This page provides a detailed status of SmartScript's methods. For a summary, see [here](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptMissingMethods).

**abort[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "abort" \o "Link to this section)**

* status: works in TO11.DR3. .
* def execute(self, variableElement):
* # Demonstrate that the tool is executing
* print "This is the first line."
* print "This is the second line."
* # A dialog should appear that displays this message text
* self.abort("Aborting the testAbort tool.")
* print "This message should not appear in the terminal window."
* return variableElement

**assignValue[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "assignValue" \o "Link to this section)**

* assignValue() - does not exist in awips2.
  + solution: call assignValueCmd() directly.
* background
  + first, notice that assignValueCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def assignValueCmd(self, elements, timeRange, value):
  + ...
  + and that in awips1, assignValue() calls assignValueCmd(), but first creates a TimeRange
  + # awips1
  + def assignValue(self, elements, hourStart, hourEnd, modelBase, value):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.assignValueCmd(elements, timeRange, value)
* to port, first create a TimeRange, then call assignValueCmd()
* # example: PopulateMOSGuide.Procedure
* # awips1
* self.assignValue(["T"], begin\_time, end\_time, "Fcst", 75)
* # awips2
* dbid = self.findDatabase("Fcst")
* timerange = self.createTimeRange(begin\_time, end\_time, "Database", dbid)
* self.assignValueCmd(["T"], timerange, 75)

**assignValueCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "assignValueCmd" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement, GridTimeRange):
* self.assignValueCmd(["T", "Td"], GridTimeRange, 50.0)
* return variableElement

**availableDatabases[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "availableDatabases" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement ):
* availableDBs = self.availableDatabases()
* for db in availableDBs:
* print db.modelName(), db.modelIdentifier()
* return variableElement

**availableParms[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "availableParms" \o "Link to this section)**

* works in TO11-DR5 - Elapsed time improved from 2 minutes (DR3) to 3.5 seconds (DR5)
* def execute(self, variableElement ):
* t1 = time.time()
* availableParms = self.availableParms()
* elapsedTime = time.time() - t1
* for parm in availableParms:
* print parm
* print "ElapsedTime:", elapsedTime
* return variableElement

**cacheElements[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "cacheElements" \o "Link to this section)**

* deprecated in DR3 and DR4. Currently a no-op:
* ##
* # @param elementNames: ignored
* #
* # @deprecated: Cacheing is controlled by the system.
* def cacheElements(self, elementNames):
* pass

**callProcedure[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "callProcedure" \o "Link to this section)**

* works in TO11-DR3
* # Main calling procedure
* def execute(self, editArea, timeRange, varDict):
* print "running testCallProcedure."
* self.callProcedure("testProc")
* print "returned from calling testProc."
* return
* # This is the procedure that gets called by the main procedure
* def execute(self, editArea, timeRange, varDict):
* print "Executing procedure testProc."
* return

**callSmartTool[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "callSmartTool" \o "Link to this section)**

* works in DR10.
* Procedure follows
* def execute(self, timeRange):
* print "Calling testTool1 with timeRange."
* self.callSmartTool("testTool1", "T", timeRange=timeRange)
* print "Returned from calling testTool1."
* print "Calling testTool2 with timeRange."
* self.callSmartTool("testTool2", "T", timeRange=timeRange)
* print "Returned from calling testTool2."
* return
* testTool1 follows - name this tool testTool1
* def execute(self, variableElement, GridTimeRange):
* print "This is smartTool named: testTool1."
* print "GridTimeRange:", GridTimeRange
* return variableElement
* testTool2 follows - name this tool testTool2
* def execute(self, variableElement):
* print "This is smartTool named: testTool2."
* return variableElement

**cancel[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "cancel" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement):
* print "Running tool testCancel."
* self.cancel()
* print "Past cancel command. You should not see this."
* return variableElement

**clearActiveEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "clearActiveEditArea" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement ):
* # clear the active edit area
* mask = self.clearActiveEditArea()
* return variableElement

**combineMode[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "combineMode" \o "Link to this section)**

* works in TO11-DR5
* def execute(self, variableElement):
* mode = self.combineMode()
* print "CombineMode now:", mode
* return variableElement

**copy[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "copy" \o "Link to this section)**

* copy() - does not exist in awips2.
  + solution: call copyCmd() directly.
* background
  + first, notice that copyCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def copyCmd(self, elements, databaseID, timeRange):
  + ...
  + and that in awips1, copy() calls copyCmd(), but first creates a TimeRange
  + # awips1
  + def copy(self, elements, hourStart, hourEnd, modelSource):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelSource)
  + self.copyCmd(elements, databaseID, timeRange)
* to port, first create a TimeRange, then call copyCmd()
* # example: Copy\_HPCGrids.Procedure
* #awips1
* self.copy(['MaxT', 'Sky', 'Td', 'MinT', 'Wind'], 90, 187,'HPCGuide')
* self.copy(['PoP'], 84, 180, 'HPCGuide')
* self.copy(['MinT'], 192, 193, 'DGEX')
* # awips2
* hpcid = self.findDatabase('HPCGuide')
* dgexid = self.findDatabase('DGEX')
* timerange = self.createTimeRange(90, 187, "Database",hpcid)
* self.copyCmd(['MaxT', 'Sky', 'Td', 'MinT', 'Wind'], hpcid, timerange)
* timerange = self.createTimeRange(84, 180, "Database",hpcid)
* self.copyCmd(['PoP'], hpcid, timerange)
* timerange = self.createTimeRange(192, 193, "Database",dgexid)
* self.copyCmd(['MinT'], dgexid, timerange)

**copyCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "copyCmd" \o "Link to this section)**

* Works in TO11-DR3
* The following PROCEDURE works.
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testCopyCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Edit"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* print "testCopyCmd"
* elementList = ["T", "Wind"]
* modelDBID = self.findDatabase("GFS40")
* print "model:", modelDBID.modelIdentifier()
* self.copyCmd(elementList, modelDBID, timeRange)
* return

**copyToCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "copyToCmd" \o "Link to this section)**

* Works in TO11-D4
* This PROCEDURE works in DR4
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # $testCopyToCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* # The MenuItems list defines the GFE menu item(s) under which the
* # Procedure is to appear.
* # Possible items are: Populate, Edit, Consistency, Verify, Hazards
* MenuItems = ["Edit"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* elementList = [("T", "T"), ("Td", "Td")]
* modelDBID = self.findDatabase("NAM12")
* print "timeRange:", timeRange
* print "DB:", modelDBID.modelIdentifier()
* self.copyToCmd(elementList, modelDBID, timeRange)
* return

**createFromScratch[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "createFromScratch" \o "Link to this section)**

* createFromScratch() - does not exist in awips2.
  + solution: call createFromScratchCmd() directly.
* background
  + first, notice that createFromScratchCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def createFromScratchCmd(self, elements, timeRange, repeat=0, duration=0):
  + ...
  + and that in awips1, createFromScratch() calls createFromScratchCmd(), but first creates a TimeRange
  + # awips1
  + def createFromScratch(self, elements, hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.createFromScratchCmd(elements, timeRange)
* to port, first create a TimeRange, then call createFromScratchCmd()
* # example: DiurnalTempTool.Procedure
* #awips1
* self.createFromScratch(["T"], lhour, lhour+1, "Fcst")
* # awips2
* dbid = self.findDatabase('Fcst')
* timerange = self.createTimeRange(lhour, lhour+1, "Database", dbid)
* self.createFromScratchCmd(['T'], timerange)

**createFromScratchCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "createFromScratchCmd" \o "Link to this section)**

Works for me in DR4 when called from a procedure. See [RNK\_Collaborate\_PoP\_Snow\_QPF.py](https://collaborate.nws.noaa.gov/trac/ncladt/browser/trunk/gfe/rnk/procedures/RNK_Collaborate_PoP_Snow_QPF.py). Not sure if it is valid to call this from a SmartTool.

* works - the following PROCEDURE works

# ----------------------------------------------------------------------------

# This software is in the public domain, furnished "as is", without technical

# support, and with no warranty, express or implied, as to its usefulness for

# any purpose.

#

# testCreateFromScratchCmd.py

#

# Author: lefebvre

# ----------------------------------------------------------------------------

MenuItems = ["Edit"]

import LogStream, time

from math import \*

import time

import AbsTime

import TimeRange

import SmartScript

class Procedure (SmartScript.SmartScript):

def \_\_init\_\_(self, dbss):

SmartScript.SmartScript.\_\_init\_\_(self, dbss)

def execute(self):

oneDay = 86400

start = int(time.time() / 3600) \* 3600

end = start + oneDay

aStart = AbsTime.AbsTime(start)

aEnd = AbsTime.AbsTime(start + oneDay)

timeRange = TimeRange.TimeRange(aStart, aEnd)

print "TimeRange:", timeRange

elementList = ["T"]

print "ElementList:", elementList

self.createFromScratchCmd(elementList, timeRange, 1, 1)

**createGrid[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "createGrid" \o "Link to this section)**

* works when called from a procedure
* def execute(self, timeRange):
* topo = self.getTopo()
* tGrid = topo \* 0.0
* tGrid = tGrid + 55.5
* self.createGrid("Fcst", "Td", "SCALAR", tGrid, timeRange)
* return

**createTimeRange[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "createTimeRange" \o "Link to this section)**

* createTimeRange() works. Example
* def execute(self):
* "SmartScript.createTimeRange() test"
* myTR = self.createTimeRange(1, 3)
* print "myTR (LT default) =", myTR
* # same test as above, since "LT" is the default
* myTR = self.createTimeRange(1, 3, "LT")
* print "myTR (LT specified) =", myTR
* myTR = self.createTimeRange(1, 3, "Z")
* print "myTR (Z) = ", myTR
* dbID = self.findDatabase("RUC80")
* if dbID.javaDbId().isValid():
* print "findDatabase(RUC80) returned:", dbID.modelIdentifier()
* myTR = self.createTimeRange(1, 3, dbID)
* #myTR = self.createTimeRange(1, 3, dbID.javaDbId()) # this works too
* print "myTR (Database) =", myTR
* myTR = self.createTimeRange(1, 3, "Database") # correctlty throws exception
* print("we shouldn't see this. An exception should be thrown")
* return None

**dayTime[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "dayTime" \o "Link to this section)**

* TO11 slice 6 update: *this call now works*!
* kept for historical purposes:
* def dayTime(self, timeRange, startHour=6, endHour=18):
* # Return 1 if start of timeRange is between the
* # startHour and endHour, Return 0 otherwise.
* # Assume timeRange is GMT and convert to local time.
* shift = self.determineTimeShift()
* startTime = timeRange.startTime() + shift
* localStartHour = startTime.hour() // FAILS!
* if localStartHour >= startHour and localStartHour < endHour:
* return 1
* else:
* return 0

**decodeEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "decodeEditArea" \o "Link to this section)**

* status: FAILS in TO11.DR7.
* def execute(self, variableElement):
* ea = self.getEditArea(self.getSiteID())
* print dir(ea)
* grid2DBit = self.getGrid2DBit(ea)
* print "got Grid2DBit"
* dir(grid2DBit)
* return variableElement
* Traceback follows
* Error executing testGetGrid2DBit: jep.JepException: <type 'exceptions.NameError'>: Method not found convertToGridpoints >>> File "<string>", line 1, in <module>
* File "/awips2/cave/etc/gfe/userPython/utilities/SmartToolInterface.py", line 153, in runTool
* return self.runMethod(moduleName, className, methodName, \*\*kwargs)
* File "/home/lefebvre/caveData/common/base/python/MasterInterface.py", line 113, in runMethod
* result = methodObj(\*\*kwargs)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/smartTools/testGetGrid2DBit.py", line 27, in execute
* grid2DBit = self.getGrid2DBit(ea)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/utilities/SmartScript.py", line 1480, in getGrid2DBit
* return self.\_\_refSetMgr.convertToGridpoints(
* jep.JepException: jep.JepException: <type 'exceptions.NameError'>: Method not found convertToGridpoints >>> File "<string>", line 1, in <module>
* File "/awips2/cave/etc/gfe/userPython/utilities/SmartToolInterface.py", line 153, in runTool
* return self.runMethod(moduleName, className, methodName, \*\*kwargs)
* File "/home/lefebvre/caveData/common/base/python/MasterInterface.py", line 113, in runMethod
* result = methodObj(\*\*kwargs)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/smartTools/testGetGrid2DBit.py", line 27, in execute
* grid2DBit = self.getGrid2DBit(ea)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/utilities/SmartScript.py", line 1480, in getGrid2DBit
* return self.\_\_refSetMgr.convertToGridpoints(
* at jep.Jep.eval(Jep.java:300)
* at com.raytheon.uf.common.python.PythonScript.internalExecute(PythonScript.java:204)
* at com.raytheon.viz.gfe.smarttool.script.SmartToolController.runToolMethod(SmartToolController.java:143)
* at com.raytheon.viz.gfe.smarttool.script.SmartToolController.executeTool(SmartToolController.java:160)
* at com.raytheon.viz.gfe.smarttool.Tool.numeric(Tool.java:591)
* at com.raytheon.viz.gfe.smarttool.Tool.execute(Tool.java:477)
* at com.raytheon.viz.gfe.smarttool.script.SmartToolJob.run(SmartToolJob.java:97)
* at org.eclipse.core.internal.jobs.Worker.run(Worker.java:55)
* Caused by: jep.JepException: <type 'exceptions.NameError'>: Method not found convertToGridpoints >>> File "<string>", line 1, in <module>
* File "/awips2/cave/etc/gfe/userPython/utilities/SmartToolInterface.py", line 153, in runTool
* return self.runMethod(moduleName, className, methodName, \*\*kwargs)
* File "/home/lefebvre/caveData/common/base/python/MasterInterface.py", line 113, in runMethod
* result = methodObj(\*\*kwargs)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/smartTools/testGetGrid2DBit.py", line 27, in execute
* grid2DBit = self.getGrid2DBit(ea)
* File "/home/lefebvre/caveData/etc/user/lefebvre/gfe/userPython/utilities/SmartScript.py", line 1480, in getGrid2DBit
* return self.\_\_refSetMgr.convertToGridpoints(
* at jep.Jep.eval(Native Method)
* at jep.Jep.eval(Jep.java:284)
* ... 7 more

**deleteCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "deleteCmd" \o "Link to this section)**

* works in TO11-DR3
* This PROCEDURE works
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testDeleteCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* # The MenuItems list defines the GFE menu item(s) under which the
* # Procedure is to appear.
* # Possible items are: Populate, Edit, Consistency, Verify, Hazards
* MenuItems = ["Edit"]
* import LogStream, time
* from math import \*
* import SmartScript
* ## For documentation on the available commands,
* ## see the SmartScript Utility, which can be viewed from
* ## the Edit Actions Dialog Utilities window
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* print "GridTimeRange:", timeRange
* self.deleteCmd(["T", "Td"], timeRange)

**deleteGrid[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "deleteGrid" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement, GridTimeRange):
* self.deleteGrid("Fcst", "Td", "SFC", GridTimeRange)
* return variableElement

**deleteObject[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "deleteObject" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement):
* myList = [0, 1, 2, 3, 4, 5]
* self.saveObject("MyList", myList, "TestCategory")
* newList = self.getObject("MyList", "TestCategory")
* if myList == newList:
* print "testDeleteObject passed first test."
* else:
* print "testDeleteObject Failed."
* self.deleteObject("MyList", "TestCategory")
* try: # this should fail
* newList = self.getObject("MyList", "TestCategory")
* print "testDeleteObject Failed."
* except: # means no object found, which is good
* print "testDeleteObject passed delete test."
* return variableElement

**determineTimeShift[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "determineTimeShift" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement):
* ts = self.determineTimeShift()
* print "TimeShift:", ts
* return variableElement

**diff2**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#diff2)

* works in TO11-DR3 - not sure if this is used at all in Smart tools.
* def execute(self, variableElement):
* l = [0, 1, 2, 3, 4, 5, 7, 9, 11, 13, 15, 19, 23]
* dGrid = self.diff2(l)
* print "Derivative:", dGrid
* return variableElement

**directionTaperGrid[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "directionTaperGrid" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement):
* ea = self.getEditArea(self.getSiteID())
* tGrid = self.directionTaperGrid(ea, "N")
* return tGrid

**eaMgr[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "eaMgr" \o "Link to this section)**

* ...

**editAreaList[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "editAreaList" \o "Link to this section)**

* works in TO11-DR3
* def execute(self, variableElement):
* eaList = self.editAreaList()
* print "EditArea List:", eaList
* return variableElement

**encodeEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "encodeEditArea" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* # Fetch the editArea TBW
* editArea = self.getEditArea("TBW")
* print "Got EditArea:", editArea
* mask = self.encodeEditArea(editArea)
* print sum(sum(mask)), "points in TBW edit area."

