LEXISNEXIS RISK SOLUTIONS

Details of HtS3’s Perl Copy Scripts that Copy To/From S3 Buckets

Contents

Introduction

This document covers perl scripts that copy files to/from THOR nodes from/to S3 buckets. HtS3 uses these perl scripts to do the copies requested by the user. All these perl scripts exist in the hooks directory of the HPCCtS3 repository: https://github.com/tlhumphrey2/HPCCtS3/tree/master/juju-charms/precise/hpcc/hooks.

Documentation for using HtS3 to configure and deploy an HPCC System to AWS as well as copy files from/to THOR nodes to/from S3 buckets can be obtained here: https://github.com/tlhumphrey2/HPCCtS3/blob/master/UsingHtS3.pdf (to download, click on "view the full file").

All of these perl scripts reside and execute on THOR nodes, i.e. EC2 instances. Currently, they are copied to the nodes of the deployed THOR by Juju Charm, the tool that HtS3 uses to configure and deploy an HPCC System to AWS.

Here are the perl scripts divided into three categories: 1) those that copy files to S3 buckets, 2) those that copy files from S3 buckets to THOR nodes, and 3) those that restore logical files once their file parts have been copied to THOR node.

Copy Files From THOR Nodes to S3 Buckets

- cpToS3.pl
    - cpLZAndMetadataFilesToS3.pl
    - cpAllFilePartsToS3.pl

Copy Files From S3 Buckets to THOR Nodes

- cpLZAndMetadataFilesFromS3ToMaster.pl
    - cpLZFilesFromS3ToMaster.pl
    - cpMetadataFilesFromS3ToNode.pl
- cpAllFilePartsFromS3ToThisSlaveNode.pl
- isFilesCopiedFromS3.sh

Restore Logical Files to THOR

- **RestoreLogicalFiles.pl**

Also, covered in this document is the perl script, **common.pl**, which is used by all the above perl scripts and contains common subroutines and the assignment of global variables.

My approach in describing what each script does is to start with the lines of code that do the main function of the script and then discuss what is needed by this code and where it is gotten.

In all these scripts, prints are made to a log file that tells what is being done by each script. These log files exist in the ubuntu home directory (~). We won't discuss these prints in what follows. But, you will notice them in the perl script screenshots (statements such as openLog and printLog).

All copies to/from S3 buckets are done using Amazon’s s3cmd command line utility. S3cmd is installed on all THOR nodes as part of what Juju Charm does when it deploys an HPCC. What tells Juju Charm to have s3cmd installed on all nodes is the addition of the apt-get install for s3cmd to the list of dependencies is the HPCC Charm dependencies directory.

In the following, lines like the following are HtS3 commands entered into a linux terminal window that initiate copies to/from S3 buckets or a Logical File Restore.

```
HtS3 restore
```

Copy Files From THOR Nodes to S3 Buckets

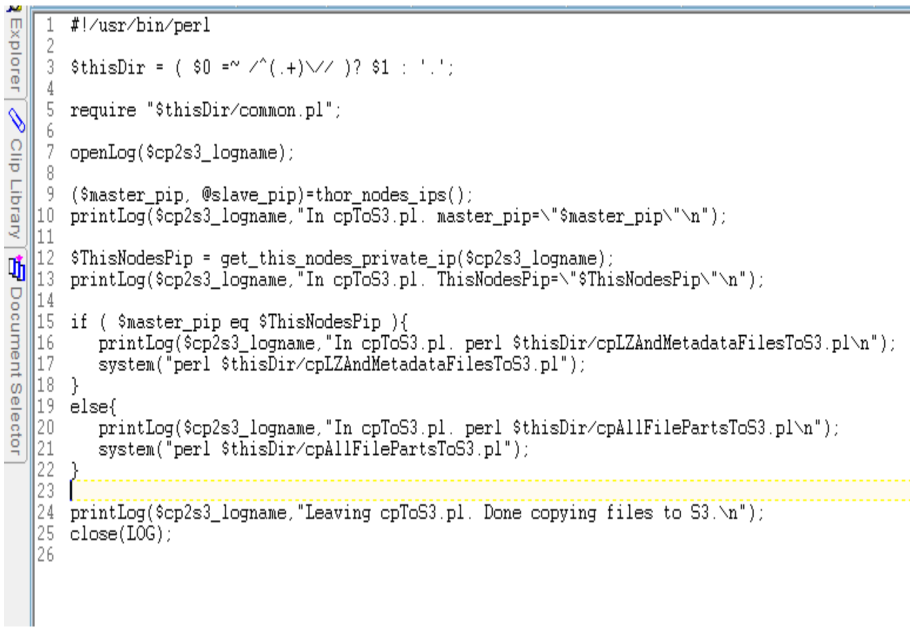
```
HtS3 cp2s3
```

When the HtS3 cp2s3 command is entered at the command prompt of your ubuntu terminal, HtS3 begins the execution of **cpToS3.pl** on each of the THOR nodes. HtS3 does this by issuing an ssh command to each of the THOR nodes with **cpToS3.pl** as the command to execute. HtS3 executes these in the background so it doesn’t have to wait for them to complete and therefore can get **cpToS3.pl** started on all THOR nodes at the same time.

For those of you not familiar with perl code, let me point out a few things about it. First, perl scripts looks somewhat like bash scripts. One difference is that scalar variable names on the left-hand-side of an assignment statement begin with ‘$’ in perl while in bash they don’t.

cpToS3.pl

Let us look at **cpToS3.pl**. The following is a screenshot of it in TextPad.



```
#!/usr/bin/perl

$thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';

require "$thisDir/common.pl";

openLog($cp2s3_logname);

($master_pip, @slave_pip)=thor_nodes_ips();
printLog($cp2s3_logname,"In cpToS3.pl. master_pip=\"$master_pip\"\n");

$ThisNodesPip = get_this_nodes_private_ip($cp2s3_logname);
printLog($cp2s3_logname,"In cpToS3.pl. ThisNodesPip=\"$ThisNodesPip\"\n");

if ( $master_pip eq $ThisNodesPip ){
   printLog($cp2s3_logname,"In cpToS3.pl. perl $thisDir/cpLZAndMetadataFilesToS3.pl\n");
   system("perl $thisDir/cpLZAndMetadataFilesToS3.pl");
}
else{
   printLog($cp2s3_logname,"In cpToS3.pl. perl $thisDir/cpAllFilePartsToS3.pl\n");
   system("perl $thisDir/cpAllFilePartsToS3.pl");
}

printLog($cp2s3_logname,"Leaving cpToS3.pl. Done copying files to S3.\n");
close(LOG);

```

All of these perl scripts use common functions in common.pl (see like 5 above). Plus, all of these perl scripts print information about the processing out to log files that exist in the (~) directory.

The heart of this code is lines 15 through 22. This were the copying begins.

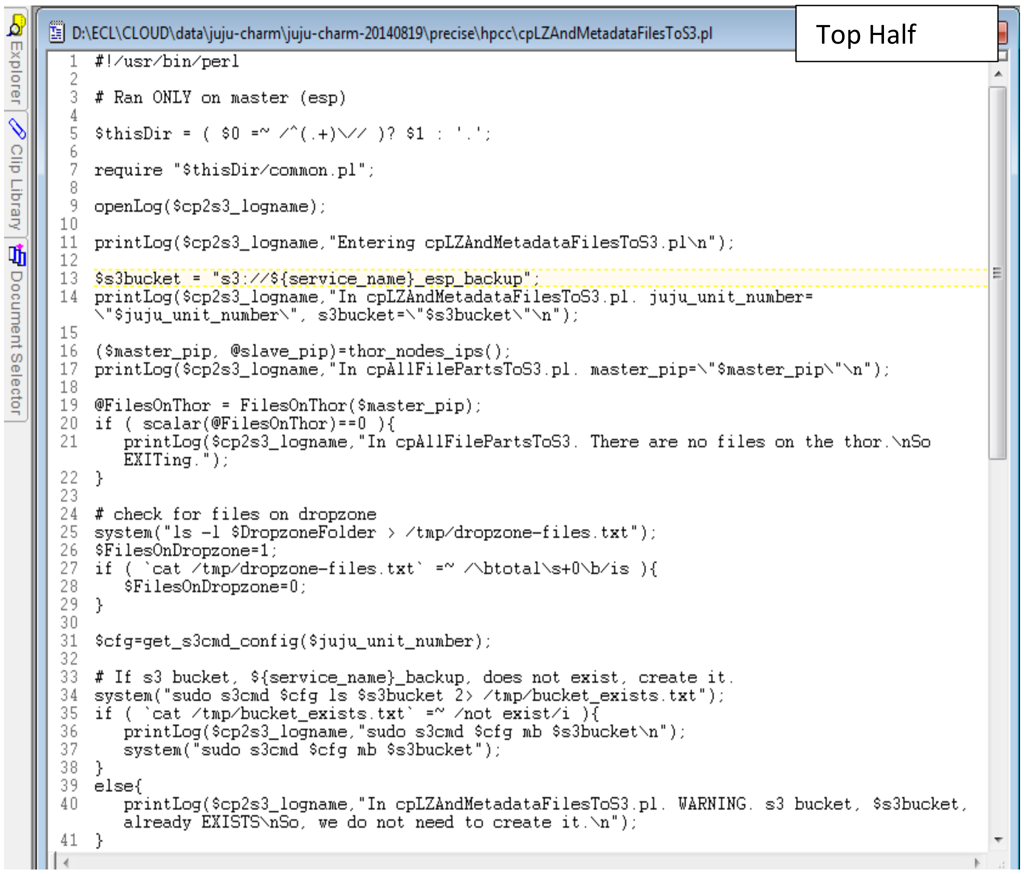
If the THOR node where **cpToS3.pl** is being executed is the master node (or ESP) then **cpLZAndMetadataFilesToS3.pl** is called to copy files on the landing zone (LZ) and metadata files. Metadata files are xml files that describe the relationship of a logical file to its file parts on the various THOR slave nodes.

