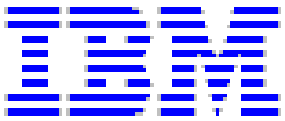

Whitepaper

AMM CLI Example Scripts

Revision Level: 1.0
Last Revised: October 9, 2008

Software License Agreement

This Agreement is between International Business Machines Corporation (“Licensor”) and you (“Licensee”), under which Licensee will obtain from Licensor a copy of the sample code for BladeCenter Advanced Management Module Command Line Interface (the “Software”), which can be found below. By acquiring and/or copying the Software, Licensee agrees to the terms and provisions of this Agreement.

1.0 LICENSE
Licensor grants to Licensee a worldwide, royalty-free, fully paid-up, nonexclusive, nontransferable, object code license to copy, sublicense, redistribute and create derivative works of the Software. Licensor is under no obligation to provide updates.

2.0 TERMINATION
This Agreement terminates automatically upon any breach of this Agreement, or at any time with written notice from Licensor. Upon termination of this Agreement, Licensee will destroy all copies of the Software. Provisions of this Agreement that by their nature extend beyond termination or expiration will survive in accordance with their terms.

3.0 LIMITATION OF LIABILITY
THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF NONINFRINGEMENT OR THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. In no event will Licensor be liable to Licensee for any lost revenues, lost profits, incidental, direct, indirect, consequential, special or punitive damages. Licensor is under no obligation to provide support for the Software to Licensee or any end user.

4.0 GENERAL PROVISIONS
Neither party relies on any representations made by the other, except as expressly provided in this Agreement. Failure to exercise a right does not constitute a waiver under this Agreement. Neither party will bring a legal action relating to the subject matter of this Agreement, against the other, more than 2 years after the cause of action arose. The parties will act in good faith to resolve disputes prior to instituting litigation. Each party waives any right to a jury trial. This Agreement is governed by the laws of the State of New York applicable to contracts executed in and performed entirely within that State govern this. This Agreement replaces all prior oral or written communications between the parties relating to the subject matter. Reproduction of this Agreement made by reliable means is considered an original, unless prohibited by local law.

Table of Contents

1 Introduction

This document provides examples of scripts that connect to the BladeCenter advanced management module (AMM) and execute CLI commands over Telnet or SSH. Scripting can help automate tasks such as gathering environmental data or setting up new systems.

2 Hardware and software requirements

To run the scripts described in this document, the advanced management module firmware must be running version 1.45x (BPET45x) or higher. Some scripts will run on previous versions of the advanced management module firmware, but might not be fully compatible.

The remote system running these scripts must be running Linux or a Linux API emulation layer capable of running Perl, Python, and Bash scripts. The system must have access to a TFTP server for uploading and downloading files and OpenSSH for creating private/public key pairs.

To run the scripts over SSH without providing a password, you must have a user account on your advanced management module that has a public key installed. This can either be done manually or by using the CreateKey, AddUser, and AddKey scripts.

For scripts ending in "_host", a host.txt file must be present in the same directory as the script. This file consists of hostnames and IP addresses on consecutive lines, with no blank lines between them.

Bash scripts connect using SSH only.

3 Example Scripts

3.1 AddKey

The AddKey script adds an SSH key to a specified user profile. To implement this script, you must know the login id number of the user profile. You must also copy the .pub key content in order to paste it in when the script asks for it. Once the key is added, you should be able to log into the host using "ssh username@host" without entering a password. The first time you log in using this method, you will be asked if you want to fingerprint the key to the host; choose yes.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-l	User Login ID number for whom you want to add the key
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.1.1 AddKey.sh

```
#!/bin/bash

user=
host=
userid=


while getopts "u:h:l:a" OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         l)
             userid=$OPTARG
             ;;
     esac
done
```

```
if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $userid ]]
then
    echo "Add a key to which Login ID Number?";
    read userid;
fi

echo "Paste the key contents below.";
read key;

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   #finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


command="ssh $name users -$userid -pk -add $key -T mm[$mm]";

data="echo `$command`";     #executes command

echo $data;
```

EXAMPLE

SSH
bash AddKey.sh –u user –h 9.37.133.212 –l 9

3.1.2 AddKey.pl

```
use Getopt::Long;

$result = GetOptions(                   #Defines Get Opt Statements
        "user=s"   => \ $user,          #Username
```

```
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "loginid=i" => \ $userid,       #User Login ID
        "type=s" => \ $type,            #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user = <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($userid eq ""){                      #If userid not opt
print "Add a key to which Login ID Number?\n";
chomp($userid =<>);
}

print "Paste the key contents below.\n";
chomp($key =<>);


if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("list -l a");
        @data = $telnet->waitfor('/system> $/i');

        $mm=findprimarymmtelnet(@data); #finds primary mm
        undef @data;                    #clears array data

        $telnet->print("users -$userid -pk -add $key -T mm[$mm]");
#adds key
        @data = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");
```

```
        pop(@data);

        print @data;
}


if($type eq "ssh"){             #ssh

        $name = $user."@".$host;

        $command ="ssh $name list -l a";
        @data = `$command`;
        checkexitcode();

        $mm = findprimarymm(@data);     #finds primary mm

        undef @data;
        $command ="";

        system("ssh $name users -$userid -pk -add $key -T mm[$mm]");
#adds key
        checkexitcode();

}

sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary
```

```
                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }
```

```
}
sub findprimarymmtelnet(@data)  #finds primary mm
{

        @data2 = split('\n',$data[0]);  #splits data into lines and puts
into data2
        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data2){                    #finds line that has mm in it
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $find = "primary";      #String to find
        $count=0;               #loop counter
        for(@mm){                       #finds line that has primary in
it
                if($_ =~ /$find/){
                        @primarymm[$count] = "$_";
                        $count++;
                }
        }
        for(@primarymm){        #gets primary mm number
                if(@primarymm[0] =~ m/(\d+)/){
                        $baystatus = $1;
                }
        }

        return $baystatus;      #return primary mm
}

sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}
```

EXAMPLE

Telnet
perl AddKey.pl –u user –p pass5 –h 9.37.133.212 –l 9 –t telnet

SSH
perl AddKey.pl –u user –h 9.37.133.212 –l 9 –t ssh

3.1.3 AddKey.py

```
import telnetlib, os
```

```
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
userid = "None";
type ="None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:l:t:a",
["username=","password=","host=","userid=","type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

    if opt in ("-l", "--userid"):     #if -l opt assigns userid
            userid = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if userid == "None":
    userid= raw_input("Add a key to which Login ID number?\n")
    userid= userid.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')
```

```
    key= raw_input("Paste the key contents below.\n")
    key= key.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telent to find mm

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("list -l a\r")
    data = telnet.read_until('system>',timeout=10)

    number=re.search(r'\bprimary\b',data).start()  #finds the string
primary and gets number space where it starts

    newdata = data[number-10:number]    #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    telnet.write("users -" +userid+ " -pk -add " +key+ " -T mm["
+mm+"]\r")  #adds key to profile
    data2 = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bError\b',data2)) == 'None':    #if statements
for output
        nothing=0
    else:
        print "\nError adding key to userid " +userid+ " profile."
        outputcheck=1;

    if str(re.search(r'\buser defined\b',data2)) == 'None':
        nothing=0
    else:
        print "\nThere is no user defined at userid " +userid+ "."
        outputcheck=1;

    if outputcheck == 0:
        print data2;


if type == "ssh":       #ssh
```

```
    key= raw_input("Paste the key contents below.\n")
    key= key.rstrip('\n')

    name= username + "@" + host     #sets up ssh command

    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    data2=[]
    data2.append(os.popen("ssh " + name + " users -" +userid+ " -pk -add
" +key+ " -T mm[" +mm+"]").read()) #executes command to add key

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bError\b',data2[0])) == 'None': #if statements
for output
        nothing=0
    else:
        print "\nError adding key to userid " +userid+ " profile."
        outputcheck=1;
```

```
    if str(re.search(r'\buser defined\b',data2[0])) == 'None':
        nothing=0
    else:
        print "\nThere is no user defined at userid " +userid+ "."
        outputcheck=1;

    if outputcheck == 0:
        print data2[0];
```

EXAMPLE

Telnet
python AddKey.py –u user –p pass5 –h 9.37.133.212 –l 9 –t telnet

SSH
python AddKey.py –u user –h 9.37.133.212 –l 9 –t ssh

3.2 AddUser

The AddUser script creates a login profile on the specified host. The profile will be created for the first available login profile. You will be prompted to enter a user name of your choice and a password for this user name that is a minimum of five characters with at least one non-alphabetic character, if not already included in the command line. After successful execution, you will see the information entered and the login id number.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-n	Username of the new login profile
-l	Password of the new login profile
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.2.1 AddUser.sh

```
#!/bin/bash

user=
host=
username=
password=
```

```
while getopts “u:h:n:p:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         n)
             username=$OPTARG
             ;;

         p)
             password=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $username ]]
then
    echo "Type the user name of your choice, must not already be used on
this host.";
    read username;
fi

if [[ -z $password ]]
then
    echo "Type a password that is a minimum of five characters with at
least one";
    echo "non-alphabetic character.";
    read password;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm
```

```
line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm



command="ssh $name users -T mm[$mm]";

`$command>data.txt`;    #executes command to find open user spot

line=`grep '\<not used\>' data.txt`;   # finds not used string to get
open user spot

`rm data.txt`;      #removes data file

userid=$(echo "$line" | sed 's/[^0-9]//g');   # gets all open user spots

useridarray=( $(echo "$userid" | sed 's/./& /g'));   #splits string that
contains all open user spots

userid=${useridarray[0]};   #assign userid to the first open user spot



command="ssh $name users -$userid -n $username -p $password -a super -T
mm[$mm]";

data="echo `$command`";         #executes command to add user


if [[ "$data" =~ "Error" ]]       #checks and prints output
then
    echo "PROFILE NOT CREATED ERROR IN PASSWORD OR USERNAME ALREADY USED
ON THIS HOST!";
    echo "Password rules:";
    echo "Passwords must be a minimum of five characters long or
empty.";
        echo "Passwords must contain at least one alphabetic and one
non-alphabetic character.";
fi

if [[ "$data" =~ "OK" ]]
then
    echo "Profile created successfully!";
    echo "Host: $host";
    echo "Username: $username";
    echo "Password: $password";
    echo "Login ID: $userid";
fi
```

```
if [[ "$data" =~ "Invalid option" ]]
then
    echo "ERROR PROFILE NOT CREATED! HOST LOGIN ID'S MAY BE ALL TAKEN."
fi

if [[ "$data" =~ "Old password" ]]
then
    echo "ERROR PROFILE NOT CREATED! USERNAME IS ALREADY TAKEN."
fi
```

EXAMPLE

SSH
bash AddUser.sh –u user –h 9.37.133.212 –n newuser –p pass5

3.2.2 AddUser.pl

```
use Getopt::Long;

$result = GetOptions(                           #Defines Get Opt
Statements
        "user=s"  => \ $user,                   #Username
        "password=s" => \ $Password,            #Password
        "host=s"  => \ $host,                   #Host Name
        "newuser=s" => \ $username,             #New Username Profile
        "loginpassword=s" => \ $userpassword,   #New User Profile
password
        "type=s" => \ $type,                    #type either ssh or
telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }
```

```
        if($username eq ""){                    #If new User not opt
        print "Type the user name of your choice, must not already be
used on this host.\n";
        chomp($username =<>);
        }
        if($userpassword eq ""){                        #If new user
password not opt
        print "\nType a password that is a minimum of five characters
with at\nleast one non-alphabetic character .\n";
        chomp($userpassword =<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("list -l a");
        @data = $telnet->waitfor('/system> $/i');


        $mm=findprimarymmtelnet(@data);                 #finds primary
mm
        undef @data;                                    #clears array
data


        $telnet->print("users -T mm[$mm]");             #finds open user
spot
        @data = $telnet->waitfor('/system> $/i');
        @splitdata = split(/\./,@data[0]);              #splits data
into sentences

        $userid=findopenuserspottelnet(@splitdata);     #finds open user
spot aka login ID
        undef @data;                                    #clears array
data

        $telnet->print("users -$userid -n $username -p $userpassword -a
super -T mm[$mm]");   #creates profile

        @data = $telnet->waitfor('/system> $/i');

        $telnet->print("exit");
}

if($type eq "ssh"){             #ssh

        if($username eq ""){                    #If new User not opt
```

```
        print "Type the user name of your choice, must not already be
used on this host.\n";
        chomp($username =<>);
        }
        if($userpassword eq ""){                        #If new user
password not opt
        print "\nType a password that is a minimum of five characters
with at\nleast one non-alphabetic character .\n";
        chomp($userpassword =<>);
        }

        $name = $user."@".$host;                #ssh command for login

        $command ="ssh $name list -l a";
        @data = `$command`;                     #executes command
        checkexitcode();

        $mm = findprimarymm(@data);             #finds primary mm

        undef @data;                            #clears variables
        $command ="";

        $command = "ssh $name users -T mm[$baystatus]";
        @data = `$command`;                     #executes command
        checkexitcode();

        $userid = findopenuserspot(@data);      #finds open user spot

        undef @data;                            #clears variables
        $command ="";

        $command = "ssh $name users -$userid -n $username -p
$userpassword -a super -T mm[$mm]";
        @data = `$command`;
        checkexitcode();
}


$old="Old password";    #user and profile name trying to create match
$error="Error";         #error check
$ok="OK";               #successfully check
$full="Invalid option"; #checks full
$authority="authority"; #authority check
$outputcheck=0;         #checks if any if statements used

for(@data){             #checks output and tells whether empty or turned
off successfully
        if($_ =~ /$error/){
                print "\nPROFILE NOT CREATED ERROR IN PASSWORD OR
USERNAME ALREADY USED ON THIS HOST!\nPassword rules:\n";
                print "Passwords must be a minimum of five characters
long or empty.\n";
```

```
                print "Passwords must contain at least one alphabetic
and one non-alphabetic character.\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "\nProfile created successfully! \nHost:
$host\nUsername: $username \nPassword: $userpassword\nLogin ID:
$userid";
                $outputcheck=1;
        }elsif($_ =~ /$full/){
                print "\nERROR PROFILE NOT CREATED! HOST LOGIN ID'S MAY
BE ALL TAKEN.\n";
                $outputcheck=1;
        }elsif($_=~ /$old/){
                print "\nERROR PROFILE NOT CREATED! USERNAME IS ALREADY
TAKEN.\n";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }
}

if($outputcheck == 0)
{
        print @data;
}

sub findprimarymmtelnet(@data)  #finds primary mm
{
        @data2 = split('\n',$data[0]);  #splits data into lines and puts
into data2
        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data2){                    #finds line that has mm in it
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $find = "primary";      #String to find
        $count=0;               #loop counter
        for(@mm){                       #finds line that has primary in
it
                if($_ =~ /$find/){
                        @primarymm[$count] = "$_";
                        $count++;
                }
        }
        for(@primarymm){        #gets primary mm number
                if(@primarymm[0] =~ m/(\d+)/){
                        $baystatus = $1;
```

```
                        }
        }

        return $baystatus;      #return primary mm
}


sub findopenuserspottelnet(@splitdata)  #This function is made to work
for this script by substracting 1 from userspot
{                               #because the data from telnet is one string
and had to spilt it
                                #so it finds the number after the not used
profile
        $find = "<not used>";
        $count =0;
        for(@splitdata){        #searches for not used profiles
                if($_ =~ /$find/){
                        @userspots[$count] = "$_";
                        $count++;
                }
        }

        if(@userspots[0] =~ m/(\d+)/){  #gets open spot number after
open spot so thats why you see -1
                $openspot = $1-1;
        }
        return $openspot;       #returns open user spot aka login ID
}

sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
```

```
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}

sub findopenuserspot(@data)     #finds open user spot
{
        $find = "<not used>";
        $count =0;
        for(@data){
                if($_ =~ /$find/){
                        @userspots[$count] = "$_";
                        $count++;
                }
        }

        if(@userspots[0] =~ m/(\d+)/){
                $openspot = $1;
        }

        return $openspot;
}

sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
```

```
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

Telnet
perl AddUser.pl –u user –p pass5 –h 9.37.133.212 –n newuser –l passw0rd –t telnet

SSH
perl AddUser.pl –u user –h 9.37.133.212 –n newuser –p pass5 –t ssh

3.2.3 AddUser.py

```
import telnetlib, os
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
user = "None";
userpassword = "None";
type = "None";
```

```
opts, args = getopt.getopt(sys.argv[1:], "u:p:h:n:l:t:a",
["username=","password=","host=","user=","userpassword=","type="]) #sets
up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

    if opt in ("-n", "--user"):   #if -n opt assings user
            user = arg

    if opt in ("-l", "--userpassword"):  #if -l opt assigns userpassword
            userpassword = arg

    if opt in ("-t", "--type"):   #if -t opt assigns type, type either
ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if user == "None":
    user= raw_input("Type the user name of your choice, must not already
be used on this host.\n")
    user= user.rstrip('\n')

if userpassword == "None":
    userpassword= raw_input("\nType a password that is a minimum of five
characters with at\nleast one non-alphabetic character.\n")
    userpassword= userpassword.rstrip('\n')


if type == "telnet":        #telnet


    if password == "None":
        password= raw_input("Enter telnet password.\n")
```

```
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telent to find mm

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("list -l a\r")
    data = telnet.read_until('system>',timeout=10)


    number=re.search(r'\bprimary\b',data).start()  #finds the string
primary and gets number space where it starts

    newdata = data[number-10:number]    #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm


    telnet.write("users -T mm[" +mm+ "]\r")   #gets users to find open
spot
    data2 = telnet.read_until('system>',timeout=10)


    try:
        number=re.search(r'\bnot used\b',data2).start()  #finds the not
used space user space
        newdata = data2[number-5:number]    #gets the not used string

        rules = re.compile('(\d+)') #sets rule to find first number
        match = rules.search(newdata)   #executes rule to find first
number which is the not used user spot

        userid=match.group(1)   #assigns open spot to userid

        telnet.write("users -" +userid+ " -n " +user+ " -p "
+userpassword+ " -a super -T mm[" +mm+"]\r") #adds user
        data3 = telnet.read_until('system>',timeout=10)
        telnet.write('exit\r')
        telnet.close()

        outputcheck=0;      #checks if any if statements used

        if str(re.search(r'\bError\b',data3)) == 'None':    #if
statements for output
            nothing=0
        else:
```