**errorReturn[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "errorReturn" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* self.errorReturn(1, "This is an error message.")
* return variableElement

**esat[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "esat" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* newGrid = self.esat(variableElement)
* return newGrid

**findDatabase[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "findDatabase" \o "Link to this section)**

* status: works in TO11-DR3.
* findDatabase always returns a database object even if no database is actually there. Check for valid database with .isValid() method.
* def execute(self, variableElement):
* dbList = ["GFS40", "RUC13", "NAM12"]
* for db in dbList:
* dbID = self.findDatabase(db)
* if dbID.isValid():
* print dir(dbID)
* else:
* print "Model does not exist:",db
* return variableElement

**forceLock[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "forceLock" \o "Link to this section)**

* status: works in TO11-DR7.
* def execute(self, variableElement, GridTimeRange):
* start = GridTimeRange.startTime()
* end = GridTimeRange.endTime()
* print "start:", start, "end:", end
* self.forceLock("Td", "SFC", start, end)
* return variableElement

**fragment[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "fragment" \o "Link to this section)**

* fragment() - does not exist in awips2.
  + solution: call fragmentCmd() directly.
* background
  + first, notice that fragmentCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def fragmentCmd(self, elements, timeRange):
  + ...
  + and that in awips1, fragment() calls fragmentCmd(), but first creates a TimeRange
  + # awips1
  + def fragment(self, elements, hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.fragmentCmd(elements, timeRange)
* to port, first create a TimeRange, then call fragmentCmd()
* # example: runQPFGroupAuto.Procedure
* #awips1
* self.fragment(['QPFhrly'], -48, 240, 'QPFhrly')
* # awips2
* dbid = self.findDatabase('QPFhrly')
* timerange = self.createTimeRange(-48, 240, "Database", dbid)
* self.fragmentCmd(['QPFhrly'], timerange)

**fragmentCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "fragmentCmd" \o "Link to this section)**

* works in DR4
* The PROCEDURE below works in DR4
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testFragmentCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Edit"]
* import LogStream, time
* from math import \*
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* self.fragmentCmd(["RH", "T"], timeRange)
* return

**getActiveEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getActiveEditArea" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement ):
* # this tool should set the active Area to TBW
* editArea = self.getActiveEditArea()
* # Convert to a mask
* mask = self.encodeEditArea(editArea)
* # print the number or points
* print sum(sum(mask)), "points in active edit area."
* return variableElement

**getActiveElement[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getActiveElement" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* activeElement = self.getActiveElement()
* print "Active Element:", activeElement.expressionName()
* return variableElement

**getComposite[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getComposite" \o "Link to this section)**

* status: Works partially but doesn't work the same as AWIPS1
* The method should return two grids: a bitmap defining the ISC area and a forecast grid.
* This method returns a grid of bytes which represents the bit mask, but no grid is returned.
* This tool works without error in AWIPS2, but not the same as AWIPS1
* def execute(self, variableElement, GridTimeRange):
* grid = self.getComposite("T", GridTimeRange)
* return grid.astype(float32)
* Update 30 Dec. 2010. Version R7 - Tom LeFebvre
* New ISC functionality was included in version R7, so the CopyFromModelISC tool was ported to AWIPS II. I discovered that the getComposite method does not work the same as in AWIPS I. To make the deadline for checking in this tool, I wrote a workaround that duplicates most of the functionality of this method. The result is that the tool works, but in operational ISC conditions, the grid returned from the ISC database will not be exactly the same as in AWIPS I. Once getComposite is working properly, the tool will be modified to it rather than the workaround. The workaround code is listed below.
* # Temporary replacement for SmartScript's getComposite.
* # Returns a grid consisting of the Fcst grid over the WFO edit area
* # and the ISC grid everywhere else.
* def getISCComposite(self, weName, timeRange, exactMatch=1, onlyISC=0):
* iscDB = self.findDatabase("ISC")
* # get the type of the grid
* parm = self.getParm(iscDB, weName, "SFC")
* parmInfo = parm.getGridInfo()
* gridType = parmInfo.getGridType()
* bits = self.getTopo() \* 0
* if gridType.equals(gridType.SCALAR):
* print "It's a SCALAR grid."
* iscGrid = self.getGrids(iscDB, weName, "SFC", timeRange, mode = "First")
* if iscGrid is not None:
* bits = bits + 1 # set all grid point to 1
* return bits, iscGrid
* elif gridType.equals(gridType.VECTOR):
* print "It's a VECTOR grid."
* iscMagGrid, iscDirGrid = self.getGrids(iscDB, weName, "SFC", timeRange, mode="First")
* if iscMagGrid is not None and iscDirGrid is not None:
* bits = bits + 1 # set all grid point to 1
* return bits, iscMagGrid, iscDirGrid
* elif gridType.equals(gridType.DISCRETE):
* print "It's a DISCRETE grid."
* byteGrid, keys = self.getGrids(iscDB, weName, "SFC", timeRange, mode="First")
* if byteGrid is not None:
* bits = bits + 1 # set all grid point to 1
* return bits, byteGrid
* elif gridType.equals(gridType.WEATHER):
* print "It's a WEATHER grid."
* byteGrid, keys = self.getGrids(iscDB, weName, "SFC", timeRange, mode="First")
* if byteGrid is not None:
* bits = bits + 1 # set all grid point to 1
* return bits, byteGrid, keys
* return None

**getConfigItem[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getConfigItem" \o "Link to this section)**

* status: fails in TO11-DR3. Returns "None" always
* def execute(self, variableElement):
* value = self.getConfigItem("EditorTimeLine\_color")
* print "EditorTimeLine\_color:", value
* return variableElement
* dr4 update: works. Returns "Yellow" for me - john.olsen
* Retested on DR4 at GSD and I still get a None response. - Tom LeFebvre

**getD2Dmodel**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#getD2Dmodel)

* status: works in TO11.D4.
* def execute(self, variableElement):
* dbid = self.findDatabase("NAM12")
* dbName = self.getD2Dmodel(dbid)
* print "database name:", dbName
* return variableElement

**getDatabase[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getDatabase" \o "Link to this section)**

* status: works in TO11.D4.
* def execute(self, variableElement, varDict):
* model = self.findDatabase("GFS40")
* print "modelID string:", model.modelIdentifier()
* dbid = self.getDatabase(model.modelIdentifier())
* print "databaseID from getDatabase:", dbid.modelIdentifier()
* return variableElement

**getDefaults[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getDefaults" \o "Link to this section)**

* getDefaults() - does not exist in awips2
* # awips1
* def getDefaults(self, hourStart, hourEnd, modelBase):
* databaseID = self.findDatabase(modelBase)
* timeRange = self.createTimeRange(hourStart, hourEnd, "Database", databaseID)
* return databaseID, timeRange
  + no SITE-level tool makes this call. It is called from awips1's SmartScript() only.
  + tools that call getDefaults() will instead need to make separate calls to findDatabase() and createTimeRange().