If the THOR node where **cpToS3.pl** is being executed is NOT the master node (or ESP) then the node must be a slave node. So, **cpAllFilePartsToS3.pl** is called to copy all file parts that exists on the slave node.

Before getting to line 15, we must know the IP of the master node ($master_pip), which we get at line 9, and the IP of the node which **cpToS3.pl** is executing on, ($ThisNodesPip), which we get at line 12.

cpLZAndMetadataFilesToS3.pl

The following are screenshots of **cpLZAndMetadataFilesToS3.pl** in TextPad. The first screenshot is the top half of the file and the 2nd is the bottom half of the file.

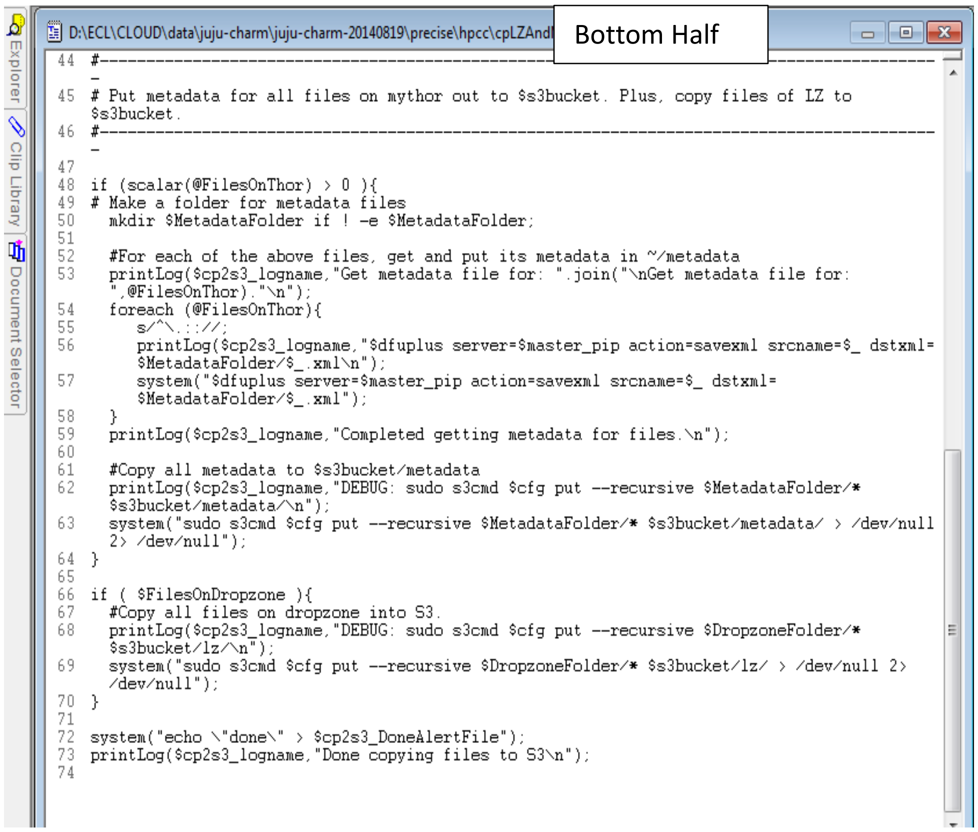


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpLZAndMetadataFilesToS3.pl

```
 1  #!/usr/bin/perl
 2
 3  # Ran ONLY on master (esp)
 4
 5  $thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';
 6
 7  require "$thisDir/common.pl";
 8
 9  openLog($cp2s3_logname);
10
11  printLog($cp2s3_logname,"Entering cpLZAndMetadataFilesToS3.pl\n");
12
13  $s3bucket = "s3://${service_name}_esp_backup";
14  printLog($cp2s3_logname,"In cpLZAndMetadataFilesToS3.pl. juju_unit_number=
    \"$juju_unit_number\", s3bucket=\"$s3bucket\"\n");
15
16  ($master_pip, @slave_pip)=thor_nodes_ips();
17  printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. master_pip=\"$master_pip\"\n");
18
19  @FilesOnThor = FilesOnThor($master_pip);
20  if ( scalar(@FilesOnThor)==0 ){
21     printLog($cp2s3_logname,"In cpAllFilePartsToS3. There are no files on the thor.\nSo
       EXITing.");
22  }
23
24  # check for files on dropzone
25  system("ls -l $DropzoneFolder > /tmp/dropzone-files.txt");
26  $FilesOnDropzone=1;
27  if ( `cat /tmp/dropzone-files.txt` =~ /\btotal\s+0\b/is ){
28     $FilesOnDropzone=0;
29  }
30
31  $cfg=get_s3cmd_config($juju_unit_number);
32
33  # If s3 bucket, ${service_name}_backup, does not exist, create it.
34  system("sudo s3cmd $cfg ls $s3bucket 2> /tmp/bucket_exists.txt");
35  if ( `cat /tmp/bucket_exists.txt` =~ /not exist/i ){
36     printLog($cp2s3_logname,"sudo s3cmd $cfg mb $s3bucket\n");
37     system("sudo s3cmd $cfg mb $s3bucket");
38  }
39  else{
40     printLog($cp2s3_logname,"In cpLZAndMetadataFilesToS3.pl. WARNING. s3 bucket, $s3bucket,
       already EXISTS\nSo, we do not need to create it.\n");
41  }
```

The following table shows what processing is done in the top half that will be used in the bottom as well as where (what lines) it was done.

Line #s	Name	Description
13	$s3bucket	Name of the s3 bucket where THOR files are stored.
16	$master_pip	Private IP of the ESP (master).
19	@FilesOnThor	List of file names on THOR
24-29	$FilesOnDropzone	Boolean indicating whether files on LZ.
33-41		Check if s3 bucket exists. If not create it.
31	$cfg	File of s3 configuration setting.


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpLZAndI
Bottom Half

```
44 #-------------------------------------------------------------------------------------------
   -
45 # Put metadata for all files on mythor out to $s3bucket. Plus, copy files of LZ to
   $s3bucket.
46 #-------------------------------------------------------------------------------------------
   -
47
48 if (scalar(@FilesOnThor) > 0 ){
49 # Make a folder for metadata files
50   mkdir $MetadataFolder if ! -e $MetadataFolder;
51
52   #For each of the above files, get and put its metadata in ~/metadata
53   printLog($cp2s3_logname,"Get metadata file for: ".join("\nGet metadata file for:
     ",@FilesOnThor)."\n");
54   foreach (@FilesOnThor){
55      s/^\.:://;
56      printLog($cp2s3_logname,"$dfuplus server=$master_pip action=savexml srcname=$_ dstxml=
        $MetadataFolder/$_.xml\n");
57      system("$dfuplus server=$master_pip action=savexml srcname=$_ dstxml=
        $MetadataFolder/$_.xml");
58   }
59   printLog($cp2s3_logname,"Completed getting metadata for files.\n");
60
61   #Copy all metadata to $s3bucket/metadata
62   printLog($cp2s3_logname,"DEBUG: sudo s3cmd $cfg put --recursive $MetadataFolder/*
     $s3bucket/metadata/\n");
63   system("sudo s3cmd $cfg put --recursive $MetadataFolder/* $s3bucket/metadata/ > /dev/null
     2> /dev/null");
64 }
65
66 if ( $FilesOnDropzone ){
67   #Copy all files on dropzone into S3.
68   printLog($cp2s3_logname,"DEBUG: sudo s3cmd $cfg put --recursive $DropzoneFolder/*
     $s3bucket/lz/\n");
69   system("sudo s3cmd $cfg put --recursive $DropzoneFolder/* $s3bucket/lz/ > /dev/null 2>
     /dev/null");
70 }
71
72 system("echo \"done\" > $cp2s3_DoneAlertFile");
73 printLog($cp2s3_logname,"Done copying files to S3\n");
74
```

The bottom half is where the heart of the copying occurs. The code in the top half gets information needed by the bottom half.