```
            print "\nPROFILE NOT CREATED ERROR IN PASSWORD OR USERNAME
ALREADY USED ON THIS HOST!\nPassword rules:"
            print "Passwords must be a minimum of five characters long
or empty."
            print "Passwords must contain at least one alphabetic and
one non-alphabetic character."
            outputcheck=1;

        if str(re.search(r'\bOK\b',data3)) == 'None':
            nothing=0
        else:
            print "\nProfile created successfully! \nHost: " +host+
"\nUsername: " +user+ "\nPassword: " +userpassword+ "\nLogin ID: "
+userid
            outputcheck=1;

        if str(re.search(r'\bOld password\b',data3)) == 'None':
            nothing=0
        else:
            print "\nERROR PROFILE NOT CREATED! USERNAME IS ALREADY
TAKEN.\n"
            outputcheck=1;

        if outputcheck == 0:
            print data3;
    except:
        print "\nHOST LOGIN ID'S ARE ALL TAKEN"


if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm


    data2=[]
    data2.append(os.popen("ssh " + name + " users -T mm["
+mm+"]").read()) #executes list command
```

```
    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    try:
        number=re.search(r'\bnot used\b',data2[0]).start()  #finds the
not used space user space
        newdata = data2[0][number-5:number] #gets the not used string

        rules = re.compile('(\d+)') #sets rule to find first number
        match = rules.search(newdata)   #executes rule to find first
number which is the not used user spot

        userid=match.group(1)   #assigns open spot to userid

        data3=[]
        data3.append(os.popen("ssh " + name + " users -" +userid+ " -n "
+user+ " -p " +userpassword+ " -a operator -T mm[" +mm+"]").read())
#places user in open spot

        outputcheck=0;      #checks if any if statements used

        if str(re.search(r'\bError\b',data3[0])) == 'None': #if
statements for output
            nothing=0
        else:
            print "PROFILE NOT CREATED ERROR IN PASSWORD OR USERNAME
ALREADY USED ON THIS HOST!\nPassword rules:"
            print "Passwords must be a minimum of five characters long
or empty."
            print "Passwords must contain at least one alphabetic and
one non-alphabetic character."
            outputcheck=1;

        if str(re.search(r'\bOK\b',data3[0])) == 'None':
            nothing=0
        else:
```

```
            print "Profile created successfully! \nHost: " +host+
"\nUsername: " +user+ "\nPassword: " +userpassword+ "\nLogin ID: "
+userid
            outputcheck=1;

        if str(re.search(r'\bOld password\b',data3[0])) == 'None':
            nothing=0
        else:
            print "ERROR PROFILE NOT CREATED! USERNAME IS ALREADY
TAKEN.\n"
            outputcheck=1;

        if outputcheck == 0:
            print data3[0];
    except:
        print "HOST LOGIN ID'S ARE ALL TAKEN"
```

EXAMPLE

Telnet
python AddUser.py –u user –p pass5 –h 9.37.133.212 –n newuser –l passw0rd –t telnet

SSH
python AddUser.py –u user –h 9.37.133.212 –n newuser –p pass5 –t ssh

3.3 AddUser_host

The AddUser_host script creates the same login profile on all of the hosts listed in the host.txt file. The profile will be created for the first available login profile that is not in use for each host. After successful execution you will see the information that you entered and the new login id number.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-n	Username of the new login profile
-p	Password of the new login profile

3.3.1 AddUser_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "newuser=s" => \ $username,     #New Username Profile
```

```
        "password=s" => \ $userpassword,        #New User Profile
password
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}

$file = 'host.txt';             #filename containing hosts

open(File,$file) || die("Could not open file!");

$host = <File>;                 #reads host from file
chomp($host);

if($username eq ""){                    #If username not opt
print "Type the user name of your choice, must not already be used on
this host.\n";
chomp($username =<>);
}

if($userpassword eq ""){                        #If password not opt
print "\nType a password that is a minimum of five characters with
at\nleast one non-alphabetic character .\n";
chomp($userpassword =<>);
}

while($host ne "")
{

        $name = $user."@".$host;

        $command ="ssh $name list -l a";
        @data = `$command`;
        checkexitcode();

        $mm = findprimarymm(@data);

        undef @data;
        $command ="";

        $command = "ssh $name users -T mm[$baystatus]";
        @data = `$command`;
        checkexitcode();

        $userid = findopenuserspot(@data);

        undef @data;
        $command ="";
```

```
        $command = "ssh $name users -$userid -n $username -p
$userpassword -a super -T mm[$mm]";
        @data = `$command`;
        checkexitcode();


        $old="Old password";    #user and profile name trying to create
match
        $error="Error";         #error check
        $ok="OK";               #successfully check
        $full="Invalid option"; #checks full
        $outputcheck=0;         #checks if any if statements used

        for(@data){             #checks output and tells whether empty
or turned off successfully
                if($_ =~ /$error/){
                        print "\nPROFILE NOT CREATED ERROR IN PASSWORD
OR USERNAME ALREADY USED ON THIS HOST!\nPassword rules:\n";
                        print "Passwords must be a minimum of five
characters long or empty.\n";
                        print "Passwords must contain at least one
alphabetic and one non-alphabetic character.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$ok/){
                        print "\nProfile created successfully! \nHost:
$host\nUsername: $username \nPassword: $userpassword\nLogin ID:
$userid";
                        $outputcheck=1;
                }elsif($_ =~ /$full/){
                        print "\nERROR PROFILE NOT CREATED! HOST LOGIN
ID'S MAY BE ALL TAKEN.\n";
                        $outputcheck=1;
                }elsif($_=~ /$old/){
                        print "\nERROR PROFILE NOT CREATED! USERNAME IS
ALREADY TAKEN.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @data;
        }

        print"\n";

        undef @data;
        $host =<File>;                  #reads host from file
        chomp($host);

}
```

```
sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}

sub findopenuserspot(@data)     #finds open user spot
{
        $find = "<not used>";
        $count =0;
        for(@data){
                if($_ =~ /$find/){
                        @userspots[$count] = "$_";
                        $count++;
                }
        }
```

```
        if(@userspots[0] =~ m/(\d+)/){
                $openspot = $1;
        }

        return $openspot;
}

sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }
```

```
}
```

EXAMPLE

perl AddUser_host.pl –u user –n newuser –p pass5

3.3.2 AddUser_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None";  #checks if opt or not
user = "None";
userpassword = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:n:p:a",
["username=","user=","userpassword="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-n", "--user"):   #if -n opt assings user
            user = arg

    if opt in ("-p", "--userpassword"):  #if -l opt assigns userpassword
            userpassword = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')

if user == "None":
    user= raw_input("Type the user name of your choice, must not already
be used on this host.\n")
    user= user.rstrip('\n')

if userpassword == "None":
    userpassword= raw_input("\nType a password that is a minimum of five
characters with at\nleast one non-alphabetic character.\n")
    userpassword= userpassword.rstrip('\n')

while host != "":   #loop until file empty

    name= username + "@" + host     #sets up ssh command
```

```
    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm


    data2=[]
    data2.append(os.popen("ssh " + name + " users -T mm["
+mm+"]").read()) #executes list command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    try:
        number=re.search(r'\bnot used\b',data2[0]).start()  #finds the
not used space user space
        newdata = data2[0][number-5:number] #gets the not used string

        rules = re.compile('(\d+)') #sets rule to find first number
        match = rules.search(newdata)   #executes rule to find first
number which is the not used user spot

        userid=match.group(1)   #assigns open spot to userid

        data3=[]
```

```
        data3.append(os.popen("ssh " + name + " users -" +userid+ " -n "
+user+ " -p " +userpassword+ " -a super -T mm[" +mm+"]").read()) #places
user in open spot

        outputcheck=0;      #checks if any if statements used

        if str(re.search(r'\bError\b',data3[0])) == 'None': #if
statements for output
            nothing=0
        else:
            print "PROFILE NOT CREATED ERROR IN PASSWORD OR USERNAME
ALREADY USED ON THIS HOST!\nPassword rules:"
            print "Passwords must be a minimum of five characters long
or empty."
            print "Passwords must contain at least one alphabetic and
one non-alphabetic character."
            outputcheck=1;

        if str(re.search(r'\bOK\b',data3[0])) == 'None':
            nothing=0
        else:
            print "Profile created successfully! \nHost: " +host+
"\nUsername: " +user+ "\nPassword: " +userpassword+ "\nLogin ID: "
+userid
            outputcheck=1;

        if str(re.search(r'\bOld password\b',data3[0])) == 'None':
            nothing=0
        else:
            print "ERROR PROFILE NOT CREATED! USERNAME IS ALREADY
TAKEN.\n"
            outputcheck=1;

        if outputcheck == 0:
            print data3[0];
    except:
        print "HOST LOGIN ID'S ARE ALL TAKEN"


    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n\n"
```

EXAMPLE

python AddUser_host.py –u user –n newuser –p pass5

3.4 BladeOff

The BladeOff script turns off the specified blade server.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-b	Blade bay number to turn off
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.4.1 BladeOff.sh

```
#!/bin/bash

user=
host=
blade=

while getopts “u:h:b:a” OPTION      #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         b)
             blade=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]           #if var not opt then prompt for user input
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $blade ]]
then
    echo "Which blade would you like to turn off?";
    read blade;
```

```
fi

name="$user@$host";

txt="ssh $name power -T blade[$blade] -off";

data="echo `$txt`";     #executes command


if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "Blade $blade has been turned off successfully."
fi

if [[ "$data" =~ "failed" ]]
then
    echo "Powering off blade $blade falied."
fi

if [[ "$data" =~ "empty" ]]
then
    echo "Target bay for blade $blade is empty."
fi

if [[ "$data" =~ "range" ]]
then
    echo "The target bay is out of range."
fi
```

EXAMPLE

SSH
bash BladeOff.sh –u user5 –h 9.37.133.212 –b 9

3.4.2 BladeOff.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "user=s"  => \ $user,   #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "blade=i" => \ $blade,  #Blade Number
        "type=s" => \ $type,  #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
```

```
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($blade == ""){                       #If Blade not opt
print "Which Blade would you like to turn off?\n";
chomp($blade =<>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("power -T blade[$blade] -off");
        @output = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");

        pop(@output);
}

if($type eq "ssh"){             #ssh

        $name = $user."@".$host;        #gets string for ssh excute

        $runner="ssh $name power -T blade[$blade] -off";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();

}

$empty="empty";
$ok="OK";
$fail = "failed";
$range = "range";
$authority="authority";
$outputcheck=0;         #checks if any if statements used
```

```
for(@output){           #checks output and tells whether empty or turned
off successfully
        if($_ =~ /$empty/){
                print "Target bay for blade $blade is empty\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "Blade $blade has been turned off
successfully.\n";
                $outputcheck=1;
        }elsif($_ =~ /$fail/){
                print "Powering off blade $blade failed.\n";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }elsif($_ =~ /$range/){
                print "The target bay is out of range.\n";
                $outputcheck=1;
        }
}

if($outputcheck == 0)
{
        print @output;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
```

```
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

Telnet
perl BladeOff.pl –u user1 –p passw0rd –h 9.37.133.212 –b 9 –t telnet

SSH
perl BladeOff.pl –u user5 –h 9.37.133.212 –b 9 -t ssh

3.4.3 BladeOff.py

```
import telnetlib, os
import sys, getopt
import re

username= "None";   #checks if opt or not
password= "None";
host = "None";
blade = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:b:t:a",
["username=","password=","host=", "blade=", "type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-b", "--blade"):      #if -b opt assigns blade
number
            blade = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg
```

```
        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if blade == "None":
    blade = raw_input("Which Blade would you like to turn off?\n")
    blade = blade.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("power -T blade[" + blade + "] -off\r")    #turns blade
off
    data = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data)) == 'None':     #if statements
for telnet output
        nothing=0
    else:
        print "\nTarget bay for blade "+str(blade)+" is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data)) == 'None':
        nothing=0
    else:
        print "\nBlade "+str(blade)+" has been turned off successfully"
```

```
        outputcheck=1;

    if str(re.search(r'\bfailed\b',data)) == 'None':
        nothing=0
    else:
        print "\nPowering off blade "+str(blade)+" failed"
        outputcheck=1;

    if str(re.search(r'\brange\b',data)) == 'None':
        nothing=0
    else:
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data)) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data;

if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]                 #creates array
    data.append(os.popen("ssh " + name + " power -T blade[" + blade + "]
-off").read()) #executes turn off blade command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    outputcheck=0;      #checks if any if statements used
```

```
    if str(re.search(r'\bempty\b',data[0])) == 'None':  #if statements
for ssh output
        nothing=0
    else:
        print "\nTarget bay for blade "+str(blade)+" is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nBlade "+str(blade)+" has been turned off successfully"
        outputcheck=1;

    if str(re.search(r'\bfailed\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nPowering off blade "+str(blade)+" failed"
        outputcheck=1;

    if str(re.search(r'\brange\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data[0];
```

EXAMPLE

Telnet
perl BladeOff.pl –u user1 –p passw0rd –h 9.37.133.212 –b 9 –t telnet

SSH
perl BladeOff.pl –u user5 –h 9.37.133.212 –b 9 -t ssh

3.5 BladeOff_all

The BladeOff_all script turns off all blade servers in the specified host.

Note: This script is designed to connect using only SSH.

Option	Description

-u	Username
-h	Hostname

3.5.1 BladeOff_all.sh

```
#!/bin/bash

user=
host=

while getopts “u:h:a” OPTION        #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]         #if not opt then prompt user for input
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

name="$user@$host";

command="ssh $name info";

`$command>data.txt`;            # executes command and puts data in file

line=`grep '\<Blade slots\>' data.txt`;   # finds Blade slots string to
get line

`rm data.txt`;      #removes data file

numofblades=$(echo "$line" | sed 's/[^1-9]//g');   # gets number of
blades

control=0;      #control for loop
```

```
while [ "$numofblades" -gt $control ]   #loops thru number of blades
do

    command2="ssh $name power -T blade[$numofblades] -off";

    data2="echo `$command2`";       #executes command


    if [[ "$data2" =~ "OK" ]]        #checks and prints output
    then
        echo "Blade $numofblades has been turned off successfully."
    fi

    if [[ "$data2" =~ "failed" ]]
    then
        echo "Powering off blade $numofblades falied."
    fi

    if [[ "$data2" =~ "empty" ]]
    then
        echo "Target bay for blade $numofblades is empty."
    fi

    if [[ "$data2" =~ "range" ]]
    then
        echo "The target bay is out of range."
    fi

    let "numofblades -= 1";
done
```

EXAMPLE

SSH
bash BladeOff_all.sh –u user5 –h 9.37.133.212

3.5.2 BladeOff_all.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
```

```
chomp($host = <>);
}

$find = "Blade slots";          #String looking for
$empty = "empty";
$ok = "OK";
$fail = "failed";
$range = "range";

$name = $user."@".$host;        #gets string for ssh excute
print "\nHost: $host\n";

$numofblades=bladescount($name);        #gets number of blades on host

while($numofblades > 0)
{
        $runner="ssh $name power -T blade[$numofblades] -off";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();


        $outputcheck=0;         #checks if any if statements used

        for(@output){           #checks output and tells whether blade
turned off successfully or empty
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$ok/){
                        print "Blade $numofblades has been turned off
successfully.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$fail/){
                        print "Powering off blade $numofblades
failed.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @output;
        }

        undef @output;

       $numofblades--;          #dec loop and blade number
}
```

```
sub bladescount($name)          #finds number of blades
{

        $command ="ssh $name info";

        @data = `$command`;

        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
```

```
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeOff_all.pl –u user5 –h 9.37.133.212

3.5.3 BladeOff_all.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"): #if -h opt assigns host
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

name= username + "@" + host #sets up ssh command


print "Host: "+host+"\n"

data=[]
data.append(os.popen("ssh " + name + " info").read())   #executes info
command

exitcodecheck=[]
```

```
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";

number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the string
Blade slots and gets number space where it starts

newdata = data[0][number:]  #puts Blade slots and data after that in a
string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

blades=match.group(1)       #assigns that number to the variable blades


array=[None]*int(blades)    #creates an array the size of the number of
blades

while blades > 0:

    array[int(blades)-1] = os.popen("ssh " + name + " power -T blade[" +
str(blades) + "] -off").read() #executes command on each blade

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bOK\b',array[int(blades)-1])) == 'None':    #if
statements for output, does search for the words off, on , or empty
        nothing=0
    else:
        print "Blade "+str(blades)+" has been turned off successfully"
        outputcheck=1;

    if str(re.search(r'\bfailed\b',array[int(blades)-1])) == 'None':
        nothing=0
    else:
        print "Powering off blade "+str(blades)+" failed"
```

```
        outputcheck=1;

    if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
        nothing=0
    else:
        print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
        outputcheck=1;

    if outputcheck == 0:
        print array[int(blades)-1];

    blades = int(blades) -1
```

EXAMPLE

python BladeOff_all.py –u user5 –h 9.37.133.212

3.6 BladeOff_host

The BladeOff_host script turns off all the blade servers for the all of the hosts listed in the host.txt file.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username

3.6.1 BladeOff_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}

$file = 'host.txt';             #filename containing hosts

open(File,$file) || die("Could not open file!");

$host = <File>;                 #reads host from file
chomp($host);
```

```
while($host ne "")
{
        $name = $user."@".$host;        #gets string for ssh excute
        print "\nHost: $host\n";

        $numofblades=bladescount($name); #gets number of blades on host

        turnbladesoff($name);           #turns blades off

        $host =<File>;                  #reads host from file
        chomp($host);

}

sub bladescount($name)          #finds number of blades
{

        $find = "Blade slots";          #String looking for

        $command ="ssh $name info";

        @data = `$command`;
        checkexitcode();

        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub turnbladesoff($name)                #turns blades off
{
  $empty = "empty";             #String looking for
  $ok = "OK";
  $fail = "failed";
  $range ="range";

  while($numofblades > 0)
  {

        $runner="ssh $name power -T blade[$numofblades] -off";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();
```

```
        $outputcheck=0;         #checks if any if statements used

        for(@output){           #checks output and tells whether blade
turn off successfully or target bay empty
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$ok/){
                        print "Blade $numofblades has been turned off
successfully\n";
                        $outputcheck=1;
                }elsif($_ =~ /$fail/){
                        print "Powering off blade $numofblades
failed.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @output;
        }

        undef @output;

