ecFlow@ECMWF

Setting the environment is done calling

```
module load ecflow/5new
```

It is possible to setup a specific version with

```
module unload ecflow; module load ecflow/5.5.0
```

Server can be started with

```
ecflow_start.sh
```

Client command can be called to get the self-contained documentation

```
ecflow_client --help
```

and the graphical interface is started with

```
# line below shall add localhost as part of
# the ecflowview->Servers list
grep localhost $HOME/.ecflowrc/servers ||echo\
  "localhost $(uname -n) $((1500 + $(id -g)))"\
  >> $HOME/ecflowrc/servers
# start the GUI:
ecflow_ui
```

Server administrator directory is **$HOME/ecflow_server/** which will contain the server log file, the check point file (binary snapshot of the server content). It is defined as variable **ECF_HOME** on the top node.

> Once the server and its GUI are started, click on Servers->localhost to connect the first time.

Next step is to load a suite into the server. The following python script can be used for suite definition, to expand the suite into a file, and to load it into the server, as shown in the ecflowview snapshot below

suite definition and loading

```
#!/usr/bin/env python
# -*- coding= UTF-8 -*-
""" course 2014
sample suite to spawn jobs on main available cpu facilities
at  ECMWF
"""

from __future__ import with_statement
import sys, os, pwd, getopt
sys.path.append('/home/ma/emos/def/o/def')
import inc_emos as ic
from inc_emos import (Family, Task, Edit, Meter, Event,
    Label, Limit, If, Complete, Time, Trigger, Client)
from ecf import get_username, Clock, Defstatus, Inlimit, Repeat, get_uid, Defs

# import ecf; ecf.USE_TRIGGER = 0 # DEBUG MODE: not to load triggers

def create_task():
    """ create example tasks wrappers and include files"""
    ### task
    content = """#!/bin/ksh
%manual
manual - this task is automatically created by cray.py
```

```
%end
%include <qsub.h>
if [[ $ARCH = hp* ]]; then export PATH=/usr/local/bin:$PATH; fi
if [[ $HOST = lxop* ]]; then export PATH=/usr/local/apps/ecflow/current:/usr/local/apps/sms/bin:$PATH; fi
%include <trap.h>
SLEEP=%SLEEP:0%
echo OK

case %ECF_PORT:0% in
0) base=900000; LOGPORT=%LOGPORT:0%;;
*) base=1000; LOGPORT=%LOGPORT:0%;;
esac
base=1000

case $ARCH in
cray)xlabel info $(printenv | grep -E '(SUBMIT_|EC_)') ;;
*) xlabel info OK
esac

printenv | sort
xevent 1
step=0
while (( $step <= 12 )); do
  xmeter step $step
  ((step = step + 1))
  sleep $SLEEP
done
%include <endt.h>
"""
    create_wrapper("test.sms", content)

    ### PYTHON TASK
    content = """$include <python_header.h>
# header files are located in the same directory as wrapper (quotes)
for step in range(0,101):
    print step
    xmeter("step", step)
else:
    print 'the loop is over'
xevent("1")
xlabel("info", "news from pure python world")

$include <python_endt.h>
"""
    create_wrapper("python.sms", content)

    content = """#!/usr/bin/env python
import os
import sys
import signal

ECF_PORT=$ECF_PORT:0$
XECF="/usr/local/apps/ecflow/current/bin/ecflow_client ";
# --port=$ECF_PORT:0$ --host=$ECF_NODE:0$ ";
def SigHandler(signum, frame):
   print "caught signal " + signum
   xabort()
   sys.exit(0);
   return

print ECF_PORT
# set -eux ?
# trap 0 ?
# time stamp per executed line

import atexit
early_exit = True
@atexit.register
def goodbye():
  if early_exit:
    print "too early"
```

```
    xabort()
  else:
    xcomplete()

# TIME_STAMP
# http://shop.oreilly.com/product/9780596007973.do # recipie p436
import syslog, time

class FunctionFileLikeWrapper():
    def __init__(self, func): self.func = func
    def write(self, msg): self.func(msg)
    def flush(self): pass

class TimeStamper(object):
    msg_format = "[%y%m%d %H:%M:%S]", time.gmtime, "%s: %s"
    msg_format = "+ %H:%M:%S", time.gmtime, "%s %s"

    def __call__(self, msg):
        tfmt, tfun, gfmt = self.msg_format
        return "%s %s\\\n" % (time.strftime(tfmt, tfun()), msg)

class TeeFileLikeWrapper():
    def __init__(self, *files): self.files = files
    def write(self, msg):
        for f in self.files: f.write(timestamp(msg.strip()))

class FlushingWrapper:
    def __init__(self, *files): self.files = files
    def write(self, msg):
        for f in self.files:
            f.write(timestamp(msg))
            # f.write(timestamp(msg.strip()))
            f.flush()

def logto(*files):
    # sys.stdout = TeeFileLikeWrapper(*files)
    sys.stdout = FlushingWrapper(*files)

syslogger = syslog.syslog
syslogfile = FunctionFileLikeWrapper(syslogger)
timestamp = TimeStamper()
logto(sys.stdout, syslogfile, open("log.tmp", "w"))
# end time stamp

if ECF_PORT > 0:
  os.environ['ECF_PORT'] = "$ECF_PORT:0$"
  os.environ['ECF_NAME'] = "$ECF_NAME:0$"
  os.environ['ECF_NODE'] = "$ECF_NODE:0$"
  os.environ['ECF_PASS'] = "$ECF_PASS:0$"

  def xinit():
    os.system(XECF + " --init " + str(os.getpid()))
    print "init"
  def xabort():
    os.system(XECF + " --abort")
  def xcomplete():
    os.system(XECF + " --complete")
  def xmeter(name, step):
    os.system(XECF + " --meter " + name + " " + str(step))
  def xevent(name):
    os.system(XECF + " --event " + name)
  def xlabel(name, msg):
    os.system(XECF + " --label " + name + " '%s'" % msg)
else:
  os.environ['SMS_PROG'] = "$SMS_PROG:0$"
  os.environ['SMSNAME'] = "$SMSNAME:0$"
  os.environ['SMSNODE'] = "$SMSNODE:0$"
  os.environ['SMSPASS'] = "$SMSPASS:0$"

  def xinit():
    os.system('smsinit ' + str(os.getpid()))
```

