

```
1 #!/usr/bin/env python
2
3 """Filename: SrbRegisterUtil.py
4 A utility to ingest simulation run output and Copy important files into SRB.
5
6 IMPORTANT: Run this script from a directory where the simulation output is
7 stored.
8 Example: SrbRegisterUtil.py is located in /home/head/Navy
9 output is located in /home/head/Navy/20120522001/
10 thus: %cd /home/head/Navy/20120522001
11 %../SrbRegisterUtil.py -r 20120522001
12
13 IMPORTANT: Please run Sinit before running this script with your credentials
14
15 NOTE:
16 1. Only 32 attributes can be registered to SRB at this moment. The first 32 tags
17 from Readme.xml is read
18 2. Unit consistency is dependent on the user writing correct and consistent
19 Readme.xml
20 3. Only 256 characters value can be stored in metadata. This script copies the
21 first 256 chars
22 4. Directory and file name with space character is not supported.
23
24 $Header: /cvs_repository/customers/HPCMP/testbed/NavyPilot/SrbRegisterUtil.py,v
25 1.5 2013/04/11 17:23:10 martin Exp $
26 """
27
28 try:
29     import os,glob,sys,getopt,pprint,subprocess,shlex
30     from xml.etree.ElementTree import ElementTree
31 except:
32     raise "Check you local Python Version. A minimum Python version required"
33
34 ##### CONFIGURATION #####
35
36 SCHEME="Name_Value"
37 COLLECTION_PATH=""
38 MAX_ATTRIBUTE=31 # start from 0 . E.g. val 19 means 20 attributes
39
40 KEY_MAXCHAR=15 # Value of maximum minus 1. For example
41 31 for 32 keychars to compensate for ending null chars
42 VAL_MAXCHAR=255 # Value of maximum minus 1. For example
43 255 for 256 valchars to compensate for ending null chars
44
45 class SrbRule:
46     pass
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
50 ruleRin = SrbRule()
51 ruleRin.name = 'restart files'
52 ruleRin.rule = '*rin*'
53 ruleRin.retention_period = 365
54 ruleRin.DR_behavior = "yes"
55
56
57 ruleXml = SrbRule()
58 ruleXml.name = 'Xml files'
59 ruleXml.rule = '*.xml'
60 ruleXml.retention_period = 365
61 ruleXml.DR_behavior = "yes"
62
63
64 ruleOcth = SrbRule()
65 ruleOcth.name = 'Output files for CTH'
66 ruleOcth.rule = 'oct*'
67 ruleOcth.retention_period = 365
68 ruleOcth.DR_behavior = "yes"
69
70 ruleHistory = SrbRule()
71 ruleHistory.name = 'History file'
72 ruleHistory.rule = ('hscth*')
73 ruleHistory.retention_period = 180
74 ruleHistory.DR_behavior = "no"
75
76 ruleHistory2 = SrbRule()
77 ruleHistory2.name = 'History file 2'
78 ruleHistory2.rule = ('hcth*')
79 ruleHistory2.retention_period = 180
80 ruleHistory2.DR_behavior = "no"
81
82
83 rulePlot = SrbRule()
84 rulePlot.name = 'Plot file'
85 rulePlot.rule = 'SPCTH/*'
86 rulePlot.retention_period = 180
87 rulePlot.DR_behavior = "no"
88
89 ruleRestart = SrbRule()
90 ruleRestart.name = 'Restart file'
91 ruleRestart.rule = 'RSCTH/*'
92 ruleRestart.retention_period = 45
93 ruleRestart.DR_behavior = "no"
94
95 ##ALE 3D##
96
97
98 ruleRestartALE3D = SrbRule()
99 ruleRestartALE3D.name = 'Restart file for ALE3D'
100 ruleRestartALE3D.rule = '*_*_*'
101 ruleRestartALE3D.retention_period = 45
102 ruleRestartALE3D.DR_behavior = "no"
103
104 rulePlotALE3D = SrbRule()
105 rulePlotALE3D.name = 'Plot file for ALE3D'
106 rulePlotALE3D.rule = '*_*.*'
```

```

107 rulePlotALE3D.retention_period = 45
108 rulePlotALE3D.DR_behavior = "no"
109
110 ruleHistoryALE3D = SrbRule()
111 ruleHistoryALE3D.name = 'history file for ALE3D'
112 ruleHistoryALE3D.rule = 'timehist.*.*/'
113 ruleHistoryALE3D.retention_period = 180
114 ruleHistoryALE3D.DR_behavior = "no"
115
116 lruleSetCTH =
    [ruleIn,ruleXml,ruleOcth,ruleHistory,ruleHistory2,rulePlot,ruleRestart]
117 lruleSetALE3D =
    [ruleIn,ruleRin,ruleXml,ruleHistoryALE3D,rulePlotALE3D,ruleRestartALE3D]
118
119
120
121
122
123 #####
124 def CheckPyVersion():
125     """ Function to check Minimum Python Version """
126     MIN_PYTHON_VER = 0x02060000
127     major          = MIN_PYTHON_VER >> 24
128     minor          = MIN_PYTHON_VER >> 16 & 0xff
129     micro          = MIN_PYTHON_VER >> 8 & 0xffff
