 'config.txt')
    --printEvent        Prints loaded event to screen
    --printGlobal       Prints out global information (for development)
    --blur1=BLUR        Randomly changes values by 'BLUR'  from tuple1.  For
                        debugging only.
    --blurRate=BLURRATE
                        Rate at which objects will be changed. (0.02 default)
```

To take a PATtuple file and save it as a flat root file (so I can, for example, compare different versions of PATtuples)

```
cplager@cmslpc11> edmOneToOneComparison.py --config=examples/jetpt_config.txt \
--tuple1=pat --file1=PatAnalyzerSkeletonSkim.root --saveAs=go2.root
```

To compare two ntuples:

```
cplager@cmslpc11> edmOneToOneComparison.py --config=examples/jetpt_config.txt  --compare \
--tuple1=pat --file1=PatAnalyzerSkeletonSkim.root \
--tuple2=GenObject --file2=go2.root
```

To dump out the contents of one file

```
cplager@cmslpc11> edmOneToOneComparison.py  --config=examples/jetpt_config.txt --printTuple \
--tuple1=pat --file1=examples/ttbar_pat_v4_numEvent200.root --numEvents1=1
runevent: event:751  run:1
jet:
   eta:  0.04  index:  0  phi: -2.80  pt: 101.90
   eta: -1.07  index:  1  phi:  0.27  pt:  79.95
   eta: -1.03  index:  2  phi:  2.31  pt:  65.46
   eta: -0.15  index:  3  phi: -0.02  pt:  58.63
```

```
   eta:  3.10  index:  4  phi: -2.68  pt:  55.44
   eta: -1.43  index:  5  phi: -1.39  pt:  41.54
   eta:  1.70  index:  6  phi:  0.81  pt:  39.78
   eta: -0.94  index:  7  phi:  1.29  pt:  20.64
   eta:  2.87  index:  8  phi:  2.33  pt:  13.51
   eta:  0.60  index:  9  phi: -0.76  pt:  12.53
   eta:  3.98  index: 10  phi:  1.74  pt:   6.96
   eta: -3.61  index: 11  phi: -0.98  pt:   5.74
   eta:  4.77  index: 12  phi: -2.82  pt:   5.73
   eta:  1.81  index: 13  phi: -1.80  pt:   5.25
   eta:  3.41  index: 14  phi:  0.85  pt:   4.96
   eta: -1.18  index: 15  phi: -0.62  pt:   4.94
   eta:  2.57  index: 16  phi: -0.06  pt:   3.11
```

Interpreting the Output

To verify that my script picks up on problems, I created a way to introduce problems where none existed. `--blur1=10 --blurRate=0.002` will randomly change 0.2% of the variables by a quanity of `10` units.

```
cplager@cmslpc11> edmOneToOneComparison.py   --config=examples/jetpt_config.txt  --compare \
--tuple1=pat --file1=examples/ttbar_pat_v4_numEvent200.root \
--tuple2=reco --file2=examples/ttbar_jetcorReco_v4_numEvent200.root \
--blur1=10 --blurRate=0.002
Comparing Two Trees
  run 1 event 882: changing 'index' of 'jet:7'
  run 1 event 882: changing 'index' of 'jet:22'
  run 1 event 893: changing 'pt' of 'jet:9'
  run 1 event 866: changing 'eta' of 'jet:2'
  run 1 event 3546: changing 'eta' of 'jet:8'
  run 1 event 3535: changing 'pt' of 'jet:14'
  run 1 event 3525: changing 'eta' of 'jet:11'
  run 1 event 3277: changing 'pt' of 'jet:9'
  run 1 event 3286: changing 'phi' of 'jet:18'
  run 1 event 3501: changing 'pt' of 'jet:3'
  run 1 event 3262: changing 'pt' of 'jet:10'
  run 1 event 3540: changing 'phi' of 'jet:9'
  run 1 event 3540: changing 'eta' of 'jet:10'
  run 1 event 790: changing 'pt' of 'jet:2'
  run 1 event 3529: changing 'pt' of 'jet:1'
  run 1 event 3516: changing 'phi' of 'jet:4'
  run 1 event 779: changing 'phi' of 'jet:11'
  run 1 event 3290: changing 'pt' of 'jet:18'
  run 1 event 3509: changing 'eta' of 'jet:18'
  run 1 event 780: changing 'pt' of 'jet:5'
  run 1 event 3514: changing 'pt' of 'jet:6'
  run 1 event 899: changing 'eta' of 'jet:14'
  run 1 event 855: changing 'phi' of 'jet:24'
  run 1 event 3504: changing 'index' of 'jet:18'
  run 1 event 3283: changing 'pt' of 'jet:5'
  run 1 event 3283: changing 'phi' of 'jet:10'
  run 1 event 762: changing 'index' of 'jet:15'
  run 1 event 3507: changing 'index' of 'jet:1'
  run 1 event 3507: changing 'pt' of 'jet:9'
  run 1 event 3289: changing 'eta' of 'jet:8'
  run 1 event 873: changing 'eta' of 'jet:9'
problems
{'jet': {'_missing': {(1, 1): 12, (2, 2): 1}, '_var': {'index': 5, 'pt': 12}}}
```

- `'_runevent': {'firstOnly': 3, 'secondOnly': 5}` would mean that there were three events in the first file that are not present in the second, and that there are five events in the second file not present in the first. (This is not there in this example)
- `'jet'`: now tells us that we will see what problems were found with jet comparisons.

- ♦ `'_missing': {(a, b): c}` means that there are `c` events where we have `a` jets in the first file that are not matched to jets in the second file and `b` jets in the second file that are not matched to the first file. In the above example='_missing': {(1, 1): 12, (2, 2): 1}= says that there are 12 events where we have a single jet in each file that is not matched and 1 event where we have two jets in each file that are not matched.

- ♦ `'_var'`: tells us which variable matches fail. For example: `'_var': {'index': 5, 'pt': 12}` tells us that the variable `index` is not matched five times and `pt` is not matched 12 times.

-- CharlesPlager - 12-Oct-2009

-- PetarMaksimovic - 27 Mar 2009

This topic: CMSPublic > SWGuidePhysicsToolsEdmOneToOneComparison
Topic revision: r11 - 2012-01-20 - GiulioEulisse