**getDiscreteKeys[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getDiscreteKeys" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* keys = self.getDiscreteKeys("Wx")
* print "Discrete keys for Wx."
* for k in keys:
* print k
* return variableElement

**getEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getEditArea" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* # Fetch the editArea TBW
* editArea = self.getEditArea("TBW")
* print "Got EditArea:", editArea

**getExprName[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getExprName" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* fcstName = self.getExprName("Fcst", "T")
* print "Fcst exprName:", fcstName
* gfsName = self.getExprName("GFS40", "T")
* print "GFS40 exprName:", gfsName
* return variableElement

**getGrid2DBit**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#getGrid2DBit)

* status: Deprecated. This is an old method that apparently broke a while back.
* Only two tools refer to this method but don't really use it.
* Suggest this method be removed from SmartScript in AWIPS II.

**getGridCell[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridCell" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* # Note!!! Your lat/lon should be specific to your domain
* x, y = self.getGridCell(27.81, -82.14)
* print "X:", x, "Y:", y
* return variableElement

**getGridCellSwath[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridCellSwath" \o "Link to this section)**

* status: works in TO11-DR7.
* def execute(self, variableElement):
* # If this works, the result will be the WFO editarea plus 5 grid cells in all directions
* ea = self.getEditArea(self.getSiteID())
* swath = self.getGridCellSwath(ea, 5)
* self.setActiveEditArea(swath)
* return variableElement

**getGridHistory[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridHistory" \o "Link to this section)**

* status: works in TO11-DR7.
* def execute(self, variableElement, GridTimeRange):
* print "GridTimeRange:", GridTimeRange
* gridHist = self.getGridHistory("Fcst", "T", "SFC", GridTimeRange)
* print "GridHist:", gridHist
* return variableElement

**getGridInfo[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridInfo" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement, GridTimeRange):
* gridInfo = self.getGridInfo("Fcst", "T", "SFC", GridTimeRange)
* print "gridInfo:"
* for g in gridInfo:
* print "GridTime:", g.gridTime()
* print "min/maxLimits:", g.minLimit(), g.maxLimit()
* print
* return variableElement

**getGridLoc[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridLoc" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* gridLoc = self.getGridLoc()
* print "GridLoc:", gridLoc
* return variableElement

**getGrids[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGrids" \o "Link to this section)**

* status: works in TO11-DR3. Exceptions: Can't pass in strings like "NAM12" or D2D\_NAM12" to specify model.
* Must pass in a fully qualified databaseID object or string in oder to get this to work. e.g. "TBW\_GRID\_D2D\_NAM12\_20100219\_1200"
* Can call findDatabase and pass in a databaseID object, or the result of DatabaseID.modelIdentifier()
* DR10 UPDATE - Now errors are generated when attempting to access databases fetched with findDatabase.
* The first test case (NAM12 IFP grid) fails in the test case below. This was working in DR3-DR9.
* def execute(self, variableElement):
* oneDay = 24 \* 3600
* today12Z = (int(time.time() / oneDay) \* oneDay) + (oneDay / 2)
* absTime = AbsTime.AbsTime(today12Z)
* timeRange = TimeRange.TimeRange(absTime, absTime + 3600)
* # IFP database test
* dbList = []
* dbList.append(self.findDatabase("NAM12"))
* dbList.append("NAM12")
* for db in dbList:
* grid = self.getGrids(db, "T", "SFC", timeRange, mode="First")
* if grid[100][100] > 20.0 and grid[100][100] < 120.0:
* print "IFP grid test passed with db:", db
* # D2D database test
* nam12D2Did = self.findDatabase("D2D\_GFS40")
* print nam12D2Did.modelIdentifier()
* grid = self.getGrids(nam12D2Did, "t", "MB500", timeRange, mode="First")
* print "value:", grid[100][100]
* if grid[100][100] > 200.0 and grid[100][100] < 320.0:
* print "D2D grid test passed."
* # this test causes errors in DR3 and DR10
* grid = self.getGrids("D2D\_NAM12", 't', "MB500", timeRange, mode="First")
* print "got IFP NAM12 grid...shape:", grid.shape
* return variableElement
* scripts that use the default mode of "TimeWtAverage" will not return multiple grids. To return multiple grids, use the "List" mode:
* def execute(self, timeRange):
* # When running this tool make sure to select a timeRange that contains multiple T grids
* print "TimeRange:", timeRange
* gridList = self.getGrids("Fcst", "T", "SFC", timeRange, mode="List")
* print "type:", type(gridList)
* print "Got", len(gridList), "from getGrids."
* return

**getGridTimes[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getGridTimes" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* now = int(time.time() / 3600) \* 3600
* startTime = AbsTime.AbsTime(now)
* timeRange, trList = self.getGridTimes("Fcst", "T", "SFC", startTime, 24)
* print "timeRange:", timeRange
* for tr in trList:
* print "TR:", tr
* return variableElement

**getIndex[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getIndex" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* byteGrid, key = variableElement
* error = False
* for k in range(len(key)):
* i = self.getIndex(key[k], key)
* if i != k:
* print "Error. getIndex failed."
* error = True
* if not error:
* print "testGetIndex passed."
* return variableElement

**getindicies[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getindicies" \o "Link to this section)**

* status: works in TO11.D3. Generally used inside SmartScript
* def execute(self, variableElement):
* a, b = self.getindicies(2, 7)
* print a
* print b
* return variableElement

**getLatLon[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getLatLon" \o "Link to this section)**

* status: works in TO11.6. Worked in TO11.5 too.
* lat, lon = self.getLatLon(50.0, 50.0) # Note coords are floats
* print "Lat:", lat, "Lon:", lon
* return variableElement
* kept for historical purposes:
  + getLatLon() fails when it calls getGridLoc().latLonCenter(coords). getGridLoc() returns a GridLocation, which has no latLonCenter method
  + workaround: add a latLonCenter method to GridLocation.java
  + public Coordinate latLonCenter( Coordinate coord ) throws TransformException {
  + return( MapUtil.gridCoordinateToLatLon(coord,PixelOrientation.CENTER,this) );
  + }