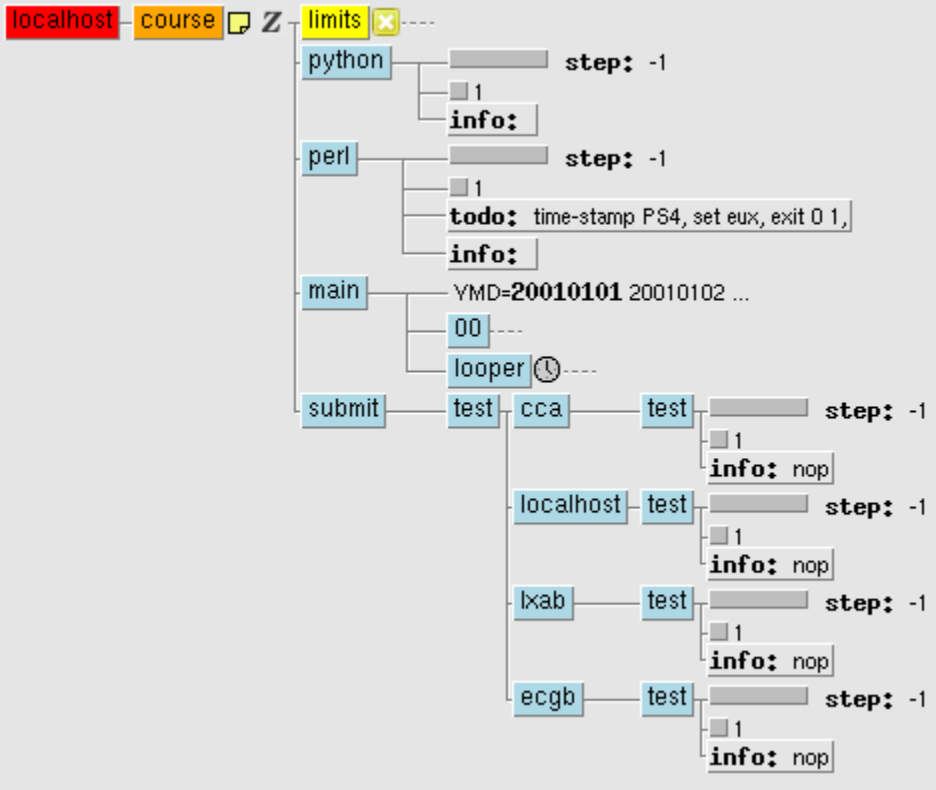
```
  def xabort():
    os.system('smsabort')
  def xcomplete():
    os.system('smscomplete')
  def xmeter(name, step):
    os.system("smsmeter " + name + " " + str(step))
  def xevent(name):
    os.system("smsevent " + name)
  def xlabel(name, msg):
    os.system("smslabel " + name + " '%s'" % msg)

signal.signal (signal.SIGHUP,  SigHandler)
signal.signal (signal.SIGINT,  SigHandler)
signal.signal (signal.SIGQUIT, SigHandler)
signal.signal (signal.SIGILL,  SigHandler)
signal.signal (signal.SIGTRAP, SigHandler)
signal.signal (signal.SIGIOT,  SigHandler)
signal.signal (signal.SIGBUS,  SigHandler)
signal.signal (signal.SIGFPE,  SigHandler)

"""
    create_wrapper("python_header.h", content)

    content = """# os.system('smscomplete')
early_exit = False
# xcomplete() # managed with atexit
# os.system('/usr/local/apps/sms/bin/ecflow/bin/ecf_client --complete')
"""
    create_wrapper("python_endt.h", content)

    ### PERL TASK
    create_wrapper("perl.sms", """#!/usr/bin/perl -w
^include <perl_header.h>
# header files are located in the same directory as wrapper (quotes)
print "Pure perl SMS task";
for ( my $step=1; $step <= 100 ; $step++ ) {
   print "this is the number $step\\\n";
   xmeter("step", $step);
}
xevent("1");

xlabel("info", "news from pure perl world");
^include <perl_endt.h>
""")
    # create_wrapper("perl.sms", content)

    source = """use strict;

my $xmeter = "smsmeter"; my $arg_m = "";
my $xlabel = "smslabel"; my $arg_l = "";
my $xevent = "smsevent"; my $arg_e = "";
my $xcomplete = "smscomplete"; my $arg_c = "";
my $xabort = "smsabort";

if (^ECF_PORT:0^ != 0) {
$ENV{'ECF_PORT'}  = "^ECF_PORT:0^" ;  # ecFlow port number
$ENV{'ECF_NODE'}  = "^ECF_NODE:0^"  ; # ecFlow host
$ENV{'ECF_NAME'}  = "^ECF_NAME:0^"  ; # task path into the suite
$ENV{'ECF_PASS'}  = "^ECF_PASS:0^"  ; # password for the job
$ENV{'ECF_TRYNO'} = "^ECF_TRYNO:0^" ; # job occurrence number
my $client = "/usr/local/apps/ecflow/current/bin/ecflow_client";
$xmeter = $client; $arg_m = "--meter";
$xlabel = $client; $arg_l = "--label";
$xevent = $client; $arg_e = "--event";
$xcomplete = $client; $arg_c = "--complete";
$xabort = $client;

system($client, "--init", "$$");
} else {
  $ENV{'SMS_PROG'} = "^SMS_PROG:0^" ; # SMS Program Number
  $ENV{'SMSNODE'}  = "^SMSNODE:0^"  ; # SMS host
```

```
  $ENV{'SMSNAME'}  = "^SMSNAME:0^"  ; # task path into the suite
  $ENV{'SMSPASS'}  = "^SMSPASS:0^"  ; # password for the job occurrence
  $ENV{'SMSTRYNO'} = "^SMSTRYNO:0^" ; # job occurrence number
}

sub xmeter($$){ my ($name, $step) = @_;
  system($xmeter, $arg_m, $name, $step); }
sub xevent($){ my ($name) = @_;
  system($xevent, $arg_e, $name); }
sub xlabel($$){ my ($name, $msg) = @_;
  system($xlabel, $arg_l, $name, $msg); }
sub xabort(){ system($xabort); }
sub xcomplete(){ system($xcomplete, $arg_c, "$$"); }

print "start";
eval '
"""
    create_wrapper("perl_header.h", source)

    source = """';
if ($@){
    print "caught signal: $@\n";
    xabort();
    exit;
  }
print "the job is now complete\n";
xcomplete();
exit;"""
    create_wrapper("perl_endt.h", source)

def create_wrapper(name, content):
    """ wrapper creation """
    print "#MSG: creating file %s/" % wdir + name