130
131     if (sys.hexversion >= MIN_PYTHON_VER):
132         return True
133     else:
134         print "Local Python is older than required Python
135 {0}.{1}.{2}.".format(major,minor,micro)
136         return False
137 #end function
138
139 def PyParseTree(rootElement):
140     """ Function to parse tree elements (XML) """
141     dElements = {}
142     lOrderTag = []
143     lElements = rootElement.getiterator() #produces a list of elements
144     for elem in lElements:
145         key = elem.tag.replace(',',' ') #strip commas
146         val = elem.text.replace(',',' ') #strip commas
147         val = val.replace('\n',' ') # strip linefeed and carr return
148
149         if dElements.has_key(key[0:KEY_MAXCHAR]):
150             print "..Error. key ({0}) appears twice. Before
151 truncation key value is ({1}). Check your Readme.xml".format(key[0:KEY_MAXCHAR],
152 key)
153             sys.exit(2)
154         #end if
155
156         if len(val)>VAL_MAXCHAR or len(key)>KEY_MAXCHAR:
157             if len(val)>VAL_MAXCHAR:
158                 print "..Warning. Truncating value for key '{0}'
159 to {1} characters.".format(key,VAL_MAXCHAR)
160             elif len(key)>KEY_MAXCHAR:
161                 print "..Warning. Truncating key {0} to

```

```

    {1}").format(key, key[0:KEY_MAXCHAR])
158         #end if
159         dElements[key[0:KEY_MAXCHAR]]=val[0:VAL_MAXCHAR]
160         lOrderTag.append(key[0:KEY_MAXCHAR])
161     else:
162         dElements[key]=val
163         lOrderTag.append(key)
164     #endif
165 #endfor
166 return dElements, lOrderTag
167
168 def version():
169     print """$Header:
    /cvs_repository/customers/HPCMP/testbed/NavyPilot/SrbRegisterUtil.py,v 1.5
    2013/04/11 17:23:10 martin Exp $"""
170 #end function
171
172 def usage():
173     print """SrbRegisterUtil.py
174     usage: SrbRegisterUtil.py <arguments>
175         -h : This help
176         -r <RunID> : Unique simulation RunID
177         --version : Print the script version
178
179     Example
180         SrbRegisterUtil.py -r 20120502101
181     """
182 # end of usage function
183
184 def execMe(string, input_stderr=1):
185     """ Helper function to execute a command line. Return rc, stdout, stderr
    """
186
187     larg = shlex.split(string)
188     ###DEBUG: print string
189     ###DEBUG: print larg
190
191     if(not input_stderr == 1):
192         #supress stderr
193         fnull = open(os.devnull, 'w')
194         p = subprocess.Popen(larg, stdout=subprocess.PIPE, stderr=fnull)
195     else:
196         p = subprocess.Popen(larg, stdout=subprocess.PIPE)
197     #endif
198
199     out = p.communicate()[0]
200     return (p.returncode, out)
201 # end function
202
203 def execSMe(string):
204     """ Helper function to execute a command line using flag shell=True """
205     ###DEBUG: print "execSMe: running {0}".format(string)
206     p = subprocess.Popen(string, stdout=subprocess.PIPE, shell=True)
207     out = p.communicate()[0]
208     return (p.returncode, out)
209 # end function
210

```

```
211 def PyIsCollectionExist(string):
212     """ Helper function. Check if the relative path supplied by caller
213         as string exists. If positive, return true, else return false
214         """
215     status = False
216     print "Checking for existence of collection {0}...".format(string)
217     (rc,out) = execMe('Sls {0} > /dev/null 2>&1'.format(string), 2)
218     #suppress stderr output with second argument != 1
219     if rc == 0:
220         status = True
221
222     return status
223 # end function
224
225 def PyIsSinit():
226     """ Check if you have done an Sinit """
227     status = False
228
229     # If user is authenticated, Spwd (SRB print working directory) should
230     work
231     (rc,out) = execMe('Spwd')
232
233     if rc == 0:
234         status = True
235     else:
236         print **Error: make sure that Sinit and Senv are in your path
237         and rerun your Sinit command again.