**getModelName[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getModelName" \o "Link to this section)**

* status: works in TO11-DR3
* def execute(self, variableElement):
* model = self.findDatabase("NAM12")
* print "modelString:", model.modelIdentifier()
* modelName = self.getModelName(model.modelIdentifier())
* print "ModelName:", modelName
* return variableElement

**getNumericMeanValue[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getNumericMeanValue" \o "Link to this section)**

* status: works in TO11-DR7
* def execute(self, variableElement, GridTimeRange):
* meanValue = self.getNumericMeanValue("GFS12", "t", ["MB750", "MB700"], GridTimeRange)
* print "MeanValue:", meanValue
* return variableElement

**getObject[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getObject" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* myList = [0, 1, 2, 3, 4, 5]
* self.saveObject("MyList", myList, "TestCategory")
* newList = self.getObject("MyList", "TestCategory")
* if myList == newList:
* print "testGetObject passed."
* return variableElement

**getParm[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getParm" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* parm = self.getParm("Fcst", "T", "SFC")
* print "Parm OfficeType:", parm.getOfficeType()
* return variableElement

**getSamplePoints[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getSamplePoints" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* points = self.getSamplePoints()
* print "Active SamplePoints:", points
* morePoints = self.getSamplePoints("ISC\_Marker\_Set\_wfo")
* print "Saved SamplePoints:", morePoints
* return variableElement

**getSite4ID**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#getSite4ID)

* status: works in TO11-DR3.
* def execute(self, variableElement):
* id = self.getSite4ID(self.getSiteID())
* print "Site4ID:", id
* return variableElement

**getSiteID[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getSiteID" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* siteID = self.getSiteID()
* print "getSiteID returned:", siteID

**getStations[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getStations" \o "Link to this section)**

* status: does not exist AWIPS2.

**getTopo[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getTopo" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* topoGrid = self.getTopo()
* print "Got topo grid"
* # check for data
* topoSum = sum(sum(topoGrid))
* print "Avg. Topo value:", topoSum / (topoGrid.shape[0] \* topoGrid.shape[1])
* return variableElement

**getVectorEditMode[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "getVectorEditMode" \o "Link to this section)**

* status: works in TO11-DR5.
* def execute(self, variableElement):
* mode = self.getVectorEditMode()
* print "Vector Edit Mode is:", mode
* return variableElement

**gfeOperatingMode[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "gfeOperatingMode" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* mode = self.gfeOperatingMode()
* print "GFE OperatingMode:", mode
* return variableElement

**highlightGrids[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "highlightGrids" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement, GridTimeRange):
* self.highlightGrids("Fcst", "T", "SFC", GridTimeRange, "blue")
* return variableElement

**interpolate[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "interpolate" \o "Link to this section)**

* interpolate() - does not exist in awips2.
  + solution: call interpolateCmd() directly.
* background
  + first, notice that interpolateCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def interpolateCmd(self, elements, timeRange,
  + interpMode="GAPS", interpState="SYNC", interval=0,
  + duration=0):
  + ...
  + and that in awips1, interpolate() calls interpolateCmd(), but first creates a TimeRange
  + # awips1
  + def interpolate(self, elements, hourStart, hourEnd, modelBase,
  + interpMode="GAPS", interpState="SYNC", interval=0,
  + duration=0):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.interpolateCmd(elements, timeRange,
  + interpMode, interpState, interval, duration)
* to port, first create a TimeRange, then call interpolateCmd()
* # example: FillGaps.Procedure
* #awips1
* self.interpolate(['T','Td','RH','Wind','WindGust','WindChill','HeatIndex','Sky','Wx','PoP','QPF','SnowAmt'],0,228,'GFS40')
* # awips2
* dbid = self.findDatabase('GFS40')
* timerange = self.createTimeRange(0, 228, "Database", dbid)
* self.interpolateCmd(['T','Td','RH','Wind','WindGust','WindChill','HeatIndex','Sky','Wx','PoP','QPF','SnowAmt'], timerange)

**interpolateCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "interpolateCmd" \o "Link to this section)**

* status: works in TO11.D4.
* This PROCEDURE works in DR4
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testInterpolateCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Edit"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* self.interpolateCmd(["T", "Td"], timeRange)
* return

**itemName[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "itemName" \o "Link to this section)**

* an eaMgr-related method. Removed from SmartScript.

**loadedParms[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "loadedParms" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* loadedParms = self.loadedParms()
* for name, level, dbid in loadedParms:
* print "name:", name, "level:", level, "model:", dbid.modelName()
* return variableElement

**loadParm[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "loadParm" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* self.loadParm("Fcst", "Haines", "SFC")
* return variableElement

**loadWEGroup[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "loadWEGroup" \o "Link to this section)**

* status: works in TO11-DR7.
* def execute(self, variableElement):
* self.loadWEGroup("FireWx")
* return variableElement

**lockedByMe[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "lockedByMe" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* trList = self.lockedByMe("T", "SFC")
* print "TimeRanges locked by me:"
* for tr in trList:
* print tr
* return variableElement

**lockedByOther[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "lockedByOther" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* trList = self.lockedByOther("T", "SFC")
* print "Locked TimeRanges:"
* for tr in trList:
* print tr
* return variableElement

**MagDirToUV**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#MagDirToUV)

* status: works in TO11-DR3.
* def execute(self, variableElement):
* for mag in [10, 20]:
* for direct in [0, 45, 90, 135, 180, 225, 270, 315]:
* u, v = self.MagDirToUV(mag, direct)
* print direct, "at", mag, "knots converts to u:", u, "v:", v
* return variableElement