You will notice two lines (lines 63 and 69) where s3cmd is used. This is where the actual copying from the THOR node to S3 buckets occurs.

Lines 48 through 64 make the metadata files using the HPCC Utility, dfuplus, and copies these files to S3 buckets, line 63; while, lines 66 through 70 copy files on the landing zone (LZ) to S3 buckets.

When, the copying is complete, at line 72, a file is created which is a signal to HtS3 that the copying is done (HtS3 periodically checks for this file and when it sees it, HtS3 knows this node has completed its copying to S3 buckets).

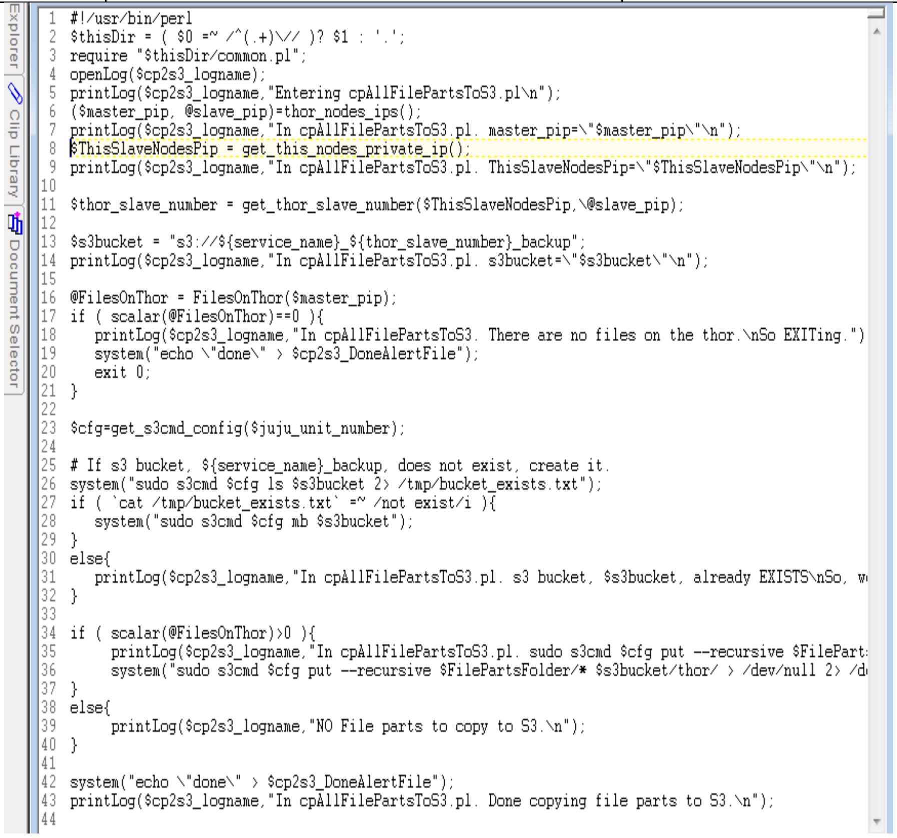
At lines 54 through 58 is a 'foreach' loop which uses dfuplus to create metadata xml files for each logical file (names are in @FilesOnThor) on the deployed THOR. These metadata files are stored in ~/metadata where they are copied to S3 buckets by line 63.

cpAllFilePartsToS3.pl

The following screenshot is of **cpAllFilePartsToS3.pl** in TextPad. File parts are copied at lines 34 through 40. The code that actually does the copying is line 36.

The following table shows processing needed by the copy process that occurred before line 34.

Line #s	Name	Description
6	$master_pip & @slave_pip	Private IPs for the master (ESP) and all slave nodes.
8	$ThisSlaveNodesPip	This slave node's private IP
11	$thor_slave_number	Node number (as seen in ECL Watch) of this slave node.
13	$s3bucket	Name of the s3 bucket where THOR files are stored.
16	@FilesOnThor	List of file names on THOR
23	$cfg	File of s3 configuaration settings.
26-32		Check if s3 bucket exists. If not create it.



```
 1  #!/usr/bin/perl
 2  $thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';
 3  require "$thisDir/common.pl";
 4  openLog($cp2s3_logname);
 5  printLog($cp2s3_logname,"Entering cpAllFilePartsToS3.pl\n");
 6  ($master_pip, @slave_pip)=thor_nodes_ips();
 7  printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. master_pip=\"$master_pip\"\n");
 8  $ThisSlaveNodesPip = get_this_nodes_private_ip();
 9  printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. ThisSlaveNodesPip=\"$ThisSlaveNodesPip\"\n");
10
11  $thor_slave_number = get_thor_slave_number($ThisSlaveNodesPip,\@slave_pip);
12
13  $s3bucket = "s3://${service_name}_${thor_slave_number}_backup";
14  printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. s3bucket=\"$s3bucket\"\n");
15
16  @FilesOnThor = FilesOnThor($master_pip);
17  if ( scalar(@FilesOnThor)==0 ){
18     printLog($cp2s3_logname,"In cpAllFilePartsToS3. There are no files on the thor.\nSo EXITing.")
19     system("echo \"done\" > $cp2s3_DoneAlertFile");
20     exit 0;
21  }
22
23  $cfg=get_s3cmd_config($juju_unit_number);
24
25  # If s3 bucket, ${service_name}_backup, does not exist, create it.
26  system("sudo s3cmd $cfg ls $s3bucket 2> /tmp/bucket_exists.txt");
27  if ( `cat /tmp/bucket_exists.txt` =~ /not exist/i ){
28     system("sudo s3cmd $cfg mb $s3bucket");
29  }
30  else{
31     printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. s3 bucket, $s3bucket, already EXISTS\nSo, w
32  }
33
34  if ( scalar(@FilesOnThor)>0 ){
35      printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. sudo s3cmd $cfg put --recursive $FilePart
36      system("sudo s3cmd $cfg put --recursive $FilePartsFolder/* $s3bucket/thor/ > /dev/null 2> /d
37  }
38  else{
39      printLog($cp2s3_logname,"NO File parts to copy to S3.\n");
40  }
41
42  system("echo \"done\" > $cp2s3_DoneAlertFile");
43  printLog($cp2s3_logname,"In cpAllFilePartsToS3.pl. Done copying file parts to S3.\n");
44
```

The last thing that **cpAllFilePartsToS3.pl** does, at line 42, is create a file which is a signal to HtS3 that the copying is done (HtS3 periodically checks for this file and when it sees it, HtS3 knows this node has completed its copying to S3 buckets).

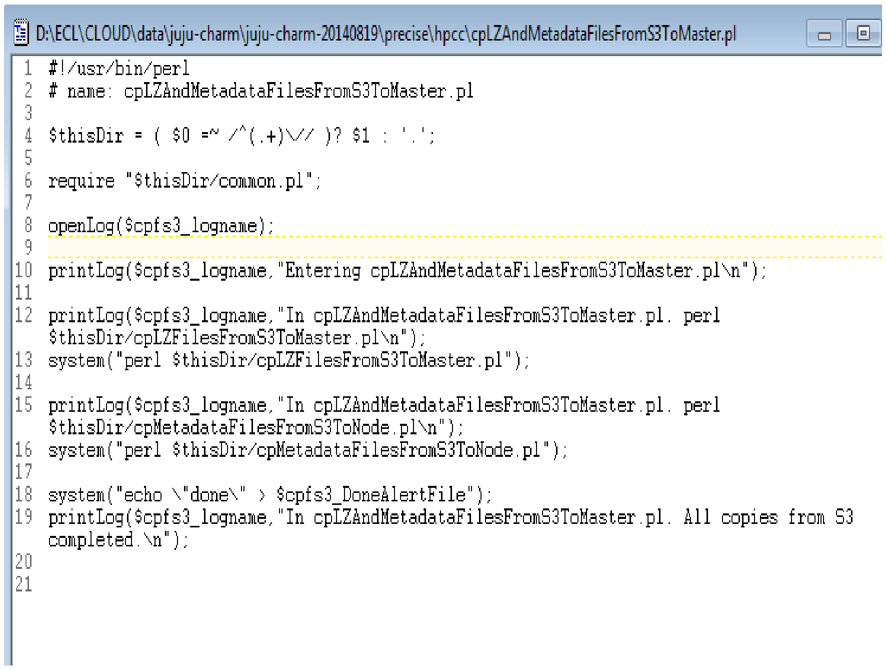
Copy Files From S3 Buckets to THOR Nodes

```
HtS3 cpfs3
```