238     return status
239
240 def PySputAll(COLLECTION_PATH):
241     """
242         Takes care of copying recursively to SRB
243
244         Example: current directory is /home/Navy/science
245         and target collection is homecollection/New_science
246
247         We want to avoid creating homecollection/New_science/science/<files>,
248         instead we want to put the files to homecollection/New_science/<files>
249         recursively.
250
251         If a user uses wildcard characters (*), it will be expanded up to 1500
252         characters. There is a risk that not all files will be saved.
253         More importantly, there could be an error because the last truncated
254         file is considered target directory while it may be wrong directory or
255         a regular file
256         """
257     status = False
258
259     (rc,out) = execSMe("Sput -Rf {0} {1}".format( os.getcwd()
260     ,COLLECTION_PATH))
261     if (rc == 0):
262         status = True
263
264     return status
```

```
262
263
264 #end function
265
266
267
268 def PyConvertToFileList(a_rule):
269     """ Convert a rule into list of files """
270
271     lFiles = []
272     if len(glob.glob(a_rule.rule)) == 0:
273         pass # this rule match nothing in this dir
274     else:
275         lFiles.append(a_rule.rule)
276
277     return lFiles
278
279 #end function
280
281
282
283
284
285
286 def main():
287     """ Main function. It takes one argument <RunID>
288
289     This is where the program begins
290
291     """
292     lOrderTag = []
293     if not CheckPyVersion():
294         sys.exit(1)
295
296     try:
297         opts,args = getopt.getopt(sys.argv[1:], "hvr:", ['version'])
298     except getopt.GetoptError, err:
299         #print help info and exit:
300         print str(err)
301         usage()
302         sys.exit(2)
303     RunID = None
304     for arg,val in opts:
305         if arg == "-h":
306             usage()
307             sys.exit()
308         elif arg == "--version" or arg == "-v":
309             version()
310             sys.exit()
311         elif arg == "-r":
312             RunID = val
313         else:
314             assert False, "unhandled option"
315
316     #endfor
317
318     # Try to open a file
```

```
319     try:
320         tree = ElementTree()
321         root = tree.parse("Readme.xml")
322
323         dElements, lOrderTag = PyParseTree(root)
324
325         if dElements['RunID'] != RunID:
326             print "Error! Argument RunID ({0}) does not match XML
file RunID ({1})".format(RunID,dElements['RunID'])
327             sys.exit()
328
329
330     except IOError:
331         print "I/O Error"
332         sys.exit(1)
333     else:
334         print "File Readme.xml is opened successfully."
335     #end of try except block
336
337
338     #Have you run Sinit?
339     if (PyIsSinit()):
340         print "Sinit is OK..."