       $numofblades--;          #dec loop and blade number
  }
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
```

```
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeOff_host.pl –u user5

3.6.2 BladeOff_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None";  #checks if opt or not


opts, args = getopt.getopt(sys.argv[1:], "u:a", ["username="]) #sets up
opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')
```

```
host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')


while host != "":   #loop until file empty

    name= username + "@" + host #sets up ssh command

    print "Host: "+host+"\n"

    data=[]
    data.append(os.popen("ssh " + name + " info").read())   #executes
info command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";


    number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the
string Blade slots and gets number space where it starts

    newdata = data[0][number:]  #puts Blade slots and data after that in
a string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

    blades=match.group(1)       #assigns that number to the variable
blades


    array=[None]*int(blades)    #creates an array the size of the number
of blades

    while blades > 0:
```

```
        outputcheck=0;      #checks if any if statements used

        array[int(blades)-1] = os.popen("ssh " + name + " power -T
blade[" + str(blades) + "] -off").read() #executes command on each blade

        if str(re.search(r'\bOK\b',array[int(blades)-1])) == 'None':
#if statements for output, does search for the words off, on , or empty
            nothing=0
        else:
            print "Blade "+str(blades)+" has been turned off
successfully"
            outputcheck=1;

        if str(re.search(r'\bfailed\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Powering off blade "+str(blades)+" failed"
            outputcheck=1;

        if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
            outputcheck=1;

        if outputcheck == 0:
            print array[int(blades)-1];

        blades = int(blades) -1



    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n"
```

EXAMPLE

python BladeOff_host.py –u user5

3.7 BladeOffName

The BladeOffName script turns off a blade server using the blade name on the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username

-h	Hostname
-b	Blade server name

3.7.1 BladeOffName.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to turn blade off.\n";
chomp($bladename =<>);
}

$find = "blade";

$name = $user."@".$host;

$count = 0;             #Loop control variables
$count2 = 0;
$count3 = 0;
$count4 = 0;
$bladenum = -1;  #Checks if Blade Found

$command ="ssh $name list -l a";
@data = `$command`;

checkexitcode();


for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}
```

```
for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
                $count2++;

        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
}

if($bladenum == -1){    #Checks if blade is found

        print"\nBlade Name Doesnt Exist On Given Host!\n";
}else{
@output = `ssh $name power -T blade[$bladenum] -off`;
checkexitcode();
}


$empty ="empty";        #variable to check if target bay empty
$ok = "OK";             #variable to check if turn on was succuessful
$fail = "failed";       #variable to check if failed



for(@output){           #checks output and tells whether blade turned
off successfully or empty
        if($_ =~ /$empty/){
                print "Target bay for $bladename is empty\n";

        }elsif($_ =~ /$ok/){
```

```
                print "Blade $bladename has been turned off
successfully.\n";

        }elsif($_ =~ /$fail/){
                print "Powering off $bladename failed.\n";

        }
}


sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }
```

```
        }

}
```

EXAMPLE

perl BladeOffName.pl –u user5 –h 9.37.133.212 –b Lewis

3.8 BladeOffName_host

The BladeOffName_host script turns off a blade server using the blade name, if that blade server is in a host listed in the host.txt file.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-b	Blade server name

3.8.1 BladeOffName_host.pl

```
$file = 'host.txt';

open(File,$file) || die("Could not open file!");

use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to turn blade off.\n";
chomp($bladename =<>);
}

$host = <File>;
chomp($host);

print "\nSearching.......";

$find = "blade";
```

```
$bladenum = -1;  #Checks if Blade Found

while(($host ne "") && (
$bladenum == -1)){      #Loop thru Hosts

$count = 0;             #loop control variables
$count2 = 0;
$count3 = 0;
$count4 = 0;

$name = $user."@".$host;

$command ="ssh $name list -l a";
@data = `$command`;
checkexitcode();

for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}


for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
                $count2++;

        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
```

```
}

if($bladenum == -1){    #Checks if blade is found

}else{
@output = `ssh $name power -T blade[$bladenum] -off`;
checkexitcode();
}

$empty ="empty";        #variable to check if target bay empty
$ok = "OK";             #variable to check if turn on was succuessful
$fail = "failed";       #variable to check if failed


for(@output){           #checks output and tells whether blade turned
off successfully or empty
        if($_ =~ /$empty/){
                print "Target bay for blade $bladename is empty\n";

        }elsif($_ =~ /$ok/){
                print "Blade $bladename has been turned off
successfully.\n";

        }elsif($_ =~ /$fail/){
                print "Powering off $bladename failed.\n";

        }

}


        $host =<File>;          #reads in host from file
        chomp($host);

        undef @data;            #clears arrays
        undef @blades;
        undef @line;
        undef @name;
}

if($bladenum == -1){
        print "\n\nBlade Name Does Not Exist!\n";
}

sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}
```

```
sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeOffName_host.pl –u user5 –b Lewis

3.9 BladeOn

The BladeOn script turns on the specified blade server.

Option	Description
-u	Username

-p	Password, used only when connecting via telnet
-h	Hostname
-b	Blade bay number to turn on
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.9.1 BladeOn.sh

```
#!/bin/bash

user=
host=
blade=

while getopts “u:h:b:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         b)
             blade=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $blade ]]
then
    echo "Which blade would you like to turn on?";
    read blade;
fi

name="$user@$host";
```

```
txt="ssh $name power -T blade[$blade] -on";

data="echo `$txt`";     #executes command


if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "Blade $blade has been turned on successfully."
fi

if [[ "$data" =~ "failed" ]]
then
    echo "Powering on blade $blade falied."
fi

if [[ "$data" =~ "empty" ]]
then
    echo "Target bay for blade $blade is empty."
fi

if [[ "$data" =~ "range" ]]
then
    echo "The target bay is out of range."
fi
```

EXAMPLE

SSH
bash BladeOn.sh –u user5 –h 9.37.133.212 –b 9

3.9.2 BladeOn.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "user=s"  => \ $user,   #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "blade=i" => \ $blade,  #Blade Number
        "type=s" => \ $type,  #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
```

```
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($blade == ""){                       #If Blade not opt
print "Which Blade would you like to turn on?\n";
chomp($blade =<>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("power -T blade[$blade] -on");
        @output = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");

        pop(@output);

}

if($type eq "ssh"){             #ssh

        $name = $user."@".$host;        #gets string for ssh excute
        print "\nHost: $host\n";

        $runner="ssh $name power -T blade[$blade] -on";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();
}

$empty="empty";                 #String looking for
$ok="OK";
$fail = "failed";
$range = "range";
$authority="authority";
$outputcheck=0;         #checks if any if statements used

for(@output){           #checks output and tells whether empty or turned
on successfully
```

```
        if($_ =~ /$empty/){
                print "Target bay for blade $blade is empty\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "Blade $blade has been turned on successfully.\n";
                $outputcheck=1;
        }elsif($_ =~ /$fail/){
                print "Powering on blade $blade failed.\n";
                $outputcheck=1;
        }elsif($_ =~ /$range/){
                print "The target bay is out of range.\n";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }
}

if($outputcheck == 0)
{
        print @output;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
```

```
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

Telnet
perl BladeOn.pl –u user1 –p passw0rd –h 9.37.133.212 –b 9 –t telnet

SSH
perl BladeOn.pl –u user5 –h 9.37.133.212 –b 9 -t ssh

3.9.3 BladeOn.py

```
import telnetlib, os
import sys, getopt
import re

username= "None";   #checks if opt or not
password= "None";
host = "None";
blade = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:b:t:a",
["username=","password=","host=", "blade=", "type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-b", "--blade"):      #if -b opt assigns blade
number
            blade = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg
```

```
if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if blade == "None":
    blade = raw_input("Which Blade would you like to turn on?\n")
    blade = blade.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("power -T blade[" + blade + "] -on\r") #turns blade on
    data = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data)) == 'None':     #if statements
for telnet output
        nothing=0
    else:
        print "\nTarget bay for blade "+str(blade)+" is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data)) == 'None':
        nothing=0
    else:
        print "\nBlade "+str(blade)+" has been turned on successfully"
        outputcheck=1;

    if str(re.search(r'\bfailed\b',data)) == 'None':
        nothing=0
    else:
```

```
        print "\nPowering on blade "+str(blade)+" failed"
        outputcheck=1;

    if str(re.search(r'\brange\b',data)) == 'None':
        nothing=0
    else:
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data)) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data;


if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]                 #creates array
    data.append(os.popen("ssh " + name + " power -T blade[" + blade + "]
-on").read()) #executes turn on blade command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";


    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data[0])) == 'None':  #if statements
for ssh output
```

```
        nothing=0
    else:
        print "\nTarget bay for blade "+str(blade)+" is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nBlade "+str(blade)+" has been turned on successfully"
        outputcheck=1;

    if str(re.search(r'\bfailed\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nPowering on blade "+str(blade)+" failed"
        outputcheck=1;

    if str(re.search(r'\brange\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data[0];
```

EXAMPLE

Telnet
python BladeOn.py –u user1 –p passw0rd –h 9.37.133.212 –b 9 –t telnet

SSH
python BladeOn.py –u user5 –h 9.37.133.212 –b 9 -t ssh

3.10 BladeOn_all

The BladeOn_all script turns on all blade servers in the specified host.

Note: This script is only designed to connect via SSH.

Option	Description
-u	Username
-h	Hostname

3.10.1 BladeOn_all.sh

```
#!/bin/bash

user=
host=

while getopts “u:h:a” OPTION        #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]         #if not opt then prompt user for input
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

name="$user@$host";

command="ssh $name info";

`$command>data.txt`;            # executes command and puts data in file

line=`grep '\<Blade slots\>' data.txt`;   # finds Blade slots string to
get line

`rm data.txt`;      #removes data file

numofblades=$(echo "$line" | sed 's/[^1-9]//g');   # gets number of
blades

control=0;      #control for loop

while [ "$numofblades" -gt $control ]   #loops thru number of blades
do
```

```
    command2="ssh $name power -T blade[$numofblades] -on";

    data2="echo `$command2`";       #executes command


    if [[ "$data2" =~ "OK" ]]        #checks and prints output
    then
        echo "Blade $numofblades has been turned on successfully."
    fi

    if [[ "$data2" =~ "failed" ]]
    then
        echo "Powering on blade $numofblades falied."
    fi

    if [[ "$data2" =~ "empty" ]]
    then
        echo "Target bay for blade $numofblades is empty."
    fi

    if [[ "$data2" =~ "range" ]]
    then
        echo "The target bay is out of range."
    fi

    let "numofblades -= 1";
done
```

EXAMPLE

SSH
bash BladeOn_all.sh –u user5 –h 9.37.133.212

3.10.2 BladeOn_all.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
```

```
$find = "Blade slots";          #String looking for
$empty = "empty";
$ok = "OK";
$fail = "failed";
$range = "range";

$name = $user."@".$host;        #gets string for ssh excute
print "\nHost: $host\n";

$numofblades=bladescount($name);        #gets number of blades on host

while($numofblades > 0)
{
        $runner="ssh $name power -T blade[$numofblades] -on";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();

        $outputcheck=0;         #checks if any if statements used

        for(@output){           #checks output and tells whether blade
turned on successfully or empty
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$ok/){
                        print "Blade $numofblades has been turned on
successfully.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$fail/){
                        print "Powering on blade $numofblades
failed.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @output;
        }

        undef @output;

       $numofblades--;          #dec loop and blade number
}


sub bladescount($name)          #finds number of blades
```

```
{

        $command ="ssh $name info";

        @data = `$command`;

        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }
```

```
        }

}
```

EXAMPLE

perl BladeOn_all.pl –u user5 –h 9.37.133.212

3.10.3 BladeOn_all.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"): #if -h opt assigns host
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

name= username + "@" + host #sets up ssh command


print "Host: "+host+"\n"

data=[]
data.append(os.popen("ssh " + name + " info").read())   #executes info
command

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
```

```
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";


number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the string
Blade slots and gets number space where it starts

newdata = data[0][number:]  #puts Blade slots and data after that in a
string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

blades=match.group(1)       #assigns that number to the variable blades


array=[None]*int(blades)    #creates an array the size of the number of
blades

while blades > 0:

    outputcheck=0;      #checks if any if statements used

    array[int(blades)-1] = os.popen("ssh " + name + " power -T blade[" +
str(blades) + "] -on").read() #executes command on each blade

    if str(re.search(r'\bOK\b',array[int(blades)-1])) == 'None':    #if
statements for output, does search for the words off, on , or empty
        nothing=0
    else:
        print "Blade "+str(blades)+" has been turned on successfully"
        outputcheck=1;

    if str(re.search(r'\bfailed\b',array[int(blades)-1])) == 'None':
        nothing=0
    else:
        print "Powering on blade "+str(blades)+" failed"
        outputcheck=1;

    if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
```

```
        nothing=0
    else:
        print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
        outputcheck=1;

    if outputcheck == 0:
        print array[int(blades)-1];

    blades = int(blades) -1
```

EXAMPLE

python BladeOn_all.py –u user5 –h 9.37.133.212

3.11 BladeOn_host

The BladeOn_host script turns on all the blade servers for the all of the hosts listed in the hoost.txt file.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username

3.11.1 BladeOn_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}

$file = 'host.txt';             #filename containing hosts

open(File,$file) || die("Could not open file!");

$host = <File>;                 #reads host from file
chomp($host);



while($host ne "")
```

```
{
        $name = $user."@".$host;        #gets string for ssh excute
        print "\nHost: $host\n";

        $numofblades=bladescount($name); #gets number of blades on host

        turnbladeson($name);            #turns blades on

        $host =<File>;                  #reads host from file
        chomp($host);

}

sub bladescount($name)          #finds number of blades
{

        $find = "Blade slots";          #String looking for

        $command ="ssh $name info";

        @data = `$command`;
        checkexitcode();

        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub turnbladeson($name)         #turns blades on
{
  $empty = "empty";             #String looking for
  $ok = "OK";
  $fail = "failed";
  $range = "range";

  while($numofblades > 0)
  {

        $runner="ssh $name power -T blade[$numofblades] -on";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();


        $outputcheck=0;         #checks if any if statements used
```

```
        for(@output){           #checks output and tells whether blade
turn on successfully or target bay empty
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$ok/){
                        print "Blade $numofblades has been turned on
successfully\n";
                        $outputcheck=1;
                }elsif($_ =~ /$fail/){
                        print "Powering on blade $numofblades
failed.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }


        undef @output;

  if($outputcheck == 0)
  {
        print @output;
  }

      $numofblades--;           #dec loop and blade number
  }
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
```

```
                print "Command Line Usage Error";
        }
        if($value == 127)
        {
                print "Command Not Found";
        }
        if($value == 126)
        {
                print "User does not have permission";
        }
        if($value == 128)
        {
                print "This value is strictly internal to AMM";
        }

    }

}
```

EXAMPLE

perl BladeOn_host.pl –u user5

3.11.2 BladeOn_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None";  #checks if opt or not


opts, args = getopt.getopt(sys.argv[1:], "u:a", ["username="]) #sets up
opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')



host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')
```

```
while host != "":   #loop until file empty

    name= username + "@" + host #sets up ssh command

    print "Host: "+host+"\n"

    data=[]
    data.append(os.popen("ssh " + name + " info").read())   #executes
info command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";


    number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the
string Blade slots and gets number space where it starts

    newdata = data[0][number:]  #puts Blade slots and data after that in
a string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

    blades=match.group(1)       #assigns that number to the variable
blades


    array=[None]*int(blades)    #creates an array the size of the number
of blades

    while blades > 0:

        array[int(blades)-1] = os.popen("ssh " + name + " power -T
blade[" + str(blades) + "] -on").read() #executes command on each blade
```

```
        outputcheck=0;      #checks if any if statements used

        if str(re.search(r'\bOK\b',array[int(blades)-1])) == 'None':
#if statements for output, does search for the words off, on , or empty
            nothing=0
        else:
            print "Blade "+str(blades)+" has been turned on
successfully"
            outputcheck=1;

        if str(re.search(r'\bfailed\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Powering on blade "+str(blades)+" failed"
            outputcheck=1;

        if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
            outputcheck=1;

        if outputcheck == 0:
            print array[int(blades)-1];

        blades = int(blades) -1


    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n"
```

EXAMPLE

python BladeOn_host.py –u user5

3.12 BladeOnName

The BladeOnName script turns on a blade server using the blade name in the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-b	Blade server name

3.12.1 BladeOnName.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to turn blade on.\n";
chomp($bladename =<>);
}


$find = "blade";

$name = $user."@".$host;

$count = 0;             #Loop control variables
$count2 = 0;
$count3 = 0;
$count4 = 0;
$bladenum = -1;  #Checks if Blade Found

$command ="ssh $name list -l a";
@data = `$command`;
checkexitcode();

for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}


for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
```

```
                $count2++;

        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
}

if($bladenum == -1){    #Checks if blade is found

        print"\nBlade Name Doesnt Exist On Given Host!\n";
}else{
@output = `ssh $name power -T blade[$bladenum] -on`;
checkexitcode();
}


$empty ="empty";        #variable to check if target bay empty
$ok = "OK";             #variable to check if turn on was succuessful
$fail = "failed";
$range = "range";

for(@output){           #checks output and tells whether empty or turned
on successfully
        if($_ =~ /$empty/){
                print "Target bay for $bladename is empty\n";
        }elsif($_ =~ /$ok/){
                print "Blade $bladename has been turned on
successfully.\n";
        }elsif($_ =~ /$fail/){
                print "Powering on $bladename failed.\n";
        }elsif($_ =~ /$range/){
                print "The target bay is out of range.\n";
        }
}
```

```
sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeOnName.pl –u user5 –h 9.37.133.212 –b Lewis

3.13 BladeOnName_host

The BladeOnName_host script turns on a blade server using the blade name if that blade is in a host listed in the host.txt file.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-b	Blade server name

3.13.1 BladeOnName_host.pl

```
$file = 'host.txt';

open(File,$file) || die("Could not open file!");

use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to turn blade on.\n";
chomp($bladename =<>);
}

$host = <File>;
chomp($host);

print "\nSearching.......";

$find = "blade";

$bladenum = -1;  #Checks if Blade Found

while(($host ne "") && (
$bladenum == -1)){      #Loop thru Hosts

$count = 0;             #loop control variables
```

```
$count2 = 0;
$count3 = 0;
$count4 = 0;

$name = $user."@".$host;