When the HtS3 cpfs3 command is entered at the command prompt of your ubuntu terminal, HtS3 begins the execution of **cpLZAndMetadataFilesFromS3ToMaster.pl** on the master (ESP) THOR node, using ssh, to copy Landing Zone files and metadata files from S3 buckets to the master node. Once this is started, HtS3 begins the execution of **cpAllFilePartsFromS3ToThisSlaveNode.pl** on each of the THOR slave nodes, using ssh, to copy file parts from S3 buckets to each of the slave nodes.

cpLZAndMetadataFilesFromS3ToMaster.pl

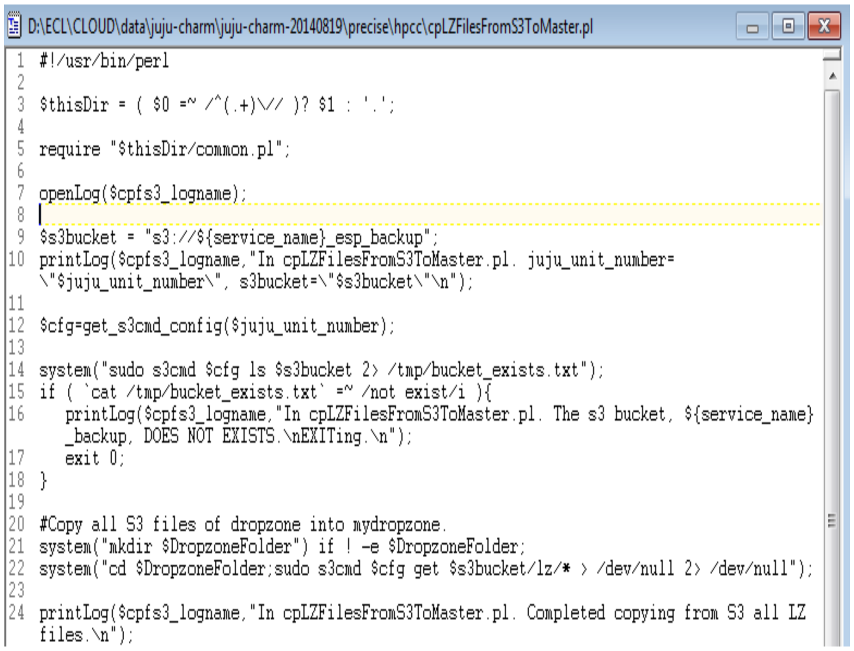
Below is a screenshot of **cpLZAndMetadataFilesFromS3ToMaster.pl** in Textpad. The copying is done by lines 13 and 16 where **cpLZFilesFromS3ToMaster.pl** and **cpMetadataFilesFromS3ToNode.pl** are called to copy from S3 buckets Landing zone files and metadata files, respectively.


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpLZAndMetadataFilesFromS3ToMaster.pl

```
 1  #!/usr/bin/perl
 2  # name: cpLZAndMetadataFilesFromS3ToMaster.pl
 3
 4  $thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';
 5
 6  require "$thisDir/common.pl";
 7
 8  openLog($cpfs3_logname);
 9
10  printLog($cpfs3_logname,"Entering cpLZAndMetadataFilesFromS3ToMaster.pl\n");
11
12  printLog($cpfs3_logname,"In cpLZAndMetadataFilesFromS3ToMaster.pl. perl
    $thisDir/cpLZFilesFromS3ToMaster.pl\n");
13  system("perl $thisDir/cpLZFilesFromS3ToMaster.pl");
14
15  printLog($cpfs3_logname,"In cpLZAndMetadataFilesFromS3ToMaster.pl. perl
    $thisDir/cpMetadataFilesFromS3ToNode.pl\n");
16  system("perl $thisDir/cpMetadataFilesFromS3ToNode.pl");
17
18  system("echo \"done\" > $cpfs3_DoneAlertFile");
19  printLog($cpfs3_logname,"In cpLZAndMetadataFilesFromS3ToMaster.pl. All copies from S3
    completed.\n");
20
21
```

cpLZFilesFromS3ToMaster.pl

Below is a screenshot of **cpLZFilesFromS3ToMaster.pl** in TextPad. Line 22 is where the copy, by s3cmd, of landing zone files from S3 buckets to the master (ESP) node occurs. Before, the copy can occur, at lines 14 through 18, the code checks to see if the S3 bucket, $s3bucket, exists. If it does not exist, the code prints to the log file and exits. Other information the copy needs: $s3bucket it gets at line 9, $cfg it gets from line 12. Also, $DropzoneFolder (synonymous with landing zone) is needed by the copy is defined in **common.pl** called at line 5.



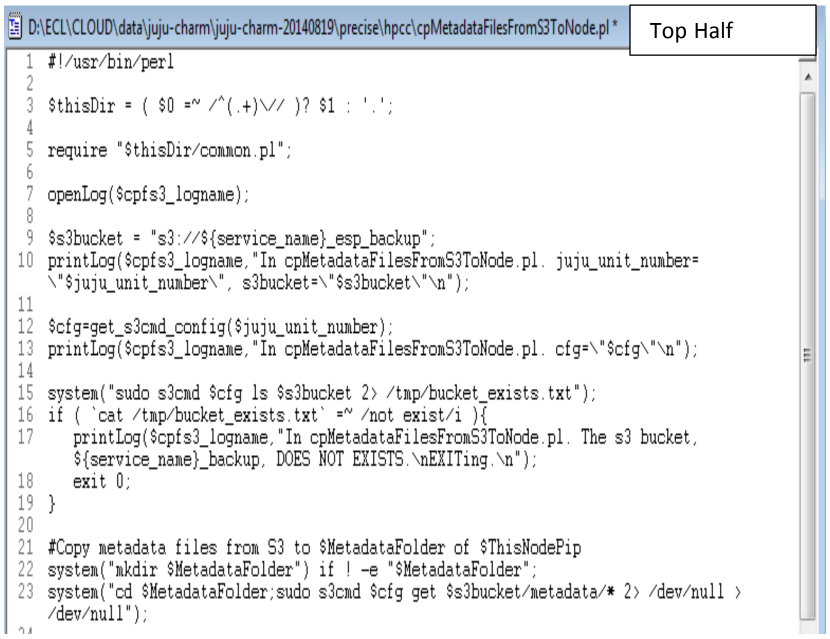
D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpLZFilesFromS3ToMaster.pl

```
#!/usr/bin/perl

$thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';

require "$thisDir/common.pl";

openLog($cpfs3_logname);

$s3bucket = "s3://${service_name}_esp_backup";
printLog($cpfs3_logname,"In cpLZFilesFromS3ToMaster.pl. juju_unit_number=
\"$juju_unit_number\", s3bucket=\"$s3bucket\"\n");

$cfg=get_s3cmd_config($juju_unit_number);

system("sudo s3cmd $cfg ls $s3bucket 2> /tmp/bucket_exists.txt");
if ( `cat /tmp/bucket_exists.txt` =~ /not exist/i ){
   printLog($cpfs3_logname,"In cpLZFilesFromS3ToMaster.pl. The s3 bucket, ${service_name}
   _backup, DOES NOT EXISTS.\nEXITing.\n");
   exit 0;
}

#Copy all S3 files of dropzone into mydropzone.
system("mkdir $DropzoneFolder") if ! -e $DropzoneFolder;
system("cd $DropzoneFolder;sudo s3cmd $cfg get $s3bucket/lz/* > /dev/null 2> /dev/null");

printLog($cpfs3_logname,"In cpLZFilesFromS3ToMaster.pl. Completed copying from S3 all LZ
files.\n");
```

cpMetadataFilesFromS3ToNode.pl

Below are two screenshots of **cpMetadataFilesFromS3ToNode.pl** in TextPad: one for the top half of **cpMetadataFilesFromS3ToNode.pl** and one for the bottom half. The copy of metadata files from s3 buckets occurs at the top and in the bottom half the private IP addresses in these metadata files are changed so they agree with those of this THOR (Currently these files contain the private IP addresses of the THOR from which they were copied to S3 buckets).

The copy, using s3cmd, occurs at line 23 in the Top Half. Also in the Top Half, the code checks to see if the S3 bucket exists. If it does not then a print to the log file records this fact followed an exit. In addition, in the Top Half, $s3bucket is defined at line 9 and $cfg is defined at line 12.