    wrapper = open(wdir + "/%s" % name, 'w')
    print >> wrapper, content
    wrapper.close()
##############################################


def dummy():
    """ create test task"""
    return Task("test")

def limits():
    """create limits famly"""
    return (Family("limits").add(
            Defstatus("complete"),
            Limit("lim", 10),
            Limit("test", 10),
            Limit("hpc", 10),
            Limit("cca", 10),
            Limit("c2a", 10),
            Inlimit(ic.psel() + "/limits:lim")))

HCRAY = "cct"
SUBMIT = ic.SUBM + " %USER% %SCHOST% %ECF_JOB% %ECF_JOBOUT% submit"
GSUB = SUBMIT.replace("%SCHOST%", "%SCHOST% %ECF_RID%")
KILL = GSUB.replace(" submit", " kill")
STATUS = GSUB.replace(" submit", " status")
CHECK = STATUS

def unit(name, schost, queue, rdir, account="", leaf=True):
    """ course suite tree leave family"""
    return Family(name).add(
        If(account != "",
           Edit(ACCOUNT= account)),
        Edit(QUEUE=    queue,
                  SCHOST=   schost,
                  ECF_OUT=  rdir,
                  LOGDIR=   rdir,),
        If (leaf, Task("test").add(
```

```
            Event(1),
            Meter("step", -1, 120, 100),
            Label("info", "nop"))))


def call_python():
    """ pure python task example"""
    return Task("python").add(
        Edit(ECF_MICRO= "$",
                  ECF_INCLUDE= wdir,
                  SCHOST= "localhost",
                  ECF_JOB_CMD= "$ECF_JOB$ > $ECF_JOBOUT$ 2>&1 &"),
        Event("1"),
        Label("info", "micro is $"),
        Meter("step", -1, 100),
        # Defstatus("complete"),
        )

def call_perl():
    """ pure perl task example"""
    return Task("perl").add(
        Label("todo", "time-stamp PS4, set eux, exit 0 1"),
        Edit(ECF_MICRO= "^",
                  ECF_INCLUDE= wdir,
                  SCHOST= "localhost",
                  ECF_JOB_CMD= "^ECF_JOB^ > ^ECF_JOBOUT^ 2>&1 &"),
        Event("1"),
        Label("info", "micro is ^"),
        Meter("step", -1, 100),
        )

##########################################################################

class Course(ic.SeedOD):
    """ example class for a suite suite definition"""
    def __init__(self):
        super(Course, self).__init__()

    def setup(self, node):
        account = "UNSET"
        global user, host, SUITE
        self.defs.add_extern("/o/main:YMD")

        node.add(
            Clock("real"),
            Defstatus("suspended"),
            Event("1"),
            Meter("step", -1, 100),
            Label("info", "click edit to update manually"),
            limits(),

            Edit(ECF_JOB_CMD= SUBMIT,
                      ECF_KILL_CMD= KILL,
                      ECF_STATUS_CMD= STATUS,
                      ECF_CHECK_CMD= CHECK,
                      ECF_EXTN= ".sms",
                      ACCOUNT=  "UNSET",
                      SCHOST= "localhost",
                      ECF_INCLUDE= idir,
                      ECF_HOME= jdir,
                      ECF_FILES= wdir,
                      QUEUE= "ns", USER= user, LOGDIR= "/tmp",
                      TOPATH= udir + "/logs",
                      SLEEP=    1,     # x120
                      ECF_TRIES= 1,),

            call_python(),

            call_perl(),
            )
```