$command ="ssh $name list -l a";
@data = `$command`;
checkexitcode();

for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}


for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
                $count2++;

        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
}

if($bladenum == -1){    #Checks if blade is found

}else{
@output = `ssh $name power -T blade[$bladenum] -on`;
checkexitcode();
```

```
}

$empty ="empty";        #variable to check if target bay empty
$ok = "OK";             #variable to check if turn on was succuessful
$fail = "failed";
$range = "range";

for(@output){           #checks output and tells whether empty or turned
on successfully
        if($_ =~ /$empty/){
                print "Target bay for $bladename is empty\n";
        }elsif($_ =~ /$ok/){
                print "Blade $bladename has been turned on
successfully.\n";
        }elsif($_ =~ /$fail/){
                print "Powering on $bladename failed.\n";
        }elsif($_ =~ /$range/){
                print "The target bay is out of range.\n";
        }
}

        $host =<File>;
        chomp($host);

        undef @data;
        undef @blades;
        undef @line;
        undef @name;
}

if($bladenum == -1){
        print "\n\nBlade Name Does Not Exist!\n";
}

sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
```

```
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeOnName_host.pl –u user5 –b Lewis

3.14 BladeState

The BladeState script returns a statement indicating the on or off state of the specified blade server.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-b	Blade bay number to check state
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.14.1 BladeState.sh

```
#!/bin/bash

user=
host=
blade=

while getopts “u:h:b:a” OPTION          #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         b)
             blade=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]           #if options not opt then prompt for input
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $blade ]]
then
    echo "Which blade would you like to check the state of?";
    read blade;
fi


name="$user@$host";


    txt="ssh $name power -T blade[$blade] -state";

    data="echo `$txt`";         #executes command


    if [[ "$data" =~ "Off" ]]       #checks and prints output
```

```
    then
        echo "Blade $blade is off."
    fi

    if [[ "$data" =~ "On" ]]
    then
        echo "Blade $blade is on."
    fi

    if [[ "$data" =~ "empty" ]]
    then
        echo "Target bay for blade $blade is empty."
    fi

    if [[ "$data" =~ "range" ]]
    then
        echo "The target bay is out of range."
    fi
```

EXAMPLE

SSH
bash BladeState.sh –u user12 –h 9.37.133.212 –b 3

3.14.2 BladeState.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "user=s"  => \ $user,           #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "blade=i" => \ $blade,          #Blade Number
        "type=s" => \ $type,            #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($blade == ""){                       #If Blade not opt
print "Which Blade would you like to check the state of?\n";
chomp($blade =<>);
```

```
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #if password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("power -T blade[$blade] -state");
        @output = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");

        pop(@output);
}

if($type eq "ssh"){             #ssh

        $name = $user."@".$host;        #gets string for ssh excute
        print "\nHost: $host\n";

        $runner="ssh $name power -T blade[$blade] -state";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();
}

$empty = "empty";
$on = "On";
$off = "Off";
$range = "range";
$authority="authority";
$outputcheck=0;         #checks if any if statements used

for(@output){           #checks output and tells whether blade on, off,
or empty
                if($_ =~ /$empty/){
                        print "Target bay for blade $blade is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$on/){
                        print "Blade $blade is On\n";
                        $outputcheck=1;
                }elsif($_ =~ /$off/){
                        print "Blade $blade is Off\n";
```

```
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }elsif($_ =~ /$authority/){
                        print "\nUser does not have the authority to
issue this command.";
                        $outputcheck=1;
                }
}

if($outputcheck == 0)
{
        print @output;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }
```

```
}
```

EXAMPLE

Telnet
perl BladeState.pl –u user –p passw0rd –h 9.37.133.212 –b 2 –t telnet

SSH
perl BladeState.pl –u user12 –h 9.37.133.212 –b 3 -t ssh

3.14.3 BladeState.py

```
import telnetlib, os
import sys, getopt
import re

username= "None";   #checks if opt or not
password= "None";
host = "None";
blade = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:b:t:a",
["username=","password=","host=", "blade=", "type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-b", "--blade"):      #if -b opt assigns blade
number
            blade = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')
```

```
if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if blade == "None":
    blade = raw_input("Which Blade would you like to check the state
of?\n")
    blade = blade.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("power -T blade[" + blade + "] -state\r")  #checks
blade state
    data = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bOff\b',data)) == 'None':   #if statements for
telnet output
        nothing=0
    else:
        print "\nBlade "+str(blade)+" is Off"
        outputcheck=1;

    if str(re.search(r'\bOn\b',data)) == 'None':
        nothing=0
    else:
        print "\nBlade "+str(blade)+" is On"
        outputcheck=1;

    if str(re.search(r'\bempty\b',data)) == 'None':
        nothing=0
    else:
        print "\nTarget bay for blade "+str(blade)+ " is empty"
        outputcheck=1;

    if str(re.search(r'\brange\b',data)) == 'None':
        nothing=0
    else:
```

```
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data)) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data;


if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]                 #creates array
    data.append(os.popen("ssh " + name + " power -T blade[" + blade + "]
-state").read()) #executes check blade state command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";


    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bOff\b',data[0])) == 'None':    #if statements
for ssh output
        nothing=0
    else:
        print "Blade "+str(blade)+" is Off"
        outputcheck=1;

    if str(re.search(r'\bOn\b',data[0])) == 'None':
```

```
        nothing=0
    else:
        print "Blade "+str(blade)+" is On"
        outputcheck=1;

    if str(re.search(r'\bempty\b',data[0])) == 'None':
        nothing=0
    else:
        print "Target bay for blade "+str(blade)+ " is empty"
        outputcheck=1;

    if str(re.search(r'\brange\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nThe target bay is out of range."
        outputcheck=1;

    if str(re.search(r'\bauthority\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nUser does not have the authority to issue this
command."
        outputcheck=1;

    if outputcheck == 0:
        print data[0];
```

EXAMPLE

Telnet
python BladeState.py –u user –p passw0rd –h 9.37.133.212 –b 2 –t telnet

SSH
python BladeState.py –u user12 –h 9.37.133.212 –b 3 -t ssh

3.15 BladeState_all

The BladeState_all script returns a statement indicating the current on or off state for each blade server in the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname

3.15.1 BladeState_all.sh

```
#!/bin/bash

user=
host=

while getopts “u:h:a” OPTION        #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]         #if not opt then prompt user for input
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

name="$user@$host";

command="ssh $name info";

`$command>data.txt`;            # executes command and puts data in file

line=`grep '\<Blade slots\>' data.txt`;   # finds Blade slots string to
get line

`rm data.txt`;      #removes data file

numofblades=$(echo "$line" | sed 's/[^1-9]//g');   # gets number of
blades

control=0;      #control for loop

while [ "$numofblades" -gt $control ]   #loops thru number of blades
do

    command2="ssh $name power -T blade[$numofblades] -state";

    data2="echo `$command2`";       #executes command
```

```
    if [[ "$data2" =~ "Off" ]]      #checks and prints output
    then
        echo "Blade $numofblades is off."
    fi

    if [[ "$data2" =~ "On" ]]
    then
        echo "Blade $numofblades is on."
    fi

    if [[ "$data2" =~ "empty" ]]
    then
        echo "Target bay for blade $numofblades is empty."
    fi

    if [[ "$data2" =~ "range" ]]
    then
        echo "The target bay is out of range."
    fi

    let "numofblades -= 1";
done
```

EXAMPLE

SSH
bash BladeState_all.sh –u user5 –h 9.37.133.212

3.15.2 BladeState_all.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}

$find = "Blade slots";          #String looking for
$empty = "empty";
$on = "On";
$off = "Off";
```

```
$range = "range";

$name = $user."@".$host;        #gets string for ssh excute
print "\nHost: $host\n";

$numofblades=bladescount($name);        #gets number of blades on host

while($numofblades > 0)
{
        $runner="ssh $name power -T blade[$numofblades] -state";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();

        $outputcheck=0;         #checks if any if statements used

        for(@output){           #checks output and tells whether blade
on, off, or empty
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$on/){
                        print "Blade $numofblades is On\n";
                        $outputcheck=1;
                }elsif($_ =~ /$off/){
                        print "Blade $numofblades is Off\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @output;
        }

        undef @output;

       $numofblades--;          #dec loop and blade number
}



sub bladescount($name)          #finds number of blades
{

        $command ="ssh $name info";

        @data = `$command`;
```

```
        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeState_all.pl –u user5 –h 9.37.133.212

3.15.3 BladeState_all.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"): #if -h opt assigns host
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

name= username + "@" + host #sets up ssh command


print "\nRunning this will return the state of all the blades for the
host.\n"

print "Host: "+host+"\n"

data=[]
data.append(os.popen("ssh " + name + " info").read())   #executes info
command

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
```

```
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";


number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the string
Blade slots and gets number space where it starts

newdata = data[0][number:]  #puts Blade slots and data after that in a
string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

blades=match.group(1)       #assigns that number to the variable blades


array=[None]*int(blades)    #creates an array the size of the number of
blades

while blades > 0:

    array[int(blades)-1] = os.popen("ssh " + name + " power -T blade[" +
str(blades) + "] -state").read() #executes command on each blade

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bOff\b',array[int(blades)-1])) == 'None':   #if
statements for output, does search for the words off, on , or empty
        nothing=0
    else:
        print "Blade "+str(blades)+" is Off"
        outputcheck=1;

    if str(re.search(r'\bOn\b',array[int(blades)-1])) == 'None':
        nothing=0
    else:
        print "Blade "+str(blades)+" is On"
        outputcheck=1;

    if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
        nothing=0
    else:
```

```
        print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
        outputcheck=1;

    if outputcheck == 0:
        print array[int(blades)-1];

    blades = int(blades) -1
```

EXAMPLE

python BladeState_all.py –u user5 –h 9.37.133.212

3.16 BladeState_host

The BladeState_host script returns a statement indicating the current on or off state for each blade server in the hosts listed in the host.txt file.

Option	Description
-u	Username

3.16.1 BladeState_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}

$file = 'host.txt';             #filename containing hosts

open(File,$file) || die("Could not open file!");

$host = <File>;                 #reads host from file
chomp($host);



while($host ne "")
{
        $name = $user."@".$host;        #gets string for ssh excute
        print "\nHost: $host\n";
```

```
        $numofblades=bladescount($name); #gets number of blades on host

        getstate($name);                 #get and print blade state

        $host =<File>;                   #reads host from file
        chomp($host);

}

sub bladescount($name)          #finds number of blades
{

        $find = "Blade slots";          #String looking for

        $command ="ssh $name info";

        @data = `$command`;
        checkexitcode();

        for(@data){                     #finds string in find
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){         #finds first number
                $numofblades = $1;
        }

        return $numofblades;
}

sub getstate($name)
{
  $empty = "empty";             #String looking for
  $on = "On";
  $off = "Off";
  $range = "range";

  while($numofblades > 0)
  {



        $runner="ssh $name power -T blade[$numofblades] -state";

        @output = `$runner`;    #excutes ssh line
        checkexitcode();

        $outputcheck=0;         #checks if any if statements used

        for(@output){           #checks output and tells whether blade
on, off, or empty
```

```
                if($_ =~ /$empty/){
                        print "Skipping....Target bay for blade
$numofblades is empty\n";
                        $outputcheck=1;
                }elsif($_ =~ /$on/){
                        print "Blade $numofblades is On\n";
                        $outputcheck=1;
                }elsif($_ =~ /$off/){
                        print "Blade $numofblades is Off\n";
                        $outputcheck=1;
                }elsif($_ =~ /$range/){
                        print "The target bay is out of range.\n";
                        $outputcheck=1;
                }
        }

        if($outputcheck == 0)
        {
                print @output;
        }

        undef @output;

       $numofblades--;          #dec loop and blade number
  }
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
```

```
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeState_host.pl –u user5

3.16.2 BladeState_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None";  #checks if opt or not


opts, args = getopt.getopt(sys.argv[1:], "u:a", ["username="]) #sets up
opt


for opt, arg in opts:
    if opt in ("-u", "--username"):  #if -u opt assigns username
            username = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

print "\nRunning this will return the state of all the blades for the
hosts.\n"


host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')


while host != "":   #loop until file empty

    name= username + "@" + host #sets up ssh command
```

```
    print "Host: "+host+"\n"

    data=[]
    data.append(os.popen("ssh " + name + " info").read())   #executes
info command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";


    number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the
string Blade slots and gets number space where it starts

    newdata = data[0][number:]  #puts Blade slots and data after that in
a string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

    blades=match.group(1)       #assigns that number to the variable
blades


    array=[None]*int(blades)    #creates an array the size of the number
of blades

    while blades > 0:

        array[int(blades)-1] = os.popen("ssh " + name + " power -T
blade[" + str(blades) + "] -state").read() #executes command on each
blade

        outputcheck=0;      #checks if any if statements used
```

```
        if str(re.search(r'\bOff\b',array[int(blades)-1])) == 'None':
#if statements for output, does search for the words off, on , or empty
            nothing=0
        else:
            print "Blade "+str(blades)+" is Off"
            outputcheck=1;

        if str(re.search(r'\bOn\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Blade "+str(blades)+" is On"
            outputcheck=1;

        if str(re.search(r'\bempty\b',array[int(blades)-1])) == 'None':
            nothing=0
        else:
            print "Skipping....Target bay for blade "+str(blades)+ " is
empty"
            outputcheck=1;

        if outputcheck == 0:
            print array[int(blades)-1];

        blades = int(blades) -1


    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n"
```

EXAMPLE

python BladeState_host.py –u user5

3.17 BladeStateName

The BladeStateName script returns a statement indicating whether a blade server is currently on or off, by the blade server name, in the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-b	Blade server name

3.17.1 BladeStateName.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to check if the blade is on or off.\n";
chomp($bladename =<>);
}

$find = "blade";

$name = $user."@".$host;

$count = 0;             #Loop control variables
$count2 = 0;
$count3 = 0;
$count4 = 0;
$bladenum = -1;  #Checks if Blade Found
$on = "On";
$off = "Off";

$command ="ssh $name list -l a";
@data = `$command`;
checkexitcode();

for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}


for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
                $count2++;
```

```
        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
}

if($bladenum == -1){    #Checks if blade is found

        print"\nBlade Name Doesnt Exist On Given Host!\n";
}else{
@output = `ssh $name power -T blade[$bladenum] -state`;
checkexitcode();
}

for(@output){
                if($_ =~ /$on/){
                        print "\nBlade $bladename is On.\n";
                }elsif($_ =~ /$off/){
                        print "\nBlade $bladename is Off.\n";
                }
        }


sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}



sub checkexitcode()
{
```

```
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BladeStateName.pl –u user5 –h 9.37.133.212 –b Lewis

3.18 BladeStateName_host

The BladeStateName_host script returns a statement indicating whether a blade server is currently on or off, by the blade server name, for blades in the hosts listed in the host.txt file.

Note: This script is only designed to connect via SSH.

Option	Description
-u	Username

-b	Blade server name

3.18.1 BladeStateName_host.pl

```
$file = 'host.txt';

open(File,$file) || die("Could not open file!");

use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "bladename=s" => \ $bladename,  #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($bladename eq ""){                   #If Blade not opt
print "Enter blade name to check if the blade is on or off.\n";
chomp($bladename =<>);
}

$host = <File>;
chomp($host);

print "\nSearching.......";

$find = "blade";

$on = "On";
$off = "Off";
$bladenum = -1;  #Checks if Blade Found

while(($host ne "") && (
$bladenum == -1)){      #Loop thru Hosts

$count = 0;             #loop control variables
$count2 = 0;
$count3 = 0;
$count4 = 0;

$name = $user."@".$host;

$command ="ssh $name list -l a";
@data = `$command`;
checkexitcode();
```

```
for(@data){                     #Gets Blade list and names
        if($_ =~ /$find/){
                @blades[$count] = "$_";
                $count++;
        }
}


for(@blades){                   #Finds Lines of Blades that contain name
searching for
        if($_ =~ /$bladename/){
                @line[$count2] = "$_";
                $count2++;

        }
}

for(@line){                     #Gets just Blade Names
        if(@line[$count3] =~ m/](.*?)\n/){
        @name[$count3] = $1;
        $count3++;
        }
}

for(@name){             #Matches for Bladename and gets Blade number

        @name[$count4]=trim(@name[$count4]);

        if(@name[$count4] eq $bladename){

        if(@line[$count4] =~ m/(\d+)/){
                $bladenum = $1;
        }

        }
$count4++;
}

if($bladenum == -1){    #Checks if blade is found

}else{
@output = `ssh $name power -T blade[$bladenum] -state`;
checkexitcode();
}

for(@output){
                if($_ =~ /$on/){
                        print "\n\nBlade $bladename is On.\n";
                }elsif($_ =~ /$off/){
                        print "\n\nBlade $bladename is Off.\n";
                }
        }
```

```
        $host =<File>;
        chomp($host);

        undef @data;
        undef @blades;
        undef @line;
        undef @name;
}

if($bladenum == -1){
        print "\n\nBlade Name Does Not Exist!\n";
}

sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
```

```
                {
                        print "This value is strictly internal to AMM";
                }
        }
}
```

EXAMPLE

perl BladeStateName_host.pl –u user5 –b Lewis

3.19 BTemp

The BTemp script returns the current temperature of the specified blade server.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-b	Blade bay number to display temperature

3.19.1 BTemp.sh

```
#!/bin/bash

user=
host=
blade=

while getopts “u:h:b:a” OPTION          #opt options
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         b)
             blade=$OPTARG
             ;;
     esac
done

if [[ -z $user ]]          #if options not opt then prompt for input
```

```
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $blade ]]
then
    echo "Which blade would you like to see the temperature of?";
    read blade;
fi


name="$user@$host";


txt="ssh $name temps -T blade[$blade]";

`$txt>data.txt`;         #executes command

num=0;                  #loop control counter for arrays

while read line         #reads each line of the file data.txt which has
the command output from above
do
    lines=( $(echo "$line" | sed 's/./& /g'));   #splits string that
contains all open user spots

        echo $line;

done < "data.txt"