**makeHeadlineGrid[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "makeHeadlineGrid" \o "Link to this section)**

* status: works in TO11-DR7.
* def execute(self, Wind):
* mag, direct = Wind
* headlineTable =[(15.0, 'SmCrftHSADV'),
* (21.0, 'SmCrftADV'),
* (34.0, 'GaleWRN'),
* (47.0, 'StormWRN'),
* (67.0, 'HurcnFrcWindWRN'),
* ]
* hazGrid = self.makeHeadlineGrid(headlineTable, mag)
* return hazGrid

**makeNumericSounding[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "makeNumericSounding" \o "Link to this section)**

* status: works in TO11-DR3 and DR5. Exceptions: Can't pass in strings like "NAM12" or D2D\_NAM12" to specify model.
* Must pass in a fully qualified databaseID object or string in oder to get this to work. e.g. "TBW\_GRID\_D2D\_NAM12\_20100219\_1200"
* Can call findDatabase and pass in a databaseID object, or the result of DatabaseID.modelIdentifier()
* Passing in a short-cut like "NAM12" or "D2D\_NAM12" causes hundreds of errors and a 2 minute delay.
* def execute(self, variableElement, GridTimeRange):
* model = "TBW\_GRID\_D2D\_NAM12\_20100219\_1200" # either of these two ways of specifying the model works
* model = self.findDatabase("D2D\_NAM12") # But not short-cut ways
* levels = ["MB850", "MB800", "MB750", "MB700"]
* ghCube, tCube = self.makeNumericSounding(model, "t", levels, GridTimeRange)
* print "Cube shapes: gh:", ghCube.shape, "t shape:", tCube.shape
* return variableElement

**mutableID[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "mutableID" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* id = self.mutableID()
* print "mutableID:", id.modelName()
* return variableElement

**noData[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "noData" \o "Link to this section)**

* status: works in TO11-DR3. This method raises an exception.
* def execute(self, variableElement):
* self.noData()
* return variableElement

**offset[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "offset" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* offsetGrid = self.offset(variableElement, 50, 50)
* return offsetGrid

**parmName[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "parmName" \o "Link to this section)**

* an eaMgr-related method. Removed from SmartScript.

**publishElements[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "publishElements" \o "Link to this section)**

* status: works in TO11-DR3
* def execute(self, variableElement, GridTimeRange):
* start = GridTimeRange.startTime()
* end = start + 24 \* 3600
* timeRange = TimeRange.TimeRange(start, end)
* self.publishElements(["T", "Td", "Wind", "Wx"], timeRange)
* return variableElement

**refData[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "refData" \o "Link to this section)**

* an eaMgr-related method. Removed from SmartScript.

**remove[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "remove" \o "Link to this section)**

* remove() - does not exist in awips2.
  + solution: call deleteCmd() directly.
* background
  + first, notice that deleteCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def deleteCmd(self, elements, timeRange):
  + ...
  + and that in awips1, remove() calls deleteCmd(), but first creates a TimeRange
  + # awips1
  + def remove(self, elements, hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.deleteCmd(elements, timeRange)
* to port, first create a TimeRange, then call deleteCmd()
* # example: TCCoastalFloodThreat.Procedure
* #awips1
* self.remove(["SurgeHtPlusTide"], -24, 240, 'Fcst')
* # awips2
* dbid = self.findDatabase('Fcst')
* timerange = self.createTimeRange(-24, 240, "Database", dbid)
* self.deleteCmd(["SurgeHtPlusTide"], timerange)

**saveEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "saveEditArea" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* refData = self.getActiveEditArea()
* print "Type:", type(refData)
* self.saveEditArea("testSaveArea", refData)
* return variableElement

**saveElements[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "saveElements" \o "Link to this section)**

* status: works mostly in TO11-DR3. Saves all elements except Wx.
* Running the following tool saves all elements, but Wx is left unsaved.
* def execute(self, variableElement):
* self.saveElements(["T", "Td", "Wind", "Wx", "RH", "MaxT", "MinT", "Sky",
* "PoP", "QPF", "SnowAmt", "SnowLevel", "FzLevel"])
* return variableElement

**saveObject[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "saveObject" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* myList = [0, 1, 2, 3, 4, 5]
* self.saveObject("MyList", myList, "TestCategory")
* newList = self.getObject("MyList", "TestCategory")
* if myList == newList:
* print "testSaveObject passed."
* return variableElement

**selectedParms[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "selectedParms" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* parms = self.selectedParms()
* for pName, pLevel, dbid in parms:
* print pName, pLevel, dbid.modelName()
* return variableElement

**setActiveEditArea[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setActiveEditArea" \o "Link to this section)**

* status: works in TO11-DR3.
* # This tool will set the active area to TBW
* def execute(self, variableElement ):
* editArea = self.getEditArea("TBW")
* self.setActiveEditArea(editArea)

**setActiveElement[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setActiveElement" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement, GridTimeRange):
* self.setActiveElement("Fcst", "T", "SFC", GridTimeRange)
* return variableElement

**setCombineMode[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setCombineMode" \o "Link to this section)**

* status: works in TO11-DR5.
* def execute(self, variableElement):
* mode = self.setCombineMode("Combine")
* return variableElement

**setToolType[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setToolType" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* self.setToolType("numeric")
* return variableElement

**setUp[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setUp" \o "Link to this section)**

* ...