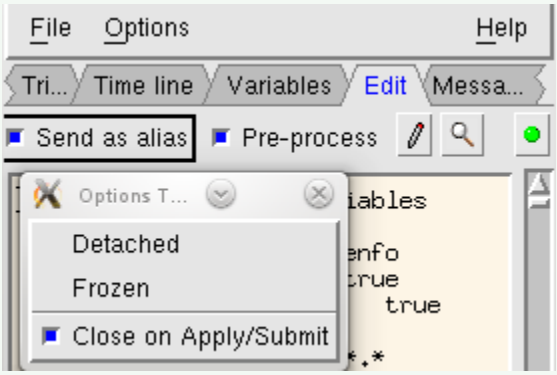
```
        msg = "have you? setup ssh login, created remote directory"
        msg += ", considered the need for a logserver"

        node.family("main").add(
            Repeat(name="YMD", kind="date",
                   start=20010101, end=20991231, step=1),
            Family("00").add(
                Task("REPLACE_ME").add(Defstatus("complete"))
                ),
            Family("loop").add(
                Label("info", "prevent direct repeat increment"),
                Trigger("./00==complete"),
                Complete("/%s:1" % node.name()),
                Time("12:00")))

        node = node.family("submit").add(
            Edit(USER=user,
                 ACCOUNT=account),
            self.submits(), )

    def submits(self):
        # global HOST, user
        ct1logs = "/sc1/sb/%s/logs" % user
        out = [ unit("cca", "cca", "ns",    ct1logs)  ]

        out += (unit("localhost", "localhost", "ns", "/tmp/%s/logs" % user),
                unit("lxab", "lxab", "serial", udir + "/logs"),
                unit("ecgb", "ecgb", "ns", udir + "/logs"),
                )
        return out

def usage():
    print sys.argv[0] + ''' -h -u [user] -n [node] -s [suite-name] \
 -e: ecflow'''

if __name__ == "__main__":
    global user, udir, ecflow, HOST
    SUITES = {"course": Course,
              }
    user = get_username()
    try:    HOST = sys.argv[-1]
    except: HOST = "localhost"
    ECFLOW  = True
    SUITE   = None
    user = get_username()
    opts, args = getopt.getopt(
          sys.argv[1:], "hp:u:es:n:p:",
          ["help", "port", "user", "ecflow", "suite", "node", "path"])

    output = None
    verbose = False
    for o, a in opts:
        if   o in ("-n", "--node"):
            HOST = a
        elif o in ("-h", "--help"):
            usage()
            sys.exit()
        elif o in ("-s", "--suite"):
            SUITE =  a
        elif o in ("-p", "--path"):
            path =  a
        elif o in ("-e", "--ecflow"):
            ecflow = True
        elif o in ("-u", "--user"):
            user = a
        elif o in ("-p", "--port"):
            port = a
        else: print "#ERR: what?", o, a; assert False, "unhandled option"

    ########### SETTINGS ################ WRAPPERS
    global wdir, rdir, fdef, jdir
```

```
    udir = pwd.getpwnam(user).pw_dir
    wdir = udir + "/ecflow_server/course"
    jdir = "/tmp/%s/ecflow" % user
    ddef = udir + "/ecflow_server"
    fdef = ddef + "/course.def"
    idir = "$HOME/ecflow_server/course/include"

    try:
        os.makedirs(wdir)
    except:
        pass
    try:
        os.makedirs(jdir)
    except:
        pass
    create_task()

    ########### SETTINGS ################ LOAD
    if 1:
        client = Client(sys.argv[3], 1000 + int(get_uid()))
        top = Course().suite()
        if 1:
          out = file("%s.exp" % top.name(), 'w'); print >>out, top
        defs = Defs()
        defs.add_suite(top)
        client.replace(sys.argv[2], defs)
        sys.exit(0)
```