`rm data.txt`;
```

EXAMPLE

SSH
bash BTemp.sh –u user –h 9.37.133.212 –b 2

3.19.2 BTemp.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
```

```
        "blades=i" => \ $blades,        #Blade Number
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;                #Sets name to ssh login command


if($blades == ""){                      #If Blade not opt
print "Which blade would you like to see the temperature of?\n";
chomp($blades =<>);
}

$command ="ssh $name temps -T blade[$blades]";  #Places get temp blade
command in array
@data = `$command`;
checkexitcode();

print "\nHost: ".$host."\n\n";
print "Blade $blades Temperature\n------------------------------\n";


$count = 5;     #Postion of first needed data excludes Titles

for(@data){                     #Loop goes for all blade comp temps, ex.
Bank Temp1 and CPU Temp1
        $number = 0;            #control for loop to break
        $counter =0;            #holds actual data spot
        @words = split(/ /, @data[$count]);  #splits line to get words
        $numofwords=@words;     #gets number of words

        if($numofwords ne 0){

        while($number < $numofwords){   #gets rid of spaces
                if(@words[$number] ne "")
                {
                        @actual[$counter]=@words[$number];
                        $counter++;
                }
                $number++
        }

        $numofactual=@actual;   #gets number of actual data

        $number=0;      #resets variable for next loop
```

```
        $counter=0;     #resets variable for next loop

        while($number < $numofactual){    #gets all temperatures
                if($actual[$number] =~ m/(\W)/) {
                        @temp[$counter] = $actual[$number];
                        $counter++;
                }
                $number++;
        }
        $numname=$number-$counter;      #gets spots of name

        $number=0;      #resets variable for next loop

        $wholename ="";                 #resets whole name
        while($number < $numname){      #gets name
                $name=$actual[$number]." ";
                $wholename = $wholename.$name;
                $number++;
        }

        @names[$count-5]=$wholename;    #puts name in an array
        @temps[$count-5]=$temp[0];      #puts blade temp in array

        $count++;               #increments count for next line
        }

}
        if($temp[0] eq ""){             #check if temp found
                print "No temperature available for blade $blades at
this time.";

        }else{                          #prints name and temps
                $number=0;
                while($number<$count-5){
                        print @names[$number]."\t".@temps[$number]."\n";
                        $number++;
                }
        }

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
```

```
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BTemp.pl –u user –h 9.37.133.212 –b 2

3.19.3 BTemp.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";
blade = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:h:b:a", ["username=",
"host=", "blade="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-b", "--blade"):      #if -b opt assigns blade
number
            blade = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
```

```
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if blade == "None":
    blade = raw_input("Which Blade would you like to see the temperature
of?\n")
    blade = blade.rstrip('\n')

name= username + "@" + host     #sets up ssh command


print "\nHost: " +host
print "\nBlade " +blade+ " Temperature\n---------------------------------
-\n"

data=[]
data.append(os.popen("ssh " + name + " temps -T blade[" + blade +
"]").read()) #executes get temperature command

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";


line = data[0].split('\n')  #splits the lines of data from the list
command into an array called lines

numoflines=len(line)        #gets number of lines of data
```

```
temperature ="None"         #checks if no available temperature

for count in range(5,numoflines):

    temperature =""         #resets temperature
    if line[count] != "":       #avoids white space lines
        words = line[count].split() #splits line into words

        numofwords=len(words)       #counts number of words

        for count2 in range(0,numofwords):  #checks to see if word is
non word aka number to find the temp number only
            rules = re.compile(r'\W+')  #sets rule to find non word
            match = rules.search(words[count2]) #executes rule on words

            if str(match) == 'None':    #if rule not found
                nothing=0
            else:
                number = count2     #when rule found assigns that spot
to var number, that spot is the first temp which is the temp of the
blade comp
                break           #breaks out of loop because data that
was needed is found
        for count3 in range(0,number):      #gets temp component name
assigns to variable
            temperature = temperature + words[count3] + " "

        temperature = temperature +"\t"+ words[number] + " "  #gets
actual temp and assigns to variable
        print temperature   #prints temperautre

if temperature == "None":   #if no temp
    print "No temperature available for blade " +blade+ " at this time."
```

EXAMPLE

python BTemp.py –u user –h 9.37.133.212 –b 2

3.20 BTemp_all

The BTemp_all script returns the current temperature for all blade servers in the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname

3.20.1 BTemp_all.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}

print "Running this will return the temperature of all the blades for
the host.\n";

$name = $user."@".$host;                #Sets name to ssh login command
print "\nHost: ".$host."\n\n";           #Displays host name

$numofblades = bladescount($name);      #Number of blades

while($numofblades > 0)                 #Loops thru all blades on host
{

        print "Blade $numofblades Temperature\n-------------------------
-----\n";

        $command ="ssh $name temps -T blade[$numofblades]";     #Places
get temp blade command in array
        @data = `$command`;
        checkexitcode();


        $count = 5;     #Postion of first needed data excludes Titles

        for(@data){                     #Loop goes for all blade comp
temps, ex. Bank Temp1 and CPU Temp1
                $number = 0;            #control for loop to break
                $counter =0;            #holds actual data spot
                @words = split(/ /, @data[$count]);  #splits line to get
words
                $numofwords=@words;     #gets number of words

                if($numofwords ne 0){

                while($number < $numofwords){   #gets rid of spaces
                        if(@words[$number] ne "")
                        {
```

```
                        @actual[$counter]=@words[$number];
                        $counter++;
                }
                $number++
        }

        $numofactual=@actual;   #gets number of actual data

        $number=0;      #resets variable for next loop
        $counter=0;     #resets variable for next loop

        while($number < $numofactual){    #gets all temperatures
                if($actual[$number] =~ m/(\W)/) {
                        @temp[$counter] = $actual[$number];
                        $counter++;
                }
                $number++;
        }
        $numname=$number-$counter;      #gets spots of name

        $number=0;      #resets variable for next loop

        $wholename ="";                 #resets whole name
        while($number < $numname){      #gets name
                $bladename=$actual[$number]." ";
                $wholename = $wholename.$bladename;
                $number++;
        }

        @names[$count-5]=$wholename;    #puts name in an array
        @temps[$count-5]=$temp[0];      #puts blade temp in
array

        $count++;               #increments count for next line
        }

}

if($temp[0] eq ""){             #check if temp found
        print "No temperature available for blade $numofblades
at this time.\n";

}else{                          #prints name and temps
        $number=0;
        while($number<$count-5){
        print @names[$number]."\t".@temps[$number]."\n";
        $number++;
        }
}

print"\n";

undef @names;                   #Clears all arrays
```

```
        undef @temps;
        undef @acutual;
        undef @temp;
        undef @words;
        undef @data;

        $numofblades--;

}

sub bladescount($name)          #finds number of blades
{
        $find = "Blade slots";

        $command ="ssh $name info";

        @data = `$command`;

        for(@data){
                if($_ =~ /$find/) {
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){
                $numofblades = $1;
        }

        return $numofblades;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
```

```
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BTemp_all.pl –u user5 –h 9.37.133.212

3.20.2 BTemp_all.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";


opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

name= username + "@" + host     #sets up ssh command
```

```
data=[]
data.append(os.popen("ssh " + name + " info").read())   #executes info
command

number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the string
Blade slots and gets number space where it starts

newdata = data[0][number:]  #puts Blade slots and data after that in a
string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

blades=match.group(1)       #assigns that number to the variable blades

print "Running this will return the temperature of all the blades for
the host.\n"

print "\nHost: " +host


while blades > 0:
    print "\nBlade " +str(blades)+ " Temperature\n----------------------
-----------"

    data2=[]
    data2.append(os.popen("ssh " + name + " temps -T blade["
+str(blades)+ "]").read()) #executes get temperature command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    line = data2[0].split('\n') #splits the lines of data from the list
command into an array called lines
```

```
    numoflines=len(line)        #gets number of lines of data

    temperature ="None"         #checks if no available temperature

    for count in range(5,numoflines):

        temperature =""         #resets temperature
        if line[count] != "":       #avoids white space lines
            words = line[count].split() #splits line into words

            numofwords=len(words)       #counts number of words

            for count2 in range(0,numofwords):  #checks to see if word
is non word aka number to find the temp number only
                rules = re.compile(r'\W+')  #sets rule to find non word
                match = rules.search(words[count2]) #executes rule on
words

                if str(match) == 'None':    #if rule not found
                    nothing=0
                else:
                    number = count2     #when rule found assigns that
spot to var number, that spot is the first temp which is the temp of the
blade comp
                    break           #breaks out of loop because data
that was needed is found
            for count3 in range(0,number):      #gets temp component
name assigns to variable
                temperature = temperature + words[count3] + " "

            temperature = temperature +"\t"+ words[number] + " "  #gets
actual temp and assigns to variable
            print temperature   #prints temperautre

    if temperature == "None":   #if no temp
        print "No temperature available for blade " +str(blades)+ " at
this time."

    blades = int(blades)-1
```

EXAMPLE

python BTemp_all.py –u user5 –h 9.37.133.212

3.21 BTemp_host

The BTemp_host script will return a statement telling the user the current temperature for all of the blades on the hosts listed in the host.txt file.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username

3.21.1 BTemp_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
$file = 'host.txt';             #filename containing host

open(File,$file) || die("Could not open file!");        #opens file

$host = <File>;                 #reads host from file
chomp($host);


while($host ne "")
{
        $name = $user."@".$host;                #Sets name to ssh login
command
        print "\nHost: ".$host."\n\n";          #Displays host name

        $numofblades=bladescount($name);        #Number of blades

        gettemp($name);         #get and print blade temp

        $host =<File>;                  #reads host from file
        chomp($host);
}

sub bladescount($name)          #finds number of blades
{
        $find = "Blade slots";

        $command ="ssh $name info";

        @data = `$command`;
        checkexitcode();

        for(@data){
                if($_ =~ /$find/) {
```

```
                        $Slots = "$_\n";
                }
        }

        if($Slots =~ m/(\d+)/){
                $numofblades = $1;
        }

        undef @data;
        return $numofblades;
}


sub gettemp($name)
{

   while($numofblades > 0)                      #Loops thru all blades
on host
   {

        print "Blade $numofblades Temperature\n-------------------------
-----\n";

        $command ="ssh $name temps -T blade[$numofblades]";     #Places
get temp blade command in array
        @data = `$command`;

        #print @data;

        $count = 5;     #Postion of first needed data excludes Titles

        for(@data){                     #Loop goes for all blade comp
temps, ex. Bank Temp1 and CPU Temp1
                $number = 0;            #control for loop to break
                $counter =0;            #holds actual data spot
                @words = split(/ /, @data[$count]);  #splits line to get
words
                $numofwords=@words;     #gets number of words

                if($numofwords ne 0){

                while($number < $numofwords){   #gets rid of spaces
                        if(@words[$number] ne "")
                        {
                                @actual[$counter]=@words[$number];
                                $counter++;
                        }
                        $number++
                }

                $numofactual=@actual;   #gets number of actual data

                $number=0;      #resets variable for next loop
```

```
                $counter=0;     #resets variable for next loop

                while($number < $numofactual){    #gets all temperatures
                        if($actual[$number] =~ m/(\W)/) {
                                @temp[$counter] = $actual[$number];
                                $counter++;
                        }
                        $number++;
                }
                $numname=$number-$counter;      #gets spots of name

                $number=0;      #resets variable for next loop

                $wholename ="";                 #resets whole name
                while($number < $numname){      #gets name
                        $bladename=$actual[$number]." ";
                        $wholename = $wholename.$bladename;
                        $number++;
                }

                @names[$count-5]=$wholename;    #puts name in an array
                @temps[$count-5]=$temp[0];      #puts blade temp in
array

                $count++;               #increments count for next line
                }

        }

        if($temp[0] eq ""){             #check if temp found
                print "No temperature available for blade $blades at
this time.\n";

        }else{                          #prints name and temps
                $number=0;
                while($number<$count-5){
                print @names[$number]."\t".@temps[$number]."\n";
                $number++;
                }
        }

        print"\n";

        undef @names;                   #Clears all arrays
        undef @temps;
        undef @acutual;
        undef @temp;
        undef @words;
        undef @data;

        $numofblades--;                 #dec blade number for loop

   }
```

```
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl BTemp_host.pl –u user5

3.21.2 BTemp_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")
```

```
username = "None";  #checks if opt or not

opts, args = getopt.getopt(sys.argv[1:], "u:a", ["username="]) #sets up
opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')

print "Running this will return the temperature of all the blades for
the host.\n"

while host != "":   #loop until file empty

    name= username + "@" + host     #sets up ssh command

    data=[]
    data.append(os.popen("ssh " + name + " info").read())   #executes
info command

    number=re.search(r'\bBlade slots\b',data[0]).start()  #finds the
string Blade slots and gets number space where it starts

    newdata = data[0][number:]  #puts Blade slots and data after that in
a string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the number of Blade Slots

    blades=match.group(1)       #assigns that number to the variable
blades

    print "\nHost: " +host


    while blades > 0:
        print "\nBlade " +str(blades)+ " Temperature\n-----------------
--------------"

        data2=[]
        data2.append(os.popen("ssh " + name + " temps -T blade["
+str(blades)+ "]").read()) #executes get temperature command
```

```
        exitcodecheck=[]
        exitcodecheck.append(os.popen("echo $?").read());
        value=exitcodecheck[0];

        if int(value) != 0:     #Checks Exit Code of ssh command
            if int(value) == 75:
                print "Operation was not performed because the device or
MM was";
                print "not in the correct state.";
            if int(value) == 70:
                print "Internal Software Error";
            if int(value) == 64:
                print "Command Line Usage Error";
            if int(value) == 127:
                print "Command Not Found";
            if int(value) == 126:
                print "User does not have permission";
            if int(value) == 128:
                print "This value is strictly internal to AMM";

        line = data2[0].split('\n') #splits the lines of data from the
list command into an array called lines

        numoflines=len(line)        #gets number of lines of data

        temperature ="None"         #checks if no available temperature

        for count in range(5,numoflines):

            temperature =""         #resets temperature
            if line[count] != "":       #avoids white space lines
                words = line[count].split() #splits line into words

                numofwords=len(words)       #counts number of words

                for count2 in range(0,numofwords):  #checks to see if
word is non word aka number to find the temp number only
                    rules = re.compile(r'\W+')  #sets rule to find non
word
                    match = rules.search(words[count2]) #executes rule
on words

                    if str(match) == 'None':    #if rule not found
                        nothing=0
                    else:
                        number = count2     #when rule found assigns
that spot to var number, that spot is the first temp which is the temp
of the blade comp
                        break           #breaks out of loop because data
that was needed is found
                for count3 in range(0,number):      #gets temp component
name assigns to variable
                    temperature = temperature + words[count3] + " "
```

```
                temperature = temperature +"\t"+ words[number] + " "
#gets actual temp and assigns to variable
                print temperature   #prints temperautre

        if temperature == "None":   #if no temp
            print "No temperature available for blade " +str(blades)+ "
at this time."

        blades = int(blades)-1  #increments blades

    newdata=""  #resets variables
    blades =""
    number =""

    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n\n"
```

EXAMPLE

python BTemp_host.py –u user5

3.22 CreateKey

The CreateKey script creates a public/private RSA key pair. The system will prompt the user to enter the folder where the key pair should be saved. Note: This script will create two files – a private key, and a public key. The public key will have a .pub file extension, and this is the key to copy into your AMM login profile when adding a key.

3.22.1 CreateKey.sh

```
ssh-keygen
```

EXAMPLE

bash CreateKey.sh

3.22.2 CreateKey.pl

```
exec ("ssh-keygen")
```

EXAMPLE

perl CreateKey.pl

3.22.3 CreateKey.py

```
import os

os.system("ssh-keygen")
```

EXAMPLE

python CreateKey.py

3.23 ExternalPortDisable

The ExternalPortDisable script allows disabling of external ports for the specified I/O module.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-s	I/O module number to disable
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.23.1 ExternalPortDisable.sh

```
#!/bin/bash

user=
host=
switch=

while getopts “u:h:s:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         s)
             switch=$OPTARG
             ;;
     esac
done
```

```
if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $switch ]]
then
    echo "Disable external port to which switch?";
    read switch;
fi

name="$user@$host";

txt="ssh $name ifconfig -ep disabled -T switch[$switch]";

data="echo `$txt`";     #executes command


if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "Switch $switch on host $host has been disabled."
fi

if [[ "$data" =~ "Error" ]]
then
    echo "ERROR SWITCH NOT DISABLED!"
fi

if [[ "$data" =~ "empty" ]]
then
    echo "Target bay for switch $switch is empty."
fi

if [[ "$data" =~ "Invalid target path" ]]
then
    echo "Invalid target path."
fi

if [[ "$data" =~ "unavailable" ]]
then
    echo "The target bay is unavailable for this chasis type."
fi
```

EXAMPLE

SSH
bash ExternalPortDisable.sh –u user –h 9.37.133.212 -s 2

3.23.2 ExternalPortDisable.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements
        "user=s"  => \ $user,           #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "switch=i" => \ $switch,        #Switch Number
        "type=s" => \ $type,            #type either ssh or telnet
);


if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($switch == ""){                      #If Switch not opt
print "Disable external port to which switch?\n";
chomp($switch =<>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("ifconfig -ep disabled -T switch[$switch]");
#disables switch
        @data = $telnet->waitfor('/system> $/i');
```

```
        $telnet->print("exit");
}


if($type eq "ssh"){             #ssh
        $name = $user."@".$host;
        $command="ssh $name ifconfig -ep disabled -T switch[$switch]";
#disables switch
        @data= `$command`;
        checkexitcode();
}

$error="Error";                 #error check
$ok="OK";                       #successfully check
$invalid="Invalid target path"; #invalid check
$empty="empty";                 #empty check
$unavailable="unavailable";     #unavailable check
$authority="authority";         #authority check
$outputcheck=0;                 #checks if any if statements used

for(@data){             #checks output and tells whether successfully or
error

        if($_ =~ /$error/){
                print "\nERROR SWITCH NOT DISABLED!\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "\nSwitch $switch on host $host has been
disabled.";
                $outputcheck=1;
        }elsif($_ =~ /$invalid/){
                print "\nInvalid target path\n";
                $outputcheck=1;
        }elsif($_ =~ /$empty/){
                print "\nThe target bay is empty\n";
                $outputcheck=1;
        }elsif($_ =~ /$unavailable/){
                print "The target bay is unavailable for this chasis
type\n";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }
}


if($outputcheck == 0)
{
        print @data;
}
```

```
sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

Telnet
perl ExternalPortDisable.pl –u user –p pass5 –h 9.37.133.212 -s 1 –t telnet