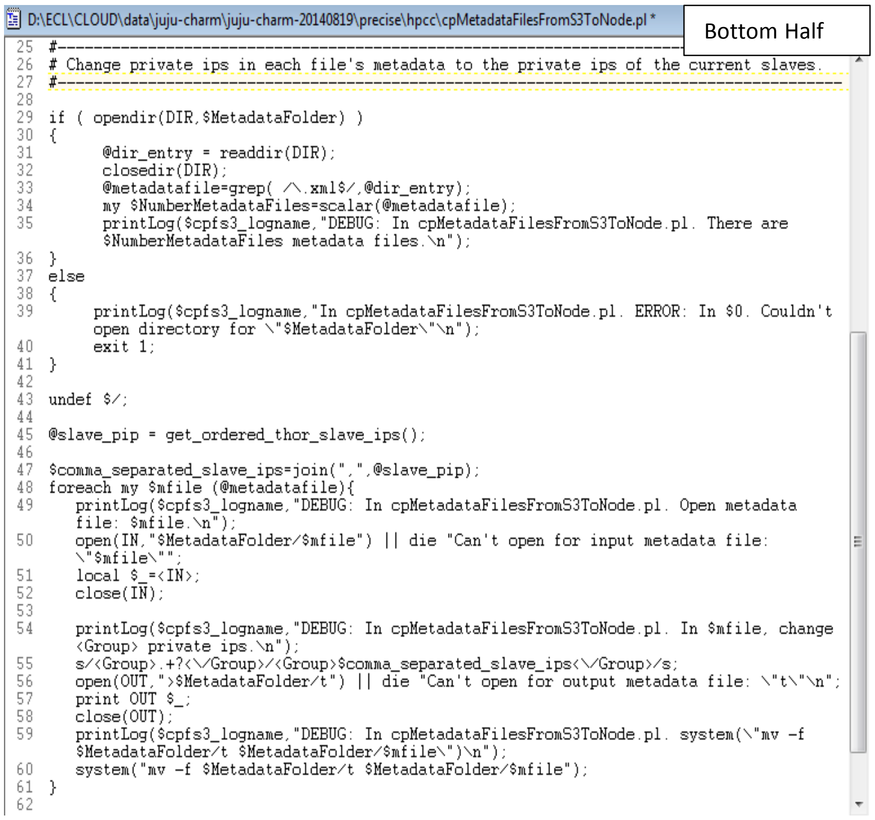

D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpMetadataFilesFromS3ToNode.pl *

Top Half

```perl
#!/usr/bin/perl

$thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';

require "$thisDir/common.pl";

openLog($cpfs3_logname);

$s3bucket = "s3://${service_name}_esp_backup";
printLog($cpfs3_logname,"In cpMetadataFilesFromS3ToNode.pl. juju_unit_number=
\"$juju_unit_number\", s3bucket=\"$s3bucket\"\n");

$cfg=get_s3cmd_config($juju_unit_number);
printLog($cpfs3_logname,"In cpMetadataFilesFromS3ToNode.pl. cfg=\"$cfg\"\n");

system("sudo s3cmd $cfg ls $s3bucket 2> /tmp/bucket_exists.txt");
if ( `cat /tmp/bucket_exists.txt` =~ /not exist/i ){
   printLog($cpfs3_logname,"In cpMetadataFilesFromS3ToNode.pl. The s3 bucket,
   ${service_name}_backup, DOES NOT EXISTS.\nEXITing.\n");
   exit 0;
}

#Copy metadata files from S3 to $MetadataFolder of $ThisNodePip
system("mkdir $MetadataFolder") if ! -e "$MetadataFolder";
system("cd $MetadataFolder;sudo s3cmd $cfg get $s3bucket/metadata/* 2> /dev/null >
/dev/null");
```

In the Bottom Half the private IP addresses of the metadata files are changed to those of the current THOR nodes. At lines 29 through 41 the names of the metadata files are placed in @metadatafile and the number of them is placed in $NumberMetadataFiles.

At line 45 we get the IP addresses of all slave nodes of the current THOR. And, at lines 47 through 61 we put the IP addresses of the slave nodes into the metadata files.


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\cpMetadataFilesFromS3ToNode.pl *
Bottom Half

```
25 #-------------------------------------------------------------------------
26 # Change private ips in each file's metadata to the private ips of the current slaves.
27 #-------------------------------------------------------------------------
28
29 if ( opendir(DIR,$MetadataFolder) )
30 {
31       @dir_entry = readdir(DIR);
32       closedir(DIR);
33       @metadatafile=grep( /\.xml$/,@dir_entry);
34       my $NumberMetadataFiles=scalar(@metadatafile);
35       printLog($cpfs3_logname,"DEBUG: In cpMetadataFilesFromS3ToNode.pl. There are
         $NumberMetadataFiles metadata files.\n");
36 }
37 else
38 {
39      printLog($cpfs3_logname,"In cpMetadataFilesFromS3ToNode.pl. ERROR: In $0. Couldn't
        open directory for \"$MetadataFolder\"\n");
40      exit 1;
41 }
42
43 undef $/;
44
45 @slave_pip = get_ordered_thor_slave_ips();
46
47 $comma_separated_slave_ips=join(",",@slave_pip);
48 foreach my $mfile (@metadatafile){
49    printLog($cpfs3_logname,"DEBUG: In cpMetadataFilesFromS3ToNode.pl. Open metadata
      file: $mfile.\n");
50    open(IN,"$MetadataFolder/$mfile") || die "Can't open for input metadata file:
      \"$mfile\"";
51    local $_=<IN>;
52    close(IN);
53
54    printLog($cpfs3_logname,"DEBUG: In cpMetadataFilesFromS3ToNode.pl. In $mfile, change
      <Group> private ips.\n");
55    s/<Group>.+?<\/Group>/<Group>$comma_separated_slave_ips<\/Group>/s;
56    open(OUT,">$MetadataFolder/t") || die "Can't open for output metadata file: \"t\"\n";
57    print OUT $_;
58    close(OUT);
59    printLog($cpfs3_logname,"DEBUG: In cpMetadataFilesFromS3ToNode.pl. system(\"mv -f
      $MetadataFolder/t $MetadataFolder/$mfile\")\n");
60    system("mv -f $MetadataFolder/t $MetadataFolder/$mfile");
61 }
62
```

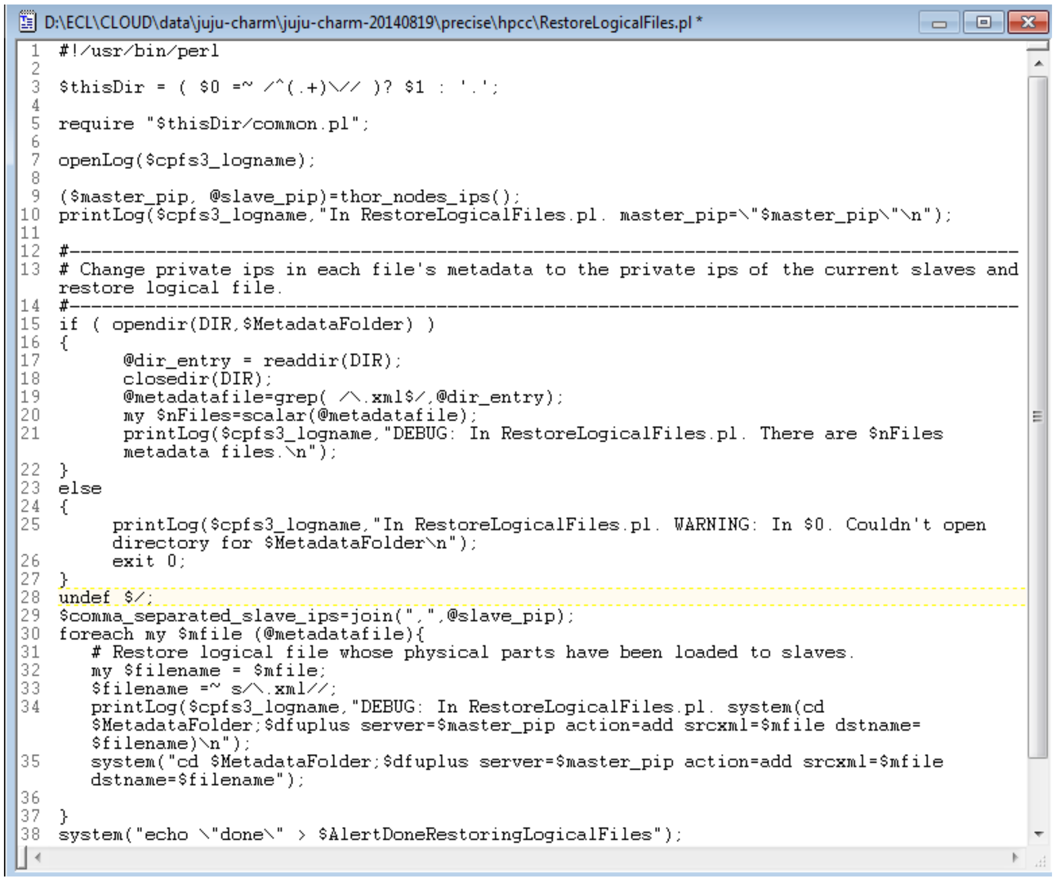
<u>Restore Logical Files to THOR</u>

HtS3 restore

RestoreLogicalFiles.pl

The following is a screenshot of **RestoreLogicalFiles.pl** in TextPad. Line35 of this screenshot is where dfuplus is used to tell the deployed THOR that all file parts of a given logical file has been copied to their respective slave nodes and therefore the logical file should be added to the THOR. The foreach loop that restores all logical files begins at line 30 and ends at line 37. This is the main function of this script.