Clicking on each task, we can check the presence of task wrapper **script** (**ECF_FILES** defined properly), then **Edit** it, **preprocess** it (**ECF_INCLUDE** defined properly, no micro character on its own), and **submit it**, as a task or as an **alias.**

Consider Options->CloseOnApply/Submit when multiple aliases must be sent from the same task in a short time.



If the task does not reach the active status, then check that **ECF_OUT** directory exists on the remote host, check that expected rsh or ssh connection does not request password , or query the queuing system while the directive may not be valid (user account, queue), yet.

When the *submit* family is working for the expected remote host(s), time to fill the *main* family with relevant tasks. Enjoy!

(Expanded) Definition file:

It consists of the following keywords for nodes and their attributes definition:

- **suite family task** endsuite endfamily endtask
- *autocancel automigrate autorestore,clock complete cron date day defstatus edit event extern inlimit label late limit meter repeat time today trigger*

- on a line, text beyond # is a comment

Comparing with SMS *text owner* are left behind.

autocancel

```
for a node to be deleted automatically
autocancel +01:00  # cancel one hour after complete
autocancel 10      # cancel 10 days after complete
autocancel 0       # cancel immediately after being complete
```

clock

```
clock real # hybrid may be used in test mode
```

complete

```
for a node, to be recursively forced complete from a condition
complete t1:1 or t1==complete
```

cron

```
to run a task regularly, task is requeued as soon as complete is received
ie no trigger on the parent task complete shall be used
task can only become complete, thanks to inherited defstatus or complete attribute
cron 23:00                    # at next 23:00
cron 10:00 20:00 01:00 # every hour from 10am to 8pm
```

date

```
date 25.12.2012
date 01.*.*
```

day

```
day monday # sunday,monday,tuesday,wednesday,thursday,friday,saturday
```

defstatus

```
defstatus complete #unknown,suspended,queued,submitted,active,aborted
```

edit

```
to attach a variable definition to a node
edit variable value
# variables to be find/and/replaced in a task wrapper
edit COMMAND  "echo OK" # %COMMAND:sleep 1%
edit TRIGGER  "t1==complete" # ecflow_client --wait="%TRIGGER:1==1%"
```

event

```
event 1 # may fit a call in task.ecf to 'ecflow_client --event=1'
event ready # ecflow_client --event=ready
```

extern

```
extern /path/to/a/external/node # to allow path's use in trigger/complete
```

inlimit

```
register the node and its kids to a limit
inlimit /limits:hpc
inlimit /suite/limits:hpc
inlimit /suite/limits:hpc 10
```

label

```
label name "default message" # task.ecf: ecflow_client name "label update"
```

late

```
late -s +00:15 -a 20:00 -c +02:00
```

limit

```
limit hpc 500
```

meter

```
meter name -1 100 90 # 90 is threshold (optional) # task.ecf: ecflow_client --meter=name 30
```

repeat

```
repeat is incremented when all nodes below are complete
an aborted task DOES prevent repeat to increment
an Operator/Analsyst/dedicated task can help carry on
repeat day        step [ENDDATE]     # only for suites
repeat integer    VARIABLE start end [step]
repeat enumerated VARIABLE first [second [third ...]]
repeat string     VARIABLE str1 [str2 ...]
repeat date       VARIABLE yyyymmdd yyyymmdd [delta]
```

time

```
task become complete ONLY when time range is over
better not to use such task in a trigger expression
time 23:00                # at next 23:00
time 10:00 20:00 01:00    # every hour from 10am to 8pm
time +00:01               # one minute after the begin suite
time +00:10 01:00 00:05   # 10-60 min after begin every 5 min
```

today

```
with such attribute, task will start straight when loaded/replaced after given time
while time attribute would make it wait the next day
today 3:00                        # today at 3:00
today 10:00 20:00 01:00    # every hour from 10am to 8pm
```

trigger

```
for a task to wait the right condition (step/meter/status/variable(int)) to start
```

Py-Def

As soon as a definition file is beyond few hundred lines, or even before, when obvious repeated patterns are used, a language like **Python** shall be used. At the Centre, a python module is used for both research and operation to reduce verbosity in suite definition (*/home/ma/emos/def/o/def/ecf.py)*

```
#!/usr/bin/env python
import sys, pwd; sys.path.append('/home/ma/emos/def/o/def')
# ipython # import ecf; help(ecf.<tab>)
from ecf import *
defs = Defs()
def fill(): # functions can generate tasks/families
        return [Task("t%02d" % i).add(Event(1),
                           Meter("step", -1, 100),
                               Label("info", ""), )
            for i in xrange(1, 10+1)]  # LIST COMPREHENSION
home = os.getenv("HOME") + "/ecflow_server"
top = Suite("test").add(
   Edit(ECF_HOME= home, # where job, local .out go
            ECF_FILES= home + "/files", ### where .ecf are found
        ECF_INCLUDE= home + "/include", ### where .h are found
        ECF_OUT= home + "out", # output remote/local location, create missing directories...
        ),
   Family("fam").add(
    Task("t00").add(
      Trigger("t01==complete"),
      Complete("t02:1 or t02==complete"),),
    fill(), ))
if __name__ == '__main__':
  uid = pwd.getpwnam(pwd.getpwuid( os.getuid() )[ 0 ]).pw_uid
  host = "localhost"
  client = Client(host,1500+ui)
  path = "/test"
  defs.add_suite(top)
  client.replace(path, defs)
```