SSH
perl ExternalPortDisable.pl –u user –h 9.37.133.212 -s 2 –t ssh

3.23.3 ExternalPortDisable.py

```
import telnetlib, os
import sys, getopt
import re
```

```
username = "None";  #checks if opt or not
password= "None";
host = "None";
switch = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:s:h:t:a",
["username=","password=","host=","switch=","type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-s", "--switch"):     #if -s opt assigns switch
number
            switch = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if switch == "None":
    switch = raw_input("Disable external port to which switch?\n")
    switch = switch.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telnet to disable port
```

```
    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("ifconfig -ep disabled -T switch[" + switch + "]\r")
    data = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data)) == 'None': #if statements for
output
        nothing=0
    else:
        print "The target bay is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data)) == 'None':
        nothing=0
    else:
        print "\nSwitch " +switch+ " on host " +host+ " has been
disabled."
        outputcheck=1;

    if str(re.search(r'\bError\b',data)) == 'None':
        nothing=0
    else:
        print "\NERROR SWITCH NOT DISABLED!"
        outputcheck=1;

    if str(re.search(r'\bInvalid target path\b',data)) == 'None':
        nothing=0
    else:
        print "\nInvalid target path."
        outputcheck=1;

    if str(re.search(r'\bunvailable\b',data)) == 'None':
        nothing=0
    else:
        print "\nThe target bay is unavailable for this chasis type"
        outputcheck=1;

    if outputcheck == 0:
        print data;

if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command


    data=[]
```

```
    data.append(os.popen("ssh " + name + " ifconfig -ep disabled -T
switch[" + switch + "]").read()) #executes disable switch command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data[0])) == 'None':  #if statements
for output
        nothing=0
    else:
        print "The target bay is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nSwitch " +switch+ " on host " +host+ " has been
disabled."
        outputcheck=1;

    if str(re.search(r'\bError\b',data[0])) == 'None':
        nothing=0
    else:
        print "\NERROR SWITCH NOT DisABLED!"
        outputcheck=1;

    if str(re.search(r'\bInvalid target path\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nInvalid target path."
        outputcheck=1;

    if str(re.search(r'\bunvailable\b',data[0])) == 'None':
        nothing=0
```

```
    else:
        print "\nThe target bay is unavailable for this chasis type"
        outputcheck=1;

    if outputcheck == 0:
        print data[0];
```

EXAMPLE

Telnet
python ExternalPortDisable.py –u user –p pass5 –h 9.37.133.212 -s 1 –t telnet

SSH
python ExternalPortDisable.py –u user –h 9.37.133.212 -s 2 –t ssh

3.24 ExternalPortEnable

The ExternalPortEnable script allows enabling of external ports for the specified I/O module.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-s	I/O module number to enable
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.24.1 ExternalPortEnable.sh

```
#!/bin/bash

user=
host=
switch=

while getopts “u:h:s:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
```

```
            ;;
        s)
            switch=$OPTARG
            ;;
    esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $switch ]]
then
    echo "Enable external port to which switch?";
    read switch;
fi

name="$user@$host";

txt="ssh $name ifconfig -ep enabled -T switch[$switch]";

data="echo `$txt`";     #executes command


if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "Switch $switch on host $host has been enabled."
fi

if [[ "$data" =~ "Error" ]]
then
    echo "ERROR SWITCH NOT ENABLED!"
fi

if [[ "$data" =~ "empty" ]]
then
    echo "Target bay for switch $switch is empty."
fi

if [[ "$data" =~ "Invalid target path" ]]
then
    echo "Invalid target path."
fi

if [[ "$data" =~ "unavailable" ]]
```

```
then
    echo "The target bay is unavailable for this chasis type."
fi
```

EXAMPLE

SSH
bash ExternalPortEnable.sh –u user –h 9.37.133.212 -s 2

3.24.2 ExternalPortEnable.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements
        "user=s"  => \ $user,           #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "switch=i" => \ $switch,        #Switch Number
        "type=s" => \ $type,            #type either ssh or telnet
);


if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($switch == ""){                      #If Switch not opt
print "Enable external port to which switch?\n";
chomp($switch =<>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
```

```
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("ifconfig -ep enabled -T switch[$switch]");
#enables switch
        @data = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");
}


if($type eq "ssh"){             #ssh
        $name = $user."@".$host;
        $command="ssh $name ifconfig -ep enabled -T switch[$switch]";
#enables switch
        @data= `$command`;
        checkexitcode();
}

$error="Error";                 #error check
$ok="OK";                       #successfully check
$invalid="Invalid target path"; #invalid check
$empty="empty";                 #empty check
$unavailable="unavailable";     #unavailable check
$authority="authority";         #authority check
$outputcheck=0;                 #checks if any if statements used

for(@data){             #checks output and tells whether successfully or
error

        if($_ =~ /$error/){
                print "\nERROR SWITCH NOT ENABLED!\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "\nSwitch $switch on host $host has been
enabled.";
                $outputcheck=1;
        }elsif($_ =~ /$invalid/){
                print "\nInvalid target path\n";
                $outputcheck=1;
        }elsif($_ =~ /$empty/){
                print "\nThe target bay is empty\n";
                $outputcheck=1;
        }elsif($_ =~ /$unavailable/){
                print "The target bay is unavailable for this chasis
type\n";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }
}
```

```
if($outputcheck == 0)
{
        print @data;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

Telnet
perl ExternalPortEnable.pl –u user –p pass5 –h 9.37.133.212 -s 1 –t telnet

SSH
perl ExternalPortEnable.pl –u user –h 9.37.133.212 -s 2 –t ssh

3.24.3 ExternalPortEnable.py

```
import telnetlib, os
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
switch = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:s:h:t:a",
["username=","password=","host=","switch=","type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-s", "--switch"):     #if -s opt assigns switch
number
            switch = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if switch == "None":
    switch = raw_input("Enable external port to which switch?\n")
    switch = switch.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
```

```
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telnet to enable port

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("ifconfig -ep enabled -T switch[" + switch + "]\r")
    data = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data)) == 'None': #if statements for
output
        nothing=0
    else:
        print "The target bay is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data)) == 'None':
        nothing=0
    else:
        print "\nSwitch " +switch+ " on host " +host+ " has been
enabled."
        outputcheck=1;

    if str(re.search(r'\bError\b',data)) == 'None':
        nothing=0
    else:
        print "\NERROR SWITCH NOT ENABLED!"
        outputcheck=1;

    if str(re.search(r'\bInvalid target path\b',data)) == 'None':
        nothing=0
    else:
        print "\nInvalid target path."
        outputcheck=1;

    if str(re.search(r'\bunvailable\b',data)) == 'None':
        nothing=0
    else:
        print "\nThe target bay is unavailable for this chasis type"
        outputcheck=1;

    if outputcheck == 0:
        print data;


if type == "ssh":       #ssh
```

```
    name= username + "@" + host     #sets up ssh command


    data=[]
    data.append(os.popen("ssh " + name + " ifconfig -ep enabled -T
switch[" + switch + "]").read()) #executes enable switch command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bempty\b',data[0])) == 'None':  #if statements
for output
        nothing=0
    else:
        print "The target bay is empty"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nSwitch " +switch+ " on host " +host+ " has been
enabled."
        outputcheck=1;

    if str(re.search(r'\bError\b',data[0])) == 'None':
        nothing=0
    else:
        print "\NERROR SWITCH NOT ENABLED!"
        outputcheck=1;

    if str(re.search(r'\bInvalid target path\b',data[0])) == 'None':
        nothing=0
    else:
```

```
        print "\nInvalid target path."
        outputcheck=1;

    if str(re.search(r'\bunvailable\b',data[0])) == 'None':
        nothing=0
    else:
        print "\nThe target bay is unavailable for this chasis type"
        outputcheck=1;

    if outputcheck == 0:
        print data[0];
```

EXAMPLE

Telnet
python ExternalPortEnable.py –u user –p pass5 –h 9.37.133.212 -s 1 –t telnet

SSH
python ExternalPortEnable.py –u user –h 9.37.133.212 -s 2 –t ssh

3.25 FirmwareUpdateMM

The FirmwareUpdateMM script flashes a .pkt file located at the provided URI to the specified advanced management module.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-d	URI of the packet file location (ex., tftp://192.178.50.125/user/fwUpdate.pkt)

3.25.1 FirmwareUpdateMM.sh

```
#!/bin/bash

user=
host=
directory=

while getopts “u:h:d:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
```

```
        h)
            host=$OPTARG
            ;;
        d)
            directory=$OPTARG
            ;;

    esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $directory ]]
then
    echo "Enter the URI of the packet file's location.";
    echo "For example : tftp://192.168.70.125/user1/file.pkt";
    read directory;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


txt="ssh $name update -u $directory -T mm[$mm]";

${txt};     #executes command
```

EXAMPLE

SSH
bash FirmwareUpdateMM.sh –u username –h 9.37.123.423 –d tftp://192.168.70.125/filename/file.pkt

3.25.2 FirmwareUpdateMM.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "directory=s" => \ $directory   #URI of pkt file
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;

$command ="ssh $name list -l a";        #excutes command
@data = `$command`;

$mm=findprimarymm(@data);


if($directory eq ""){                   #URI of pkt file
print "Enter the URI of the packet file's location.\n";
print "For example : tftp://192.168.70.125/user1/file.pkt\n";
chomp($directory =<>);
}

system("ssh $name update -u $directory -T mm[$mm]");
checkexitcode();

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }
```

```
        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
```

```
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl FirmwareUpdateMM.pl –u username –h 9.37.123.423 –d tftp://192.168.70.125/filename/file.pkt

3.25.3 FirmwareUpdateMM.py

```
import os
import sys, getopt
import re

username = "None"   #checks if opt or not
host = "None"
directory = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:d:a", ["username=",
"host=", "directory="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg
```

```
    if opt in ("-d", "--directory"):  #if -d opt assigns directory of
file to flash
            directory = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if directory == "None":
    directory= raw_input("\nEnter the URI of the packet file's
location.\nFor example: tftp://192.168.70.125/user1/file.pkt\n")
    directory= directory.rstrip('\n')

name= username + "@" + host     #sets up ssh command

data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";

number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

newdata = data[0][number-10:number] #gets the primary mm string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

mm=match.group(1)       #assigns that number to primary mm
```

```
os.system("ssh " + name + " update -u " +directory+ " -T mm[" +mm+"]")
#executes command for update
```

EXAMPLE

python FirmwareUpdateMM.py –u username –h 9.37.123.423 –d tftp://192.168.70.125/filename/file.pkt

3.26 FirmwareUpdateMM_host

The FirmwareUpdateMM_host script flashes a .pkt file located at the provided URI to the specified advanced management module.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-d	URI of the packet file location (ex., tftp://192.178.50.125/user/fwUpdate.pkt)

3.26.1 FirmwareUpdateMM_host.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "directory=s" => \ $directory   #URI of pkt file
);

$file = 'host.txt';             #filename containing hosts

open(File,$file) || die("Could not open file!");

$host = <File>;                 #reads host from file
chomp($host);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}

if($directory eq ""){                   #directory of file on tftp
server
print "Enter the URI of the packet file's location.\n";
print "For example : tftp://192.168.70.125/user1/file.pkt\n";
chomp($directory =<>);
```

```
}

while($host ne "")
{

        $name = $user."@".$host;

        print "Host: ".$host."\n";

        $command ="ssh $name list -l a";        #excutes command
        @data = `$command`;
        checkexitcode();

        $mm=findprimarymm(@data);


        system("ssh $name update -u $directory -T mm[$mm]");
        checkexitcode();

        $host =<File>;                  #reads host from file
        chomp($host);
}

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace
```

```
                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}



sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
```

```
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl FirmwareUpdateMM_host.pl –u username –d tftp://192.168.70.125/filename/file.pkt

3.26.2 FirmwareUpdateMM_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None"   #checks if opt or not
directory = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:d:a", ["username=",
"directory="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-d", "--directory"):  #if -d opt assigns directory of
file to flash
            directory = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if directory == "None":
    directory= raw_input("\nEnter the URI of the packet file's
location.\nFor example: tftp://192.168.70.125/aaron/file.pkt\n")
    directory= directory.rstrip('\n')

host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')

while host != "":   #loop until file empty
    name= username + "@" + host     #sets up ssh command
```

```
    print "Host: "+host;

    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    os.system("ssh " + name + " update -u " +directory+ " -T mm["
+mm+"]") #executes command for update

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n"
```

EXAMPLE

python FirmwareUpdateMM_host.py –u username –d tftp://192.168.70.125/filename/file.pkt

3.27 GetInventory

The GetInventory script displays the device inventory for the specified host along with the device serial numbers.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname

3.27.1 GetInventory.sh

```
#!/bin/bash

user=
host=


while getopts “u:h:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi


name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command

declare -a command;     #creates arrays
```

```
declare -a words;
num=0;                  #loop control counter for arrays

while read line         #reads each line of the file data.txt which has
the command output from above
do
    for word in ${line[@]}; do          #gets the words from each line
    words[$num]=$word;                  #sets word to array words
    command[$num]="ssh $name info -T $word";    #sets command to array
command
    let "num += 1";
    done
done < "data.txt"

`rm data.txt`;      #removes data file

number=${#command[@]};  #gets the number of commands
control=0;              #loop control counter

while [ "$number" -gt $control ]   #loops thru number of commands
do
    `${command[control]}>output.txt`;       #executes command

    line=`grep '\<FRU serial no\>' output.txt`;   # finds serial numbers
    line2=`grep '\<Mach serial number\>' output.txt`;

    `rm output.txt`;

   if [[ $line == "" && $line2 == "" ]]     #checks if nothin found
   then
        nothin=0;
   else
        echo
        echo ${words[$control]};    #prints serials and inventory name
        echo $line2;
        echo $line;
    fi

    let "control += 1";             #increments loop counter
done
```

EXAMPLE

SSH
bash GetInventory.sh –u user –h 9.37.133.212

3.27.2 GetInventory.pl

```
use Getopt::Long;

$result = GetOptions(
```

```
        "username=s"  => \ $user,
        "host=s"  => \ $host,
);

if($user eq ""){
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){
print "Enter host name.\n";
chomp($host = <>);
}

$find ="FRU serial no.";                #variables for serial find
$find2 ="Mach serial number";

$name = $user."@".$host;

print "\nHost: ".$host."\n\n";
print "Inventory\t\t\t\tSerial Numbers";
print "\n-----------------------------------------------------------\n";

$command ="ssh $name list -l a";        #executes command
@data = `$command`;
checkexitcode();

$count = 3;             #starts where data starts
for(@data){

        @words=split(/ /, @data[$count]);       #splits each line into
words
        $numofwords=@words;             #counts number of words

        if($numofwords ne 0){           #avoids white space

        $command="";
        $command ="ssh $name info -T @words[0]";  #excutes the info
command on all inventory to get serial number
        @data2 = `$command`;
        checkexitcode();

        for(@data2){                            #if serial found set it
to serial variable
                if($_ =~ /$find/){
                        $FRUserial = "$_";
                }
        }

        for(@data2){
                if($_ =~ /$find2/){
                        print trim(@data[$count]);     #prints name and
serial number for inventory
                        $Machserial = "$_";
```

```
                }
        }

        print "\t\t".$Machserial."\t\t\t\t".$FRUserial; #prints Mach
serial and FRU serial
        $count++;
        print "\n";
        }

        undef @data2;           #resets variables
        $Machserial = "";
        $FRUserial = "";
}

sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
```

```
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl GetInventory.pl –u user –h 9.37.133.212

3.27.3 GetInventory.py

```
import os
import sys, getopt
import re

username = "None";  #checks if opt or not
host = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')


name= username + "@" + host     #sets up ssh command

data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

line = data[0].split('\n')  #splits the lines of data from the list
command into an array called lines
```

```
numoflines=len(line)        #gets number of lines of data

for count in range(3,numoflines):   #loop goes from first needed data to
the end of the number of lines

    if line[count] != "":       #avoids empty lines of white space
        words = line[count].split() #splits each line into words


        data2=[]
        try:
            data2.append(os.popen("ssh " + name + " info -T " +
words[0]).read())   #words[0] is first word for next command to execute

            exitcodecheck=[]
            exitcodecheck.append(os.popen("echo $?").read());
            value=exitcodecheck[0];

            if int(value) != 0:     #Checks Exit Code of ssh command
                if int(value) == 75:
                    print "Operation was not performed because the
device or MM was";
                    print "not in the correct state.";
                if int(value) == 70:
                    print "Internal Software Error";
                if int(value) == 64:
                    print "Command Line Usage Error";
                if int(value) == 127:
                    print "Command Not Found";
                if int(value) == 126:
                    print "User does not have permission";
                if int(value) == 128:
                    print "This value is strictly internal to AMM";

            line2 = data2[0].split('\n')  #splits data2 at the new line
char to break data into lines

            numoflines2=len(line2)  # gets number of lines created from
data2

            for count2 in range(0,numoflines2): #searches each line for
the serial numbers

                if str(re.search(r'\bMach serial
number\b',line2[count2])) == 'None':
                    nothing=0
                else:
                    print words[0]      #print inventory name
                    print line2[count2] #prints Mach serial
                if str(re.search(r'\bFRU serial\b',line2[count2])) ==
'None':
                    nothing=0
                else:
```

```
                print line2[count2] #prints FRU serial
            print "\n"

        except:
            nothing=0
```

EXAMPLE

python GetInventory.py –u user –h 9.37.133.212

3.28 GetInventory_host

The GetInventory_host script displays the device inventory for all of the hosts listed in the host.txt file, along with the serial numbers for all devices.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username

3.28.1 GetInventory_host.pl

```
$file = 'host.txt';

open(File,$file) || die("Could not open file!");

use Getopt::Long;

$result = GetOptions(

        "username=s"  => \ $user,
);

if($user eq ""){
print "Enter username.\n";
chomp($user= <>);
}

$host = <File>;
chomp($host);

$find ="FRU serial no.";                #variables for serial find
$find2 ="Mach serial number";

while($host ne "")
{

        $name = $user."@".$host;
```

```
        print "\nHost: ".$host."\n\n";
        print "Inventory\t\t\t\tSerial Numbers";
        print "\n-------------------------------------------------------------
----\n";

        $command ="ssh $name list -l a";        #executes command
        @data = `$command`;
        checkexitcode();

        $count = 3;             #starts where data starts
        for(@data){

                @words=split(/ /, @data[$count]);       #splits each
line into words
                $numofwords=@words;             #counts number of words

                if($numofwords ne 0){           #avoids white space

                $command="";
                $command ="ssh $name info -T @words[0]";  #excutes the
info command on all inventory to get serial number
                @data2 = `$command`;
                checkexitcode();

                for(@data2){                            #if serial found
set it to serial variable
                        if($_ =~ /$find/){
                                $FRUserial = "$_";
                        }
                }

                for(@data2){
                        if($_ =~ /$find2/){
                                $Machserial = "$_";
                        }
                }
                print trim(@data[$count]);              #prints name and
serial number for inventory
                print "\t\t".$Machserial."\t\t\t\t".$FRUserial;
                $count++;
                print "\n";
                }
        }


        undef @data;
        undef @data2;

        $host =<File>;
        chomp($host);
}
```

```
sub trim($)             #trims white space on bladename
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl GetInventory_host.pl –u username

3.28.2 GetInventory_host.py