The above mentioned foreach loop needs: 1) @metadatafile, the list of metadata files for all files to be stored, which is filled at line 19 (these metadata files can be found in the directory, $MetadataFolder, which is defined in common.pl (see line 5)); 2) $master_pip, which is assigned at line 9.


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\RestoreLogicalFiles.pl *

```
#!/usr/bin/perl

$thisDir = ( $0 =~ /^(.+)\// )? $1 : '.';

require "$thisDir/common.pl";

openLog($cpfs3_logname);

($master_pip, @slave_pip)=thor_nodes_ips();
printLog($cpfs3_logname,"In RestoreLogicalFiles.pl. master_pip=\"$master_pip\"\n");

#-----------------------------------------------------------------------------------------
# Change private ips in each file's metadata to the private ips of the current slaves and
restore logical file.
#-----------------------------------------------------------------------------------------
if ( opendir(DIR,$MetadataFolder) )
{
      @dir_entry = readdir(DIR);
      closedir(DIR);
      @metadatafile=grep( /\.xml$/,@dir_entry);
      my $nFiles=scalar(@metadatafile);
      printLog($cpfs3_logname,"DEBUG: In RestoreLogicalFiles.pl. There are $nFiles
      metadata files.\n");
}
else
{
     printLog($cpfs3_logname,"In RestoreLogicalFiles.pl. WARNING: In $0. Couldn't open
     directory for $MetadataFolder\n");
     exit 0;
}
undef $/;
$comma_separated_slave_ips=join(",",@slave_pip);
foreach my $mfile (@metadatafile){
   # Restore logical file whose physical parts have been loaded to slaves.
   my $filename = $mfile;
   $filename =~ s/\.xml//;
   printLog($cpfs3_logname,"DEBUG: In RestoreLogicalFiles.pl. system(cd
   $MetadataFolder;$dfuplus server=$master_pip action=add srcxml=$mfile dstname=
   $filename)\n");
   system("cd $MetadataFolder;$dfuplus server=$master_pip action=add srcxml=$mfile
   dstname=$filename");

}
system("echo \"done\" > $AlertDoneRestoringLogicalFiles");
```

Common.pl

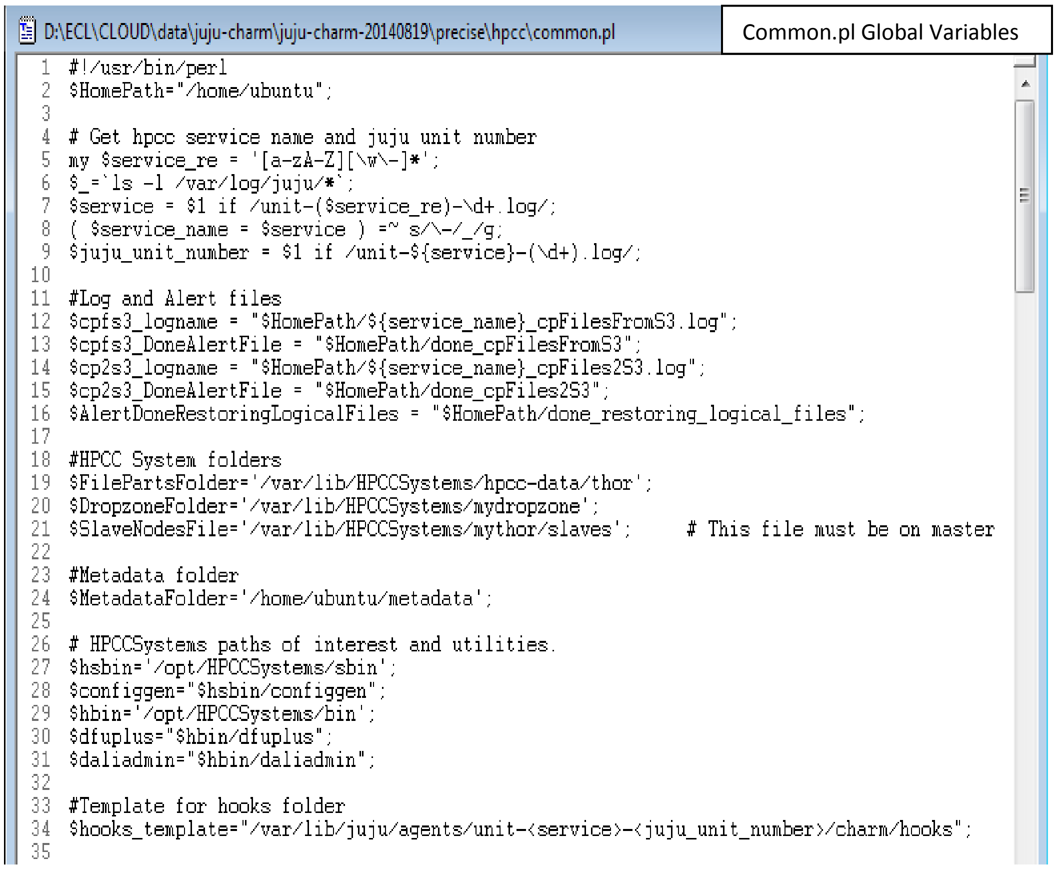
Below, there are 5 screenshots of different sections of **common.pl**. The first screenshot shows the assignment of global variables which is at the beginning of **common.pl**. The other four screenshots are of common subroutines of **common.pl**.

Global Variables Assigned in Common.pl

The following screenshot shows the lines of code of **common.pl** where global variables are defined. Also, the following table gives a brief description of these global variables.

Table of Global Variables Defined in common.pl		
Global Variable	Line Defined	Description
$HomePath	2	Ubuntu's home path on any THOR node.
$service	7	Name of service as given in 'juju deploy' command.
$juju_unit_number	9	Juju assigned number for this instance.
Log And Alert Files		
$cpfs3_logname	12	Log file name for copy from S3 buckets.
$cpfs3_DoneAlertFile	13	Signal done alert file name for copy from S3 buckets.
$cp2s3_logname	14	Log file name for copy to S3 buckets.

Table of Global Variables Defined in common.pl		
$cp2s3_DoneAlertFile	15	Signal done alert file name for copy to S3 buckets.
$AlertDoneRestoringLogicalFile	16	Signal all files restored.
HPCC System Folders		
$FilePartsFolder	19	Folder where files parts exists or will be copied.
$DropzoneFolder	20	Dropzone (or landing zone) folder.
$SlaveNodesFile	21	NOT USED
$MetadataFolder	24	Folder where metadata files are stored.
Other HPCC Paths		
$hsbin	27	HPCC sbin folder
$configgen	28	Path of HPCC configgen tool.
$hbin	29	HPCC bin folder
$dfuplus	30	Path of HPCC dfuplus tool.
$daliadmin	31	Path of HPCC daliadmin tool.