Task-Wrapper (ecf-file) and header-files

ecf-file

```
#!/usr/bin/env ksh
%manual
  DESCRIPTION: ...
    input(s): ...
    output(s): ...
  OPERATORS: ...
  ANALYST: ...
%end
%comment
  # ...
%end
%include <qsub.h>
%include <head.h>
# main section
 %COMMAND:printenv% # a variable may contain a command

ecflow_client --wait="%TRIGGER:1==1%"
# or a embedded blocking/trigger condition

ecflow_client --event=1
ecflow_client --meter=step 30
ecflow_client --label="updating"
%nopp
  # no preprocessing, here
%end
# a directive to include a file without preprocessing
cat > test.pl <<\EOF
%includenopp <test.pl>
EOF
%ecfmicro @
# from now _at_ is the micro character for directives and variables
# ...
# and revert to percent:
@ecfmicro %
%include <tail.h>
```

head.h

```
#!/bin/ksh
# Defines the variables that are needed for any communication with ECF
export ECF_PORT=%ECF_PORT%    # The server port number
export ECF_HOST=%ECF_HOST%    # The name of ecf host that issued this task
export ECF_NAME=%ECF_NAME%    # The name of this current task
export ECF_PASS=%ECF_PASS%    # A unique password, ...
export ECF_RID=$$             # record the process id. Used for zombie detection
# set as FREE on the server with menu
# ecFlowUI=>Special=>FreePassword, to accept communication
# with a "zombie" with invalid pass
set -eux; export PATH=/usr/local/apps/ecflow/%ECF_VERSION%/bin:$PATH
ERROR() {
   set +e; wait; ecflow_client --abort=trap; trap 0; exit 0
}
trap '{ ERROR ; }' 0 1 2 3 4 5 6 7 8 10 12 13 15
ecflow_client --init=$$
```

tail.h

```
wait; ecflow_client --complete; trap 0; exit 0
```

suite skeleton targeting several destinations at the Centre (HPC, ecgate, linux cluster)

```
python /home/ma/emos/def/o/def/cray.py
```

Custom GUI

Tkinter can be used to setup an simple GUI client. Suite traverser is described in the cookbook.

As an example, overview.py is provided, while SMS/CDP has a similar command available.