```
import os
import sys, getopt
import re

fin = open("host.txt","r")

username = "None";  #checks if opt or not

opts, args = getopt.getopt(sys.argv[1:], "u:a", ["username="]) #sets up
opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg


if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')


host= fin.readline()        #reads host from a file
host= host.rstrip('\r\n')

while host != "":   #loop until file empty

    name= username + "@" + host     #sets up ssh command

    print "\nHost: " +host
    print "\n------------------------------------------------------------
\n"


    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    line = data[0].split('\n')  #splits the lines of data from the list
command into an array called lines


    numoflines=len(line)        #gets number of lines of data

    for count in range(3,numoflines):   #loop goes from first needed
data to the end of the number of lines

        if line[count] != "":       #avoids empty lines of white space
            words = line[count].split() #splits each line into words


            data2=[]
```

```
            try:
                data2.append(os.popen("ssh " + name + " info -T " +
words[0]).read())   #words[0] is first word for next command to execute

                exitcodecheck=[]
                exitcodecheck.append(os.popen("echo $?").read());
                value=exitcodecheck[0];

                if int(value) != 0:     #Checks Exit Code of ssh command
                    if int(value) == 75:
                        print "Operation was not performed because the
device or MM was";
                        print "not in the correct state.";
                    if int(value) == 70:
                        print "Internal Software Error";
                    if int(value) == 64:
                        print "Command Line Usage Error";
                    if int(value) == 127:
                        print "Command Not Found";
                    if int(value) == 126:
                        print "User does not have permission";
                    if int(value) == 128:
                        print "This value is strictly internal to AMM";

                line2 = data2[0].split('\n')  #splits data2 at the new
line char to break data into lines

                numoflines2=len(line2)  # gets number of lines created
from data2

                for count2 in range(0,numoflines2): #searches each line
for the serial numbers

                    if str(re.search(r'\bMach serial
number\b',line2[count2])) == 'None':
                        nothing=0
                    else:
                        print words[0]      #print inventory name
                        print line2[count2] #prints Mach serial
                    if str(re.search(r'\bFRU serial\b',line2[count2]))
== 'None':
                        nothing=0
                    else:
                        print line2[count2] #prints FRU serial
                print "\n"

            except:
                nothing=0

    host= fin.readline()        #reads host from file
    host= host.rstrip('\r\n')
    print "\n\n"
```

EXAMPLE

python GetInventory_host.py –u username

3.29 ReadConfig

The ReadConfig script downloads the configuration file from a TFTP server to the specified host.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-i	IP address of TFTP server
-d	Directory of file on TFTP server

3.29.1 ReadConfig.sh

```
#!/bin/bash

user=
host=
ip=
directory=

while getopts “u:h:i:d:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         i)
             ip=$OPTARG
             ;;
         d)
             directory=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
```

```
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $ip ]]
then
    echo "Enter the ip address of the tftp server.";
    read ip;
fi

if [[ -z $directory ]]
then
    echo "Enter the directory of the tftp configuration file to";
    echo "read that file to the host $host.";
    echo "For example : /aaron/file.cfg";
    read directory;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


txt="ssh $name read -config file -i $ip -l $directory -T mm[$mm]";

data="echo `$txt`";     #executes command

echo $data;
```

EXAMPLE

SSH
bash ReadConfig.sh –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg

3.29.2 ReadConfig.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "ip=s" =>  \ $ip,               #Ip address of tftp server
        "directory=s" => \ $directory   #directory of file on tftp
server
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;

$command ="ssh $name list -l a";        #excutes command
@data = `$command`;
checkexitcode();

$mm=findprimarymm(@data);


if($ip eq ""){                  #If Ip address of tftp server not opt
print "Enter the ip address of the tftp server.\n";
chomp($ip =<>);
}

if($directory eq ""){                   #directory of file on tftp
server
print "Enter the directory of the tftp configuration file to read\n that
file to the host $host.";
print "\nFor example : /aaron/file.cfg\n";
chomp($directory =<>);
}

system("ssh $name read -config file -l $directory -i $ip -T mm[$mm]");
checkexitcode();

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
```

```
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
```

```
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl ReadConfig.pl –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg

3.29.3 ReadConfig.py

```
import os
import sys, getopt
import re

username = "None"   #checks if opt or not
host = "None"
ip = "None"
directory = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:i:d:a", ["username=",
"host=", "ip=", "directory="]) #sets up opt


for opt, arg in opts:
```

```
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

    if opt in ("-i", "--ip"):     #if -i opt assigns ip address of tftp
server
            ip = arg

    if opt in ("-d", "--directory"):  #if -d opt assigns directory of
file to flash
            directory = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if ip == "None":
    ip= raw_input("\nEnter the ip address of the tftp server.\n")
    ip= ip.rstrip('\n')

if directory == "None":
    directory= raw_input("\nEnter the directory of the tftp file to read
the configuration file\nonto " +host+ ".\nFor example:
/aaron/file.cfg\n")
    directory= directory.rstrip('\n')

name= username + "@" + host     #sets up ssh command

data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

newdata = data[0][number-10:number] #gets the primary mm string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

mm=match.group(1)       #assigns that number to primary mm


os.system("ssh " + name + " read -config file -i " +ip+ " -l "
+directory+ " -T mm[" +mm+"]") #executes command for read config
```

```
exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";
```

EXAMPLE

python ReadConfig.py –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg

3.30 RemoveKey

The RemoveKey script allows removal of an SSH key from the specified user profile. You must know the login id number of the user profile to remove a key from that profile. You must also know the place number of the key. If there is only one key, the default place number is 1.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-l	User Login ID of user profile to remove key from
-k	Place of key
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.30.1 RemoveKey.sh

```
#!/bin/bash
```

```
user=
host=
userid=
keyplace=

while getopts “u:h:l:k:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         l)
             userid=$OPTARG
             ;;
         k)
             keyplace=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $userid ]]
then
    echo "Remove a key to which Login ID Number?";
    read userid;
fi

if [[ -z $keyplace ]]
then
    echo "Which Place is the key in?";
    echo "If only one key the default place is 1.";
    read keyplace;
fi

name="$user@$host";

command="ssh $name list -l a";
```

```
`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


command="ssh $name users -$userid -pk -$keyplace -remove -T mm[$mm]";

data="echo `$command`";         #executes command to remove user

if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "Key removed from Login ID $userid successfully."
else
    echo $data;
fi
```

EXAMPLE

SSH
bash RemoveKey.sh –u user –h 9.37.133.212 –l 5 –p 1

3.30.2 RemoveKey.pl

```
use Getopt::Long;

$result = GetOptions(                   #Defines Get Opt Statements
        "user=s"  => \ $user,           #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "loginid=i" => \ $userid,       #User Login ID
        "keyplace=i" => \ $keyplace,    #Keys Place Number
        "type=s" => \ $type,  #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($userid eq ""){                      #If userid not opt
```

```
print "Remove a key from which Login ID Number?\n";
chomp($userid =<>);
}
if($keyplace eq ""){                    #If place not opt
print "Which Place is the key in?\nIf only one key the default place is
1.\n";
chomp($keyplace =<>);
}

if($type eq "telnet"){          #telnet
        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("list -l a");
        @data = $telnet->waitfor('/system> $/i');

        $mm=findprimarymmtelnet(@data); #finds primary mm

        undef @data;    #clears array data

        $telnet->print("users -$userid -pk -$keyplace -remove -T
mm[$mm]"); #removes key
        @output = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");

        pop(@output);

        print @output;
}

if($type eq "ssh"){             #ssh
        $name = $user."@".$host;

        $command ="ssh $name list -l a";
        @data = `$command`;

        $mm = findprimarymm(@data);     #finds primary mm

        undef @data;                    #clears variables
        $command ="";
```

```
        $command ="ssh $name users -$userid -pk -$keyplace -remove -T
mm[$mm]"; #removes key
        @data = `$command`;
        checkexitcode();

        print @data;
}

sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}


sub findprimarymmtelnet(@data)  #finds primary mm
{
```

```
        @data2 = split('\n',$data[0]);  #splits data into lines and puts
into data2
        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data2){                    #finds line that has mm in it
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $find = "primary";      #String to find
        $count=0;               #loop counter
        for(@mm){                       #finds line that has primary in
it
                if($_ =~ /$find/){
                        @primarymm[$count] = "$_";
                        $count++;
                }
        }
        for(@primarymm){        #gets primary mm number
                if(@primarymm[0] =~ m/(\d+)/){
                        $baystatus = $1;
                }
        }

        return $baystatus;      #return primary mm
}

sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
```

```
                print "Internal Software Error";
        }
        if($value == 64)
        {
                print "Command Line Usage Error";
        }
        if($value == 127)
        {
                print "Command Not Found";
        }
        if($value == 126)
        {
                print "User does not have permission";
        }
        if($value == 128)
        {
                print "This value is strictly internal to AMM";
        }

    }

}
```

EXAMPLE

Telnet
perl RemoveKey.pl –u user –p pass5 –h 9.37.133.212 –l 8 –p 1 –t telnet

SSH
perl RemoveKey.pl –u user –h 9.37.133.212 –l 5 –p 1 –t ssh

3.30.3 RemoveKey.py

```
import telnetlib, os
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
userid = "None";
keyplace = "None";
type = "None";
check = "false";    #checks if there is an if statement for the output,
if not prints output to screen

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:l:k:t:a",
["username=","password=","host=","userid=","keyplace=","type="]) #sets
up opt


for opt, arg in opts:
```

```
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

    if opt in ("-l", "--userid"):     #if -l opt assigns userid
            userid = arg

    if opt in ("-k", "--keyplace"):   #if -k opt assigns keyplace
            keyplace = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if userid == "None":
    userid= raw_input("Remove a key from which Login ID number?\n")
    userid= userid.rstrip('\n')

if keyplace == "None":
    keyplace= raw_input("Which Place is the key in?\nIf only one key the
default place is 1.\n")
    keyplace= keyplace.rstrip('\n')



if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telent to find mm

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
```

```
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("list -l a\r")
    data = telnet.read_until('system>',timeout=10)

    number=re.search(r'\bprimary\b',data).start()  #finds the string
primary and gets number space where it starts

    newdata = data[number-10:number]    #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    telnet.write("users -" +userid+ " -pk -" +keyplace+ " -remove -T
mm[" +mm+"]\r")  #removes key from profile
    data2 = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    if str(re.search(r'\bError\b',data2)) == 'None':    #if statements
for output
        nothing=0
    else:
        print "\nError adding key to userid " +userid+ " profile."
        check = "true";

    if str(re.search(r'\buser defined\b',data2)) == 'None':
        nothing=0
    else:
        print "\nThere is no user defined at userid " +userid+ "."
        check = "true";

    if str(re.search(r'\bno key installed\b',data2)) == 'None':
        nothing=0
    else:
        print "\nThere is no key installed at userid " +userid+ " place
" +keyplace+"."
        check = "true";

    if check == "false":
        print data2;


if type == "ssh":       #ssh


    name= username + "@" + host     #sets up ssh command

    data=[]
```

```
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    data2=[]
    data2.append(os.popen("ssh " + name + " users -" +userid+ " -pk -"
+keyplace+ " -remove -T mm[" +mm+"]").read()) #executes command to
remove key

    if str(re.search(r'\bError\b',data2[0])) == 'None': #if statements
for output
        nothing=0
    else:
        print "\nError adding key to userid " +userid+ " profile."
        check = "true";

    if str(re.search(r'\buser defined\b',data2[0])) == 'None':
        nothing=0
    else:
        print "\nThere is no user defined at userid " +userid+ "."
        check = "true";

    if str(re.search(r'\bno key installed\b',data2[0])) == 'None':
```

```
        nothing=0
    else:
        print "\nThere is no key installed at userid " +userid+ " place
" +keyplace+"."
        check = "true";

    if check == "false":
        print data2[0];
```

EXAMPLE

Telnet
python RemoveKey.py –u user –p pass5 –h 9.37.133.212 –l 8 –p 1 –t telnet

SSH
python RemoveKey.py –u user –h 9.37.133.212 –l 5 –p 1 –t ssh

3.31 RemoveUser

The RemoveUser script removes a user login profile from the specified host. You must know a user login id number to remove the profile from the host.

Option	Description
-u	Username
-p	Password, used only when connecting via telnet
-h	Hostname
-l	User Login ID number of user to remove
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.31.1 RemoveUser.sh

```
#!/bin/bash

user=
host=
userid=

while getopts “u:h:l:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
```

```
        h)
            host=$OPTARG
            ;;
        l)
            userid=$OPTARG
            ;;
    esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $userid ]]
then
    echo "Type the login id of the user you would like to remove.";
    read userid;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


command="ssh $name users -$userid -clear -T mm[$mm]";

data="echo `$command`";         #executes command to remove user


if [[ "$data" =~ "Error" ]]       #checks and prints output
then
    echo "ERROR PROFILE NOT REMOVED, MAKE SURE YOU HAVE PROPER STATUS TO
REMOVE THIS PROFILE!";
fi

if [[ "$data" =~ "OK" ]]
```

```
then
    echo "Login ID: $userid removed successfully from host $host.";
fi
```

EXAMPLE

SSH
bash RemoveUser.sh –u user –h 9.37.133.212 –l 4

3.31.2 RemoveUser.pl

```
use Getopt::Long;

$result = GetOptions(                    #Defines Get Opt Statements
        "user=s"  => \ $user,            #Username
        "password=s" => \ $Password,     #Password
        "host=s"  => \ $host,            #Host Name
        "loginid=i" => \ $userid,        #User Login ID
        "type=s" => \ $type,             #type either ssh or telnet
);

if($type eq ""){                         #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                         #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                         #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}
if($userid eq ""){                       #If userid not opt
print "Type the login id of the user you would like to remove.\n";
chomp($userid = <>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }

        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
```

```
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("list -l a");
        @data = $telnet->waitfor('/system> $/i');

        $mm=findprimarymmtelnet(@data);         #finds primary mm

        undef @data;                            #clears array data

        $telnet->print("users -$userid -clear -T mm[$mm]");     #removes
user
        @data = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");
}

if($type eq "ssh"){             #ssh

        $name = $user."@".$host;

        $command ="ssh $name list -l a";
        @data = `$command`;
        checkexitcode();

        $mm = findprimarymm(@data);             #finds primary mm
        undef @data;
        $command="";

        $command = "ssh $name users -$userid -clear -T mm[$mm]";
#removes user
        @data = `$command`;
        checkexitcode();
}



$error="Error";         #error check
$ok="OK";               #successfully check
$authority="authority"; #authority check
$outputcheck=0;                 #checks if any if statements used

for(@data){             #checks output and tells whether empty or turned
off successfully

        if($_ =~ /$error/){
                print "\nERROR PROFILE NOT REMOVED, MAKE SURE YOU HAVE
PROPER STATUS TO REMOVE THIS PROFILE!\n";
                $outputcheck=1;
        }elsif($_ =~ /$ok/){
                print "\nLogin ID $userid on $host was removed
successfully.";
                $outputcheck=1;
        }elsif($_ =~ /$authority/){
```

```
                print "\nUser does not have the authority to issue this
command.";
                $outputcheck=1;
        }
}

if($outputcheck == 0)
{
        print @data;
}

sub findprimarymmtelnet(@data)  #finds primary mm
{
        @data2 = split('\n',$data[0]);  #splits data into lines and puts
into data2
        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data2){                    #finds line that has mm in it
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $find = "primary";      #String to find
        $count=0;               #loop counter
        for(@mm){                       #finds line that has primary in
it
                if($_ =~ /$find/){
                        @primarymm[$count] = "$_";
                        $count++;
                }
        }
        for(@primarymm){        #gets primary mm number
                if(@primarymm[0] =~ m/(\d+)/){
                                $baystatus = $1;
                        }
        }

        return $baystatus;      #return primary mm
}


sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
```

```
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
```

```
                print "\nnot in the correct state.";
        }
        if($value == 70)
        {
                print "Internal Software Error";
        }
        if($value == 64)
        {
                print "Command Line Usage Error";
        }
        if($value == 127)
        {
                print "Command Not Found";
        }
        if($value == 126)
        {
                print "User does not have permission";
        }
        if($value == 128)
        {
                print "This value is strictly internal to AMM";
        }

    }

}
```

EXAMPLE

Telnet
perl RemoveUser.pl –u user –p pass5 –h 9.37.133.212 –l 5 –t telnet

SSH
perl RemoveUser.pl –u user –h 9.37.133.212 –l 4 –t ssh

3.31.3 RemoveUser.py

```
import telnetlib, os
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
userid = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:l:t:a",
["username=","password=","host=","userid=","type="]) #sets up opt


for opt, arg in opts:
```

```
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

    if opt in ("-l", "--userid"):     #if -h opt assigns host
            userid = arg

    if opt in ("-t", "--type"):   #if -t opt assigns type, type either
ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if userid == "None":
    userid= raw_input("Type the login id of the user you would like to
remove.\n")
    userid= userid.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telent to find mm

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("list -l a\r")
    data = telnet.read_until('system>',timeout=10)


    number=re.search(r'\bprimary\b',data).start()  #finds the string
primary and gets number space where it starts
```