**setVectorEditMode[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "setVectorEditMode" \o "Link to this section)**

* status: works in TO11-DR5.
* def execute(self, variableElement):
* self.setVectorEditMode("Magnitude Only")
* return variableElement

**split[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "split" \o "Link to this section)**

* split() - does not exist in awips2.
  + solution: call splitCmd() directly.
* background
  + first, notice that splitCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def splitCmd(self, elements, timeRange):
  + ...
  + and that in awips1, split() calls splitCmd(), but first creates a TimeRange
  + # awips1
  + def split(self, elements, hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.splitCmd(elements, timeRange)
* to port, first create a TimeRange, then call splitCmd()
* # example: Copy\_Grids\_From\_Select\_Source.Procedure
* #awips1
* self.split(['Wx'],starth+12,starth+24,"HPCGuide")
* # awips2
* dbid = self.findDatabase('HPCGuide')
* timerange = self.createTimeRange(starth+12, starth+24, "Database", dbid)
* self.interpolateCmd(['Wx'], timerange)

**sortUglyStr[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "sortUglyStr" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, Wx):
* grid, keys = Wx
* for k in keys:
* print "Unsorted:", k
* sortedStr = self.sortUglyStr(k)
* print "Sorted:", sortedStr
* return Wx

**splitCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "splitCmd" \o "Link to this section)**

* status: works in TO11.D4.
* This PROCEDURE works.
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testSplitCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Populate"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* self.splitCmd(["T", "Td", "Wind", "Wx"], timeRange)

**statusBarMsg[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "statusBarMsg" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* self.statusBarMsg("Status Bar Message Test.", "R")
* return variableElement

**taperGrid[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "taperGrid" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* ea = self.getEditArea(self.getSiteID())
* taperGrid = self.taperGrid(ea, 5)
* return taperGrid

**timeShift[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "timeShift" \o "Link to this section)**

* timeShift() - does not exist in awips2.
  + solution: call timeShiftCmd() directly.
* background
  + first, notice that timeShiftCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def timeShift(self, elements, copyOnly, shiftAmount,
  + hourStart, hourEnd, modelBase):
  + ...
  + and that in awips1, timeShift() calls timeShiftCmd(), but first creates a TimeRange
  + # awips1
  + def timeShift(self, elements, copyOnly, shiftAmount,
  + hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.timeShiftCmd(elements, copyOnly, shiftAmount, timeRange)
* to port, first create a TimeRange, then call timeShiftCmd()
* # example: Extend\_Grids.Procedure
* #awips1
* self.timeShift(['MinT'],1,24,169,181,'DGEX')
* # awips2
* dbid = self.findDatabase('DGEX')
* timerange = self.createTimeRange(169, 181, "Database", dbid)
* self.timeShiftCmd(['Wx'], 1, 24, timerange)

**timeShiftCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "timeShiftCmd" \o "Link to this section)**

* status: works in TO11.D4.
* This PROCEDURE works.
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testTimeShiftCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Populate"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self,timeRange):
* self.timeShiftCmd(["T", "Td", "Wind", "Wx"], True, 6, timeRange)

**unCacheElements[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "unCacheElements" \o "Link to this section)**

* status: fails in TO11-DR3. Has no effect. Throws no error.
* Probably won't need this method since it was a solution to a bug in AWIPS I.
* Suggest we remove it.
* def execute(self, variableElement):
* print "Calling unCacheElements"
* self.unCacheElements("T", "Td")
* return variableElement

**unloadWE[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "unloadWE" \o "Link to this section)**

* status: works in TO11-DR3.
* def execute(self, variableElement):
* self.unloadWE("Fcst", "Haines", "SFC")
* return variableElement

**variableList-model[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "variableList-model" \o "Link to this section)**

* status: fails in TO11-DR3.
* This tool should solicit a model from the user.
* The result is a dialog that says "ERROR: ModelVar"
* VariableList = [("ModelVar" , "", "model")]
* import SmartScript
* class Tool (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, variableElement, varDict):
* value = varDict["ModelVar"]
* print value
* return variableElement

**UVToMagDir**[**¶**](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues#UVToMagDir)

* status: works in TO11-DR7.
* def execute(self, variableElement):
* for u in [-10.0, -5.0, 0.0, 5.0, 10.0]:
* for v in [-10.0, -5.0, 0.0, 5.0, 10.0]:
* mag, direct = self.UVToMagDir(u, v)
* print "u:", u, "v:", v, " converts to mag:", mag, "dir:", direct
* return variableElement

**wxMask[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "wxMask" \o "Link to this section)**

* status: works in TO11-DR3.
* This tool will select all points with Rain.
* def execute(self, Wx):
* mask = self.wxMask(Wx, ":R:")
* ea = self.decodeEditArea(mask)
* self.setActiveEditArea(ea)
* return Wx

**zero[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "zero" \o "Link to this section)**

* zero() - does not exist in awips2.
  + solution: call zeroCmd() directly.
* background
  + first, notice that zeroCmd() has the same signature in both awips1 and awips2
  + # awips1 and awips2
  + def zeroCmd(self, elements, timeRange):
  + ...
  + and that in awips1, zero() calls timeShiftCmd(), but first creates a TimeRange
  + # awips1
  + def zero(self, elements, hourStart, hourEnd, modelBase):
  + databaseID, timeRange = self.getDefaults(hourStart, hourEnd, modelBase)
  + self.zeroCmd(elements, timeRange)
* to port, first create a TimeRange, then call zeroCmd()
* # example: SHV\_Zero\_Wx\_PoP.Procedure
* #awips1
* self.zero(elements, 0, 192, 'MRF')
* # awips2
* dbid = self.findDatabase('MRF')
* timerange = self.createTimeRange(0, 192, "Database", dbid)
* self.zeroCmd(elements, timerange)

**zeroCmd[¶](https://collaborate.nws.noaa.gov/trac/ncladt/wiki/SmartScriptIssues" \l "zeroCmd" \o "Link to this section)**

* status: works in TO11.D4.
* This PROCEDURE works in DR4
* # ----------------------------------------------------------------------------
* # This software is in the public domain, furnished "as is", without technical
* # support, and with no warranty, express or implied, as to its usefulness for
* # any purpose.
* #
* # testZeroCmd.py
* #
* # Author: lefebvre
* # ----------------------------------------------------------------------------
* MenuItems = ["Populate"]
* import LogStream, time
* from math import \*
* import time
* import AbsTime
* import TimeRange
* import SmartScript
* class Procedure (SmartScript.SmartScript):
* def \_\_init\_\_(self, dbss):
* SmartScript.SmartScript.\_\_init\_\_(self, dbss)
* def execute(self, timeRange):
* self.zeroCmd(["T", "Td", "Wind", "Wx"], timeRange)
* return