341         status = 0
342     #end of IsSinit block
343
344     lfull_path = os.getcwd().split('/')
345     COLLECTION_PATH=RunID
346
347     if(PyIsCollectionExist(COLLECTION_PATH)):
348         print "Collection {0} exists. Stop. Danger of overwriting older
collection. Please remove collection manually before restarting this
script.".format(COLLECTION_PATH)
349         status = 1      # I have to stop here. Danger of overwriting
existing collection
350     else:
351         print "Using new collection {0} ....".format(COLLECTION_PATH)
352
353     #end if
354
355     if status == 0 :
356
357         #start Ingesting files to SRB here
358         if(not PySputAll(COLLECTION_PATH)):
359             print "Sput call has failed"
360             status = 1
361         #end if
362     #end if
363
364     ###
365     #Set Retention_Period and DR_behavior based on lruleSet
366     #####
367     if dElements['AppCode'] == 'ALE3D':
368         lruleSet = lruleSetALE3D
369     elif dElements['AppCode'] == 'CTH':
370         lruleSet = lruleSetCTH
371     else:
```

```

372         print "Error. Unable to select ruleSet for application code {0}.
Perhaps you should declare and set lruleSet{1}".format(dElements['AppCode'])
373         sys.exit(1)
374     #end if
375
376
377     if status == 0 :
378         for my_rule in lruleSet:
379             target = PyConvertToFileList(my_rule)
380             if len(target) > 0: #ignore empty targets (rules that
doesn't match)
381                 (rc,out) = execSMe("Sscheme -w -R -val
Admin.Retention_Period::{0} {1}/{2}".format(my_rule.retention_period,
COLLECTION_PATH, target[0]))
382                 if rc != 0:
383                     print "Unable to set retention period
for rule: {0} to: {1}".format(my_rule.name,my_rule.retention_period)
384                 #end if
385
386                 (rc,out) = execSMe("Sscheme -w -R -val
Admin.DR_Behavior::{0} {1}/{2}".format(my_rule.DR_behavior, COLLECTION_PATH,
target[0]))
387                 if rc != 0:
388                     print "Unable to set DR_behavior for
rule: {0} to: {1}".format(my_rule.name, my_rule.DR_behavior)
389                 #end if
390             #end if
391
392
393         # end of for loop
394
395     #end if
396
397     ###
398     #Check the existence of Name_Value scheme set in SCHEME var
399     ###
400
401     if status == 0 :
402         # Command to get the scheme listing. Sgets Name_Value
403         (rc,out) = execMe("Sgets {0}".format(SCHEME))
404         if rc != 0:
405             status = 1
406             print "***Unable to find Schema ({0}). Ask your admin to
create it for you.".format(SCHEME)
407         elif SCHEME not in out:
408             status = 1
409             print "***Unable to find Schema ({0}). Ask your admin to
create it for you.".format(SCHEME)
410
411
412
413     if status == 0:
414         ###
415         # Write Metadata to collection. Commandline equivalent is:
416         # Scheme -w -scheme Name_Value -val Value[1]::"20120518001" PTW1800
417         #
418         # and so on... For brevity, you can chain up the name value pair as

```



```
    comma, separated list like so:
419     # Name[0]::"AppCode",Value[0]::"ALE3D" and so on...
420     ###
421     del lOrderTag[0] # sentinel element
422     lKeys = sorted(dElements.keys())
423
424     sBeginCmd = "Sscheme -w -scheme {0} -val ".format(SCHEME)
425     iCnt = 0
426
427     # Let's read our dictionary from XML file and register
428     # each name_value pair one by one
429     #
430     for key in lOrderTag:
431         if iCnt > MAX_ATTRIBUTE:
432             break
433         sCmd = sBeginCmd +
434         "Name[{0}]::\"{1}\",Value[{2}]::\"{3}\"".format(iCnt,key,iCnt,dElements[key]) +
435         "\" \" + COLLECTION_PATH
436         (rc,out) = execMe(sCmd)
437         if (rc != 0):
438             print "**Unable to assign metadata {0}={1} to
439             this collection".format(key,dElements[key])
440             status = 1
441             #end if
442
443             iCnt = iCnt+1
444             #end of for loop
445
446     #end of if
447
448     if status == 0:
449         print "Success. all metadata has been recorded"
450     else:
451         print "There is an error in this operation. See error message
452         above and run again"
453
454 #end of main function
455
456 if __name__ == "__main__":
457     main()
458
```