overview

```
 #!/usr/bin/env python
""" tkinter use example with ecflow
  from a NCEP request, while overview was part of CDP commands
"""

import Tkinter as tki
import ecflow  as ec
from threading import Thread
import Queue, sys, time, os, pwd
from scrolledlist import ScrolledList
# thanks to NMT
# from=http://infohost.nmt.edu/tcc/help/lang/python/examples/scrolledlist/
# firefox $from ; wget $from/scrolledlist.py

PROGRAM_NAME =  "ecflowview-overview"
BUTTON_FONT  =  ('times', 12)
MONO_FONT    =  ('lucidatypewriter', 14, 'bold')
DEBUG = 0

COLORS = { "aborted": "red",
           "active":  "green",
           "submitted": "cyan",
           "complete": "yellow",
           "suspended": "orange",
           "queued": "blue",
           "unknwon": "grey" }

class Label(object):
    """ a class to encapsulate what was a global variable"""
    inst = None

    def __init__(self, item): Label.inst = item

    @classmethod
    def update(cls):
        if Label.inst is None: return
        Label.inst.set(time.strftime(
                "%a, %d %b %Y %H:%M:%S"))

class MenuBar(tki.Frame):
    def __init__(self, parent):
        tki.Frame.__init__(self, parent)
        self.__helpButton = self.__createHelp()
        self.__helpButton.grid(row=0, column=3)
        self.__updateButton = tki.Button(
            self, text='Update',
            font= BUTTON_FONT,
            command= parent.update)
        self.__updateButton.grid(row=0, column=2)

    def __createHelp(self):
        mb = tki.Menubutton(self, font=BUTTON_FONT,
                        relief= tki.RAISED,
                        text= 'Help')
        menu = tki.Menu(mb)
        mb['menu'] = menu
```

```
        url = "https://confluence.ecmwf.int/display/ECFLOW/Documentation"
        def url1(): self.__url(url= url)
        def url2(): self.__url(url="http://effbot.org/tkinterbook/")
        menu.add_command(command= url1, label="confluence tutorial?")
        menu.add_command(command= url2, label="tkinter?")
        return mb

    def __url(self, url=None):
        if url is None: return
        os.system("firefox " + url)

class TaskList(tki.Frame):
    NAME_WIDTH = 40
    NAME_LINES = 80

    def __init__(self, parent, kind):
        tki.Frame.__init__(self, parent)
        self.__kind = kind
        self.__callback = None
        self.__label = tki.Label(self, font=BUTTON_FONT,
                                 background= COLORS[kind],
                                 text= kind)
        self.__label.grid(row=0, column=0, sticky= tki.W)
        self.__scrolledList = ScrolledList(
            self,
            width= self.NAME_WIDTH,
            height= self.NAME_LINES,
            callback= self.__callback)
        self.__scrolledList.grid(row=1, column=0)

    def insert(self, path): self.__scrolledList.append(path)

    def clear(self):        self.__scrolledList.clear()

running = [True]
class PaceKeeper():
    PACE = 60
    def __init__(self, item, queue):
        thr = Thread(target=self.process,
               args=(queue, running))
        self._item = item
        thr.start()

    def process(self, queue, running):
        while running:
            queue.put(self._item.update)
            time.sleep(self.PACE)

    def run(self): self.update()

    def update(self, verbose=False):
        while True:
            print time.clock()
            self._item.update()
            time.sleep(self.PACE)

class Client(object):
    """ a class to focus on client-ecFlow-server comm"""

    def __init__(self, one="local-localhost@31415"):
        try:    nick, hhh = one.split("-")
        except: hhh = one; nick = None