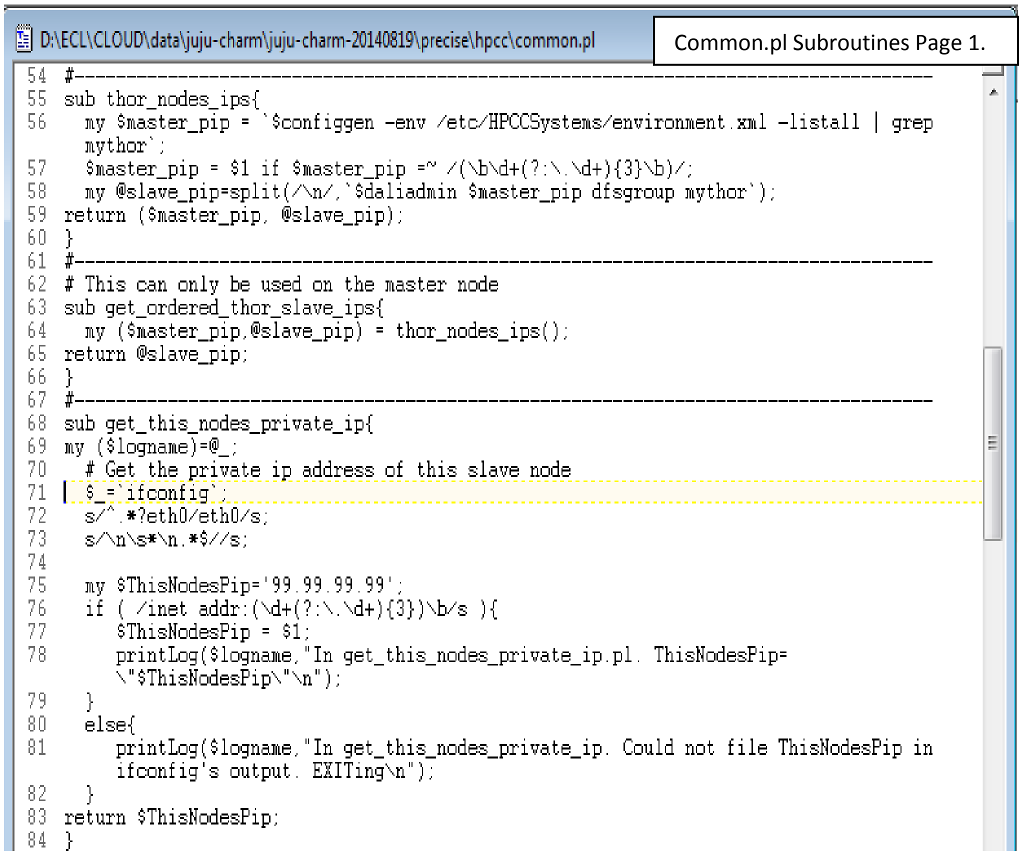

D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\common.pl
Common.pl Global Variables

```
 1  #!/usr/bin/perl
 2  $HomePath="/home/ubuntu";
 3
 4  # Get hpcc service name and juju unit number
 5  my $service_re = '[a-zA-Z][\w\-]*';
 6  $_=`ls -l /var/log/juju/*`;
 7  $service = $1 if /unit-($service_re)-\d+.log/;
 8  ( $service_name = $service ) =~ s/\-/_/g;
 9  $juju_unit_number = $1 if /unit-${service}-(\d+).log/;
10
11  #Log and Alert files
12  $cpfs3_logname = "$HomePath/${service_name}_cpFilesFromS3.log";
13  $cpfs3_DoneAlertFile = "$HomePath/done_cpFilesFromS3";
14  $cp2s3_logname = "$HomePath/${service_name}_cpFiles2S3.log";
15  $cp2s3_DoneAlertFile = "$HomePath/done_cpFiles2S3";
16  $AlertDoneRestoringLogicalFiles = "$HomePath/done_restoring_logical_files";
17
18  #HPCC System folders
19  $FilePartsFolder='/var/lib/HPCCSystems/hpcc-data/thor';
20  $DropzoneFolder='/var/lib/HPCCSystems/mydropzone';
21  $SlaveNodesFile='/var/lib/HPCCSystems/mythor/slaves';     # This file must be on master
22
23  #Metadata folder
24  $MetadataFolder='/home/ubuntu/metadata';
25
26  # HPCCSystems paths of interest and utilities.
27  $hsbin='/opt/HPCCSystems/sbin';
28  $configgen="$hsbin/configgen";
29  $hbin='/opt/HPCCSystems/bin';
30  $dfuplus="$hbin/dfuplus";
31  $daliadmin="$hbin/daliadmin";
32
33  #Template for hooks folder
34  $hooks_template="/var/lib/juju/agents/unit-<service>-<juju_unit_number>/charm/hooks";
35
```

There are four screenshots that follow that shows all subroutines in **common.pl**. Also below, is a table the gives a short description of each subroutine.

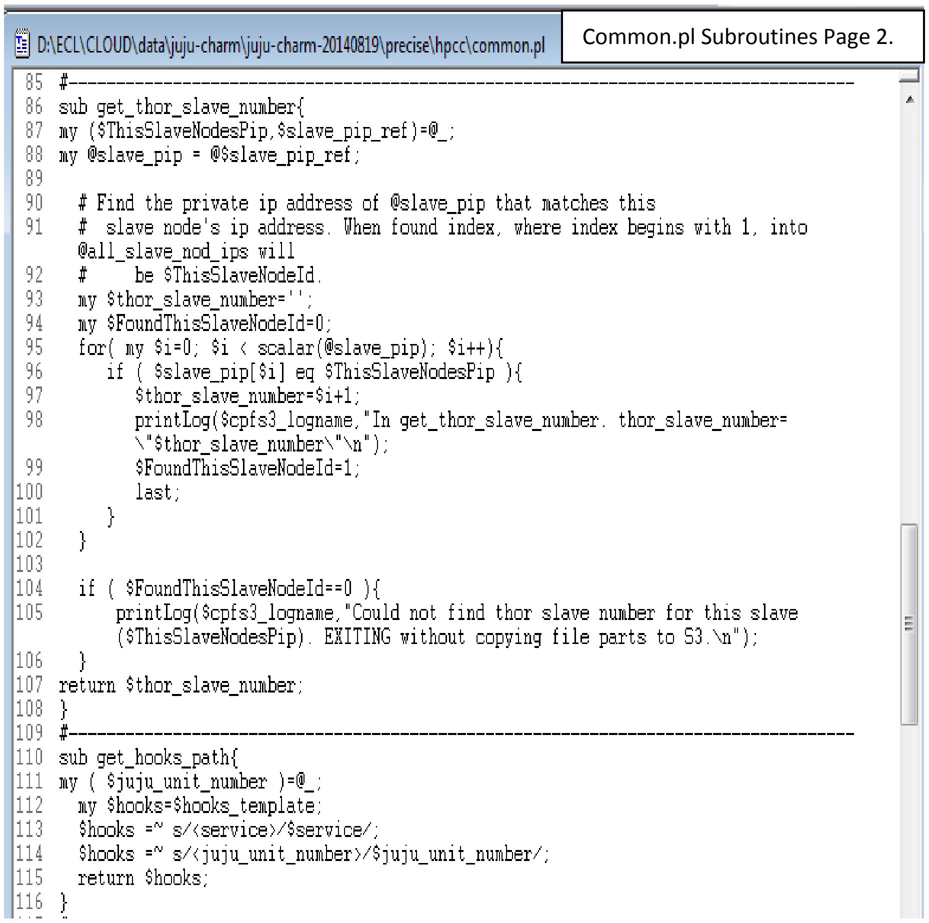
Subroutines In common.pl		
Subroutine Name	**Lines #s**	**Description**
thor_nodes_ips	55-60	Returns $master_pip & @slave_pip.

Subroutines In common.pl		
Subroutine Name	**Lines #s**	**Description**
get_ordered_thor_slave_nodes	63-66	Returns, @slave_pip, a list of slave private IPs. Returned in ascending order of THOR node number.
get_this_nodes_private_ip	68-84	From the output of 'ifconfig' exacts this node's private IP address.
get_thor_slave_number	85-108	Returns the THOR node number which it gets from the ordered list of slave IPs.
get_hooks_path	110-116	Returns the full path to the HPCC Charms hooks folder.
get_s3cmd_config	118-132	Get the configuration file of settings.
FilesOnThor	134-143	Uses dfuplus to get a list of all files on THOR.
cpAllFilePartsOnS3	145-188	Copies all file parts in S3 bucket to slave node. It places each file part in the same folder as the S3 bucket has it.


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\common.pl

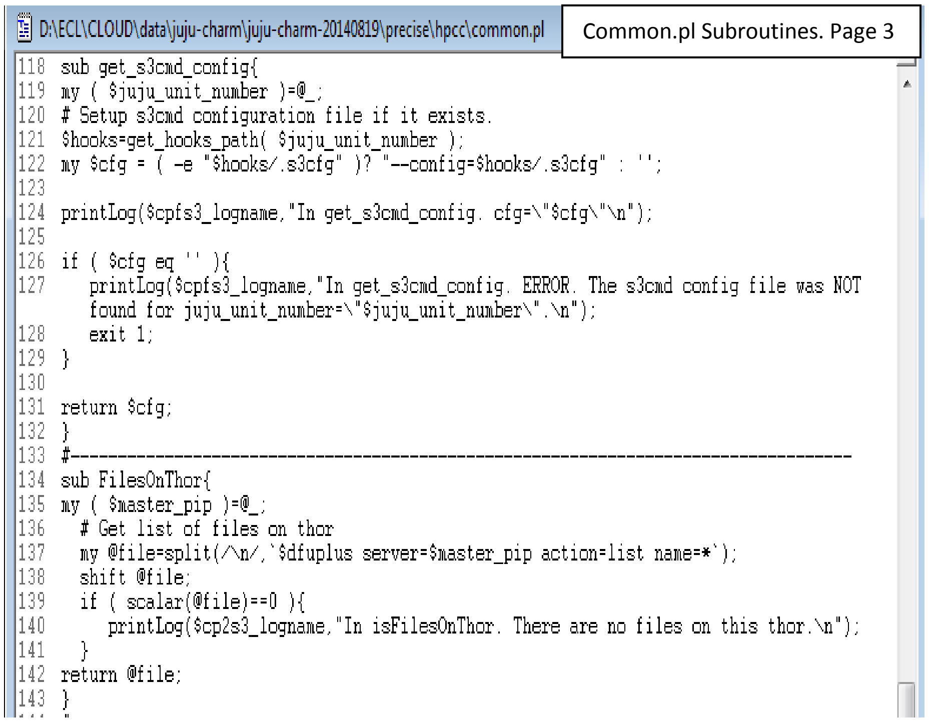
Common.pl Subroutines Page 1.

```
54 #------------------------------------------------------------------------------------
55 sub thor_nodes_ips{
56   my $master_pip = `$configgen -env /etc/HPCCSystems/environment.xml -listall | grep
     mythor`;
57   $master_pip = $1 if $master_pip =~ /(\b\d+(?:\.\d+){3}\b)/;
58   my @slave_pip=split(/\n/,`$daliadmin $master_pip dfsgroup mythor`);
59 return ($master_pip, @slave_pip);
60 }
61 #------------------------------------------------------------------------------------
62 # This can only be used on the master node
63 sub get_ordered_thor_slave_ips{
64   my ($master_pip,@slave_pip) = thor_nodes_ips();
65 return @slave_pip;
66 }
67 #------------------------------------------------------------------------------------
68 sub get_this_nodes_private_ip{
69 my ($logname)=@_;
70   # Get the private ip address of this slave node
71   $_=`ifconfig`;
72   s/^.*?eth0/eth0/s;
73   s/\n\s*\n.*$//s;
74
75   my $ThisNodesPip='99.99.99.99';
76   if ( /inet addr:(\d+(?:\.\d+){3})\b/s ){
77      $ThisNodesPip = $1;
78      printLog($logname,"In get_this_nodes_private_ip.pl. ThisNodesPip=
        \"$ThisNodesPip\"\n");
79   }
80   else{
81      printLog($logname,"In get_this_nodes_private_ip. Could not file ThisNodesPip in
        ifconfig's output. EXITing\n");
82   }
83 return $ThisNodesPip;
84 }
```


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\common.pl

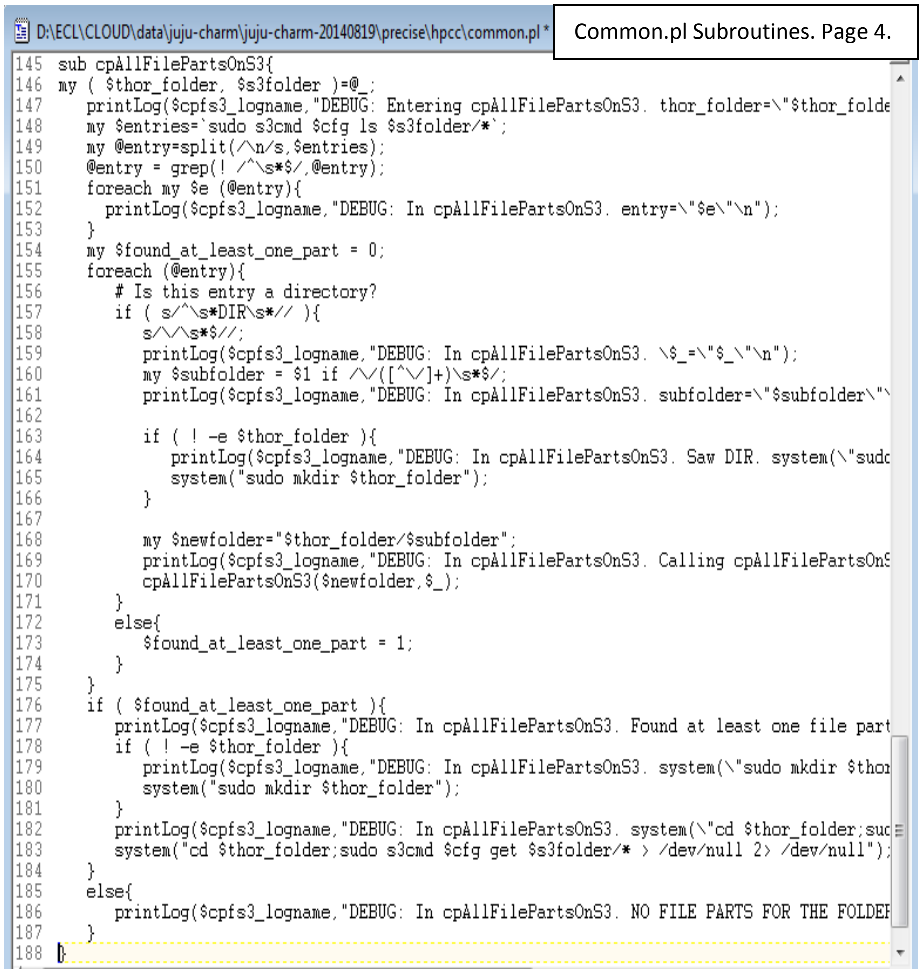
Common.pl Subroutines Page 2.

```perl
#-------------------------------------------------------------------------------------
sub get_thor_slave_number{
my ($ThisSlaveNodesPip,$slave_pip_ref)=@_;
my @slave_pip = @$slave_pip_ref;

  # Find the private ip address of @slave_pip that matches this
  #  slave node's ip address. When found index, where index begins with 1, into
  @all_slave_nod_ips will
  #      be $ThisSlaveNodeId.
  my $thor_slave_number='';
  my $FoundThisSlaveNodeId=0;
  for( my $i=0; $i < scalar(@slave_pip); $i++){
     if ( $slave_pip[$i] eq $ThisSlaveNodesPip ){
        $thor_slave_number=$i+1;
        printLog($cpfs3_logname,"In get_thor_slave_number. thor_slave_number=
        \"$thor_slave_number\"\n");
        $FoundThisSlaveNodeId=1;
        last;
     }
  }

  if ( $FoundThisSlaveNodeId==0 ){
      printLog($cpfs3_logname,"Could not find thor slave number for this slave
      ($ThisSlaveNodesPip). EXITING without copying file parts to S3.\n");
  }
return $thor_slave_number;
}
#-------------------------------------------------------------------------------------
sub get_hooks_path{
my ( $juju_unit_number )=@_;
  my $hooks=$hooks_template;
  $hooks =~ s/<service>/$service/;
  $hooks =~ s/<juju_unit_number>/$juju_unit_number/;
  return $hooks;
}
```


D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\common.pl

Common.pl Subroutines. Page 3

```perl
sub get_s3cmd_config{
my ( $juju_unit_number )=@_;
# Setup s3cmd configuration file if it exists.
$hooks=get_hooks_path( $juju_unit_number );
my $cfg = ( -e "$hooks/.s3cfg" )? "--config=$hooks/.s3cfg" : '';

printLog($cpfs3_logname,"In get_s3cmd_config. cfg=\"$cfg\"\n");

if ( $cfg eq '' ){
   printLog($cpfs3_logname,"In get_s3cmd_config. ERROR. The s3cmd config file was NOT
   found for juju_unit_number=\"$juju_unit_number\".\n");
   exit 1;
}

return $cfg;
}
#-------------------------------------------------------------------------------------
sub FilesOnThor{
my ( $master_pip )=@_;
  # Get list of files on thor
  my @file=split(/\n/,`$dfuplus server=$master_pip action=list name=*`);
  shift @file;
  if ( scalar(@file)==0 ){
     printLog($cp2s3_logname,"In isFilesOnThor. There are no files on this thor.\n");
  }
return @file;
}
```



D:\ECL\CLOUD\data\juju-charm\juju-charm-20140819\precise\hpcc\common.pl *

Common.pl Subroutines. Page 4.

```perl
sub cpAllFilePartsOnS3{
my ( $thor_folder, $s3folder )=@_;
   printLog($cpfs3_logname,"DEBUG: Entering cpAllFilePartsOnS3. thor_folder=\"$thor_folde
   my $entries=`sudo s3cmd $cfg ls $s3folder/*`;
   my @entry=split(/\n/s,$entries);
   @entry = grep(! /^\s*$/,@entry);
   foreach my $e (@entry){
     printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. entry=\"$e\"\n");
   }
   my $found_at_least_one_part = 0;
   foreach (@entry){
      # Is this entry a directory?
      if ( s/^\s*DIR\s*// ){
         s/\/\s*$//;
         printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. \$_=\"$_\"\n");
         my $subfolder = $1 if /\/([^\/]+)\s*$/;
         printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. subfolder=\"$subfolder\"\

         if ( ! -e $thor_folder ){
            printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. Saw DIR. system(\"sudo
            system("sudo mkdir $thor_folder");
         }

         my $newfolder="$thor_folder/$subfolder";
         printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. Calling cpAllFilePartsOnS
         cpAllFilePartsOnS3($newfolder,$_);
      }
      else{
         $found_at_least_one_part = 1;
      }
   }
   if ( $found_at_least_one_part ){
      printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. Found at least one file part
      if ( ! -e $thor_folder ){
         printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. system(\"sudo mkdir $thor
         system("sudo mkdir $thor_folder");
      }
      printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. system(\"cd $thor_folder;suc
      system("cd $thor_folder;sudo s3cmd $cfg get $s3folder/* > /dev/null 2> /dev/null");
   }
   else{
      printLog($cpfs3_logname,"DEBUG: In cpAllFilePartsOnS3. NO FILE PARTS FOR THE FOLDEF
   }
}
```