        try:    host, port = hhh.split("@")
        except: host = "localhost"; port = 31415

        if nick is None: self.nick = "%s@%d" % (host, port)
        print "# client creation", nick, host, port
        self.nick = nick
        self.client = ec.Client(host, port)

    def process(self, win):
```

```
        Label.update()
        self.client.sync_local()
        defs = self.client.get_defs()
        if defs is None: print("# %s-%: empty content" % (
                self.host, self.port))
        Label.update()
        for suite in defs.suites:
            self.process_nc(suite, win)

    def process_nc(self, node, win):
        for item in node.nodes:
            if isinstance(item, ec.Task):
                self.process_node(item, win)
            else: self.process_nc(item, win)

    def process_node(self, node, wins):
        for kind, win in wins.items():
            status = "%s" % node.get_state()
            if status != kind: continue
            win.insert("%s:%s" % (self.nick, node.get_abs_node_path()))
       # print  self.nick, node.get_abs_node_path(), status

class Application(tki.Frame):
    def __init__(self, master=None, client=None, queue=None):
        tki.Frame.__init__(self, master)
        if client is None:
            self.__clients = [ Client("localhost@31415"), ]
        elif type(client) == set:
            self.__clients = client
        else: self.__clients = [ client ]

        self.__queue = queue
        width =  640
        height = 780
        self.canvas = tki.Canvas(width=width, height=height, bg='black')
        self.grid()
        self.createWidgets()
        self.canvas.after(50, self.check_queue)

    def createWidgets(self):
        rowx = 1
        glob = Label(tki.StringVar(self))
        root = self
        self.__menuBar = MenuBar(root)
        self.__menuBar.grid(row=0, column=0,sticky=tki.W)
        self.label = tki.Label(root,textvariable=Label.inst)
        self.label.grid(row=0, column=2,sticky=tki.E)
        self.__wins = dict()
        rowx += 1
        colx = 0
        kinds = ("active", "aborted", "submitted")
        for kind in kinds:
            self.__wins[kind] = TaskList(root, kind)
            self.__wins[kind].grid(row=rowx, column= colx,
                                   sticky=tki.S + tki.E + tki.W)
            colx += 1
        self.update()

    def check_queue(self):
        try:   self.__queue.get(block=False)
        except Queue.Empty: pass
        else: self.update()
        self.canvas.after(50, self.check_queue)

    def update(self):
        Label.update()
        for kind, win in self.__wins.items():
            win.clear()
        for client in self.__clients:
            if type(client) == list:
                for clt in client:
```

```
                    clt.process(self.__wins)
            else:
                try: client.process(self.__wins)
                except: pass
        pass

def get_username(): return pwd.getpwuid( os.getuid() )[ 0 ]
def get_uid():      return pwd.getpwnam(get_username()).pw_uid

if __name__ == '__main__':
    try:    port = 1500 + int(get_uid())
    except: port = 31415

    if len(sys.argv) == 0:
        clients = [ Client("localhost%d" % port) ]
    else:
        clients = []
        for num in xrange(1, len(sys.argv)):
            clients.append( Client(sys.argv[num]) )

    queue = Queue.Queue()
    # app = AppSms(host, port, queue)
    app = Application(client= clients, queue= queue)
    app.master.title(PROGRAM_NAME)
    app.columnconfigure(0, weight=1)
    app.rowconfigure(0, weight=1)

    PaceKeeper(app, queue)
    app.mainloop()
```