```
    newdata = data[number-10:number]    #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm


    telnet.write(" users -" +userid+ " -clear -T mm[" +mm+"]\r")
#removes profile
    data2 = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bError\b',data2)) == 'None':    #if statements
for output
        nothing=0
    else:
        print "ERROR PROFILE NOT REMOVED, MAKE SURE YOU HAVE PROPER
STATUS TO REMOVE THIS PROFILE!"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data2)) == 'None':
        nothing=0
    else:
        print "Login ID: " +userid+ "\tHost: " +host+ "\nRemoved
Successfully"
        outputcheck=1;

    if outputcheck == 0:
        print data2;


if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number
```

```
    mm=match.group(1)       #assigns that number to primary mm

    if userid == "None":
        userid= raw_input("Type the login id of the user you would like
to remove.\n")
        userid= userid.rstrip('\n')

    data2=[]
    data2.append(os.popen("ssh " + name + " users -" +userid+ " -clear -
T mm[" +mm+"]").read()) #executes list command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";

    outputcheck=0;      #checks if any if statements used

    if str(re.search(r'\bError\b',data2[0])) == 'None': #if statements
for output
        nothing=0
    else:
        print "ERROR PROFILE NOT REMOVED, MAKE SURE YOU HAVE PROPER
STATUS TO REMOVE THIS PROFILE!"
        outputcheck=1;

    if str(re.search(r'\bOK\b',data2[0])) == 'None':
        nothing=0
    else:
        print "Login ID: " +userid+ "\tHost: " +host+ "\nRemoved
Successfully"
        outputcheck=1;

    if outputcheck == 0:
        print data2[0];
```

EXAMPLE

Telnet
python RemoveUser.py –u user –p pass5 –h 9.37.133.212 –l 5 –t telnet

SSH
python RemoveUser.py –u user –h 9.37.133.212 –l 4 –t ssh

3.32 ResetMM

The ResetMM script resets the specified advanced management module.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname

3.32.1 ResetMM.sh

```
#!/bin/bash

user=
host=


while getopts “u:h:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
```

```
fi


name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm

echo "Reseting......please wait.";

command="ssh $name reset -T mm[$mm]";

"`$command`";         #executes command to add user
```

EXAMPLE

SSH
bash ResetMM.sh –u username –h 9.37.123.423

3.32.2 ResetMM.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;

$command ="ssh $name list -l a";        #excutes command
@data = `$command`;
```

```
$mm=findprimarymm(@data);

print "Reseting.............please wait.\n";

system("ssh $name reset -T mm[$mm]");
checkexitcode();

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
```

```
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\not in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl ResetMM.pl –u username –h 9.37.123.423

3.32.3 ResetMM.py

```
import os
```

```
import sys, getopt
import re

username = "None"   #checks if opt or not
host = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:a", ["username=",
"host="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

name= username + "@" + host     #sets up ssh command

data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

newdata = data[0][number-10:number] #gets the primary mm string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

mm=match.group(1)       #assigns that number to primary mm


os.system("ssh " + name + " reset -T mm[" +mm+"]") #executes command for
mm reset

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
```

```
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";
```

EXAMPLE

python ResetMM.py –u username –h 9.37.123.423

3.33 ServiceData

The ServiceData script copies the service data to a file on a TFTP server so that the data can be viewed.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-i	IP address of TFTP server
-d	Path to file on TFTP server; this should have a .tgz extension

3.33.1 ServiceData.sh

```
#!/bin/bash

user=
host=
ip=
directory=

while getopts “u:h:i:d:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
```

```
            ;;
        h)
            host=$OPTARG
            ;;
        i)
            ip=$OPTARG
            ;;
        d)
            directory=$OPTARG
            ;;

    esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $ip ]]
then
    echo "Enter the ip address of the tftp server.";
    read ip;
fi

if [[ -z $directory ]]
then
    echo "Enter the directory of the tftp file where you would like the
service data to be saved.";
    echo "For example : /aaron/file.tgz";
    read directory;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm
```

```
txt="ssh $name displaysd -save $directory -i $ip -T mm[$mm]";

data="echo `$txt`";     #executes command

if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "File saved successfully to $directory on $ip."
else
    echo $data;
fi
```

EXAMPLE

SSH
bash ServiceData.sh –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.tgz

3.33.2 ServiceData.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "ip=s" =>  \ $ip,               #Ip address of tftp server
        "directory=s" => \ $directory   #directory of file on tftp
server
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;

$command ="ssh $name list -l a";        #excutes command
@data = `$command`;
checkexitcode();

$mm=findprimarymm(@data);


if($ip eq ""){                  #If Ip address of tftp server not opt
print "Enter the ip address of the tftp server.\n";
chomp($ip =<>);
```

```
}

if($directory eq ""){                   #directory of file on tftp
server
print "Enter the directory of the tftp file where you would like the
service data to be saved.";
print "For example : /aaron/file.tgz\n";
chomp($directory =<>);
}

system("ssh $name displaysd -save $directory -i $ip -T mm[$mm]");
checkexitcode();

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
```

```
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl ServiceData.pl –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.tgz

3.33.3 ServiceData.py

```
import os
import sys, getopt
import re

username = "None"   #checks if opt or not
host = "None"
ip = "None"
directory = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:i:d:a", ["username=",
"host=", "ip=", "directory="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

    if opt in ("-i", "--ip"):     #if -i opt assigns ip address of tftp
server
            ip = arg

    if opt in ("-d", "--directory"):  #if -d opt assigns directory of
file to flash
            directory = arg

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if ip == "None":
    ip= raw_input("\nEnter the ip address of the tftp server.\n")
    ip= ip.rstrip('\n')

if directory == "None":
    directory= raw_input("\nEnter the directory of the tftp file where
you would like the service data to be saved.\nFor example:
/aaron/file.tgz\n")
    directory= directory.rstrip('\n')

name= username + "@" + host     #sets up ssh command
```

```
data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

newdata = data[0][number-10:number] #gets the primary mm string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

mm=match.group(1)       #assigns that number to primary mm


os.system("ssh " + name + " displaysd -save " +directory+ " -i " +ip+ "
-T mm[" +mm+"]") #executes command for service data

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
    if int(value) == 126:
        print "User does not have permission";
    if int(value) == 128:
        print "This value is strictly internal to AMM";
```

EXAMPLE

python ServiceData.py –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.tgz

3.34 ViewUsers

The ViewUsers script displays all user profiles on the specified host.

Option	Description
-u	Username

-p	Password, used only when connecting via telnet
-h	Hostname
-t	Type of session (ssh or telnet), for Perl and Python scripts only

3.34.1 ViewUsers.sh

```
#!/bin/bash

user=
host=


while getopts “u:h:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi


name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file
```

```
mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


command="ssh $name users -T mm[$mm]";

`$command > data.txt`;         #executes command to add user

while read line         #reads each line of the file data.txt which has
the command output from above
do
    echo $line;

done < "data.txt"

`rm data.txt`;      #removes data file
```

EXAMPLE

SSH
bash ViewUsers.sh –u user –h 9.37.133.212

3.34.2 ViewUsers.pl

```
use Getopt::Long;

$result = GetOptions(                   #Defines Get Opt Statements
        "user=s"  => \ $user,           #Username
        "password=s" => \ $Password,    #Password
        "host=s"  => \ $host,           #Host Name
        "type=s" => \ $type,            #type either ssh or telnet
);

if($type eq ""){                        #If type not opt
print "Use ssh or telnet?\n";
chomp($type=<>);
}
if($user eq ""){                        #If Username not opt
print "Enter username.\n";
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}

if($type eq "telnet"){          #telnet

        if($Password eq ""){                    #If Password not opt
        print "Enter password.\n";
        chomp($Password=<>);
        }
```

```
        use Net::Telnet;
        $telnet = new Net::Telnet ( Timeout=>10,
        Errmode=>'die');
        $telnet->open("$host");
        $telnet->waitfor('/username: $/i');
        $telnet->print("$user");
        $telnet->waitfor('/password: $/i');
        $telnet->print("$Password");
        $telnet->waitfor('/system> $/i');
        $telnet->print("list -l a");
        @data = $telnet->waitfor('/system> $/i');

        $mm=findprimarymmtelnet(@data); #finds primary mm

        undef @data;    #clears array data

        $telnet->print("users -T mm[$mm]");             #excutes find
users
        @output = $telnet->waitfor('/system> $/i');
        $telnet->print("exit");

        pop(@output);
        print @output;
}

if($type eq "ssh"){             #ssh

        $name = $user."@".$host;

        $command ="ssh $name list -l a";
        @data = `$command`;
        checkexitcode();

        $mm = findprimarymm(@data);     #finds primary mm

        undef @data;
        $command ="";

        system("ssh $name users -T mm[$mm]");   #prints users
        checkexitcode();
}

sub findprimarymm(@data)        #finds primary mm
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
```

```
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;      #return primary mm
}


sub findprimarymmtelnet(@data)  #finds primary mm
{
        @data2 = split('\n',$data[0]);  #splits data into lines and puts
into data2
        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data2){                    #finds line that has mm in it
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $find = "primary";      #String to find
        $count=0;               #loop counter
        for(@mm){                       #finds line that has primary in
it
                if($_ =~ /$find/){
                        @primarymm[$count] = "$_";
                        $count++;
```

```
                }
        }
        for(@primarymm){        #gets primary mm number
                if(@primarymm[0] =~ m/(\d+)/){
                                $baystatus = $1;
                        }
        }

        return $baystatus;      #return primary mm
}

sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }
```

```
        }

}
```

EXAMPLE

Telnet
perl ViewUsers.pl –u user –p pass5 –h 9.37.133.212 –t telnet

SSH
perl ViewUsers.pl –u user –h 9.37.133.212 –t ssh

3.34.3 ViewUsers.py

```
import telnetlib, os
import sys, getopt
import re

username = "None";  #checks if opt or not
password= "None";
host = "None";
type = "None";

opts, args = getopt.getopt(sys.argv[1:], "u:p:h:t:a",
["username=","password=","host=","type="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

    if opt in ("-p", "--password"):   #if -p opt assigns username
            password = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

        if opt in ("-t", "--type"):   #if -t opt assigns type, type
either ssh or telnet
            type = arg

if type == "None":
    type = raw_input("Use ssh or telnet?\n")
    type = type.rstrip('\n')

if username == "None":          #if not opt then ask for user input
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
```

```
    host= host.rstrip('\n')


if type == "telnet":        #telnet

    if password == "None":
        password= raw_input("Enter password.\n")
        password= password.rstrip('\n')

    telnet = telnetlib.Telnet(host)     #telent to find mm

    telnet.read_until('username:',timeout=10)
    telnet.write(username+"\r")
    telnet.read_until('password:',timeout=10)
    telnet.write(password+"\r")
    telnet.read_until('system>',timeout=10)
    telnet.write("list -l a\r")
    data = telnet.read_until('system>',timeout=10)

    number=re.search(r'\bprimary\b',data).start()  #finds the string
primary and gets number space where it starts

    newdata = data[number-10:number]    #gets the primary mm string

    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm


    telnet.write("users -T mm[" +mm+ "]\r")     #gets users data
    data2 = telnet.read_until('system>',timeout=10)
    telnet.write('exit\r')
    telnet.close()

    print "\n" + data2      #prints data



if type == "ssh":       #ssh

    name= username + "@" + host     #sets up ssh command

    data=[]
    data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

    number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

    newdata = data[0][number-10:number] #gets the primary mm string
```

```
    rules = re.compile('(\d+)') #sets rule to find first number
    match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

    mm=match.group(1)       #assigns that number to primary mm

    os.system("ssh " + name + " users -T mm[" +mm+"]") #executes users
info command

    exitcodecheck=[]
    exitcodecheck.append(os.popen("echo $?").read());
    value=exitcodecheck[0];

    if int(value) != 0:     #Checks Exit Code of ssh command
        if int(value) == 75:
            print "Operation was not performed because the device or MM
was";
            print "not in the correct state.";
        if int(value) == 70:
            print "Internal Software Error";
        if int(value) == 64:
            print "Command Line Usage Error";
        if int(value) == 127:
            print "Command Not Found";
        if int(value) == 126:
            print "User does not have permission";
        if int(value) == 128:
            print "This value is strictly internal to AMM";
```

EXAMPLE

Telnet
python ViewUsers.py –u user –p pass5 –h 9.37.133.212 –t telnet

SSH
python ViewUsers.py –u user –h 9.37.133.212 –t ssh

3.35 WriteConfig

The WriteConfig script copies the configuration file for the specified host to a file on a TFTP server.

Note: This script is designed to connect using only SSH.

Option	Description
-u	Username
-h	Hostname
-i	IP address of the TFTP server

-d	Path to the file on the TFTP server

3.35.1 WriteConfig.sh

```
#!/bin/bash

user=
host=
ip=
directory=

while getopts “u:h:i:d:a” OPTION          #opts option
do
     case $OPTION in
         u)
             user=$OPTARG
             ;;
         h)
             host=$OPTARG
             ;;
         i)
             ip=$OPTARG
             ;;
         d)
             directory=$OPTARG
             ;;

     esac
done

if [[ -z $user ]]           #prompts for input if not opt
then
    echo "Enter username.";
    read user;
fi

if [[ -z $host ]]
then
    echo "Enter host name.";
    read host;
fi

if [[ -z $ip ]]
then
    echo "Enter the ip address of the tftp server.";
    read ip;
fi

if [[ -z $directory ]]
then
```

```
    echo "Enter the directory of the tftp file to write the
configuration";
    echo "file of $host to it.";
    echo "For example : /aaron/file.cfg";
    read directory;
fi

name="$user@$host";

command="ssh $name list -l a";

`$command>data.txt`;     #executes command to find primary mm

line=`grep '\<primary\>' data.txt`;   # finds primary string to get
primary mm line

`rm data.txt`;      #removes data file

mm=$(echo "$line" | sed 's/[^1-9]//g');   # gets primary mm


txt="ssh $name write -config file -i $ip -l $directory -T mm[$mm]";

data="echo `$txt`";     #executes command

if [[ "$data" =~ "OK" ]]        #checks and prints output
then
    echo "File written successfully to $directory on $ip."
else
    echo $data;
fi
```

EXAMPLE

SSH
bash WriteConfig.sh –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg

3.35.2 WriteConfig.pl

```
use Getopt::Long;

$result = GetOptions(           #Defines Get Opt Statements

        "username=s"  => \ $user,       #Username
        "host=s"  => \ $host,           #Host Name
        "ip=s" =>  \ $ip,               #Ip address of tftp server
        "directory=s" => \ $directory   #directory of file on tftp
server
);

if($user eq ""){                        #If Username not opt
print "Enter username.\n";
```

```
chomp($user= <>);
}
if($host eq ""){                        #If Host not opt
print "Enter host name.\n";
chomp($host = <>);
}


$name = $user."@".$host;

$command ="ssh $name list -l a";        #excutes command
@data = `$command`;
checkexitcode();

$mm=findprimarymm(@data);


if($ip eq ""){                  #If Ip address of tftp server not opt
print "Enter the ip address of the tftp server.\n";
chomp($ip =<>);
}

if($directory eq ""){                   #directory of file on tftp
server
print "Enter the directory of the tftp file to write the
configuration\nfile of $host to it. ";
print "\nFor example : /aaron/file.cfg\n";
chomp($directory =<>);
}

system("ssh $name write -config file -i $ip -l $directory -T mm[$mm]");
checkexitcode();

sub findprimarymm(@data)
{

        $find = "mm";           #String to find
        $count=0;               #loop counter
        for(@data){                     #finds lines that have mm
contained in them
                if($_ =~ /$find/){
                        @mm[$count] = "$_";
                        $count++;
                }
        }

        $count=0;               #reset loop counter
        for(@mm){               #gets the word after the mm to the end
of the line
                if(@mm[$count] =~ m/](.*?)\n/){
                @bay[$count] = $1;      #assigns the words after mm to
array called bay
                $count++;
```

```
                }
        }

        $count=0;
        for(@bay){              #checks bay array for the word primary

                @bay[$count]=trim(@bay[$count]);        #trims
whitespace

                if(@bay[$count] eq "primary"){  #if primary found assign
to baystatus

                        if(@mm[$count] =~ m/(\d+)/){
                                $baystatus = $1;
                        }

                }
        $count++;
        }
return $baystatus;
}


sub trim($)             #trims white space on mm bay
{
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

sub checkexitcode()
{
        $exitcodecheck="echo $?";
        $value= `$exitcodecheck`;

        if($value != 0)                 #Checks Exit Codes on ssh
command
        {
                if($value == 75)
                {
                        print "Operation was not performed because the
device or MM was";
                        print "\nnot in the correct state.";
                }
                if($value == 70)
                {
                        print "Internal Software Error";
                }
                if($value == 64)
                {
                        print "Command Line Usage Error";
                }
```

```
                if($value == 127)
                {
                        print "Command Not Found";
                }
                if($value == 126)
                {
                        print "User does not have permission";
                }
                if($value == 128)
                {
                        print "This value is strictly internal to AMM";
                }

        }

}
```

EXAMPLE

perl WriteConfig.pl –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg

3.35.3 WriteConfig.py

```
import os
import sys, getopt
import re

username = "None"   #checks if opt or not
host = "None"
ip = "None"
directory = "None"

opts, args = getopt.getopt(sys.argv[1:], "u:h:i:d:a", ["username=",
"host=", "ip=", "directory="]) #sets up opt


for opt, arg in opts:
    if opt in ("-u", "--username"):   #if -u opt assigns username
            username = arg

        if opt in ("-h", "--host"):   #if -h opt assigns host
            host = arg

    if opt in ("-i", "--ip"):     #if -i opt assigns ip address of tftp
server
            ip = arg

    if opt in ("-d", "--directory"):  #if -d opt assigns directory of
file to flash
            directory = arg

if username == "None":          #if not opt then ask for user input
```

```
    username= raw_input("Enter username.\n")
    username= username.rstrip('\n')

if host == "None":
    host= raw_input("Enter host name.\n")
    host= host.rstrip('\n')

if ip == "None":
    ip= raw_input("\nEnter the ip address of the tftp server.\n")
    ip= ip.rstrip('\n')

if directory == "None":
    directory= raw_input("\nEnter the directory of the tftp file to
write the configuration file\nof " +host+ " to it.\nFor example:
/aaron/file.cfg\n")
    directory= directory.rstrip('\n')

name= username + "@" + host     #sets up ssh command

data=[]
data.append(os.popen("ssh " + name + " list -l a").read()) #executes
list command

number=re.search(r'\bprimary\b',data[0]).start()  #finds the string
primary and gets number space where it starts

newdata = data[0][number-10:number] #gets the primary mm string

rules = re.compile('(\d+)') #sets rule to find first number
match = rules.search(newdata)   #executes rule to find first number
which is the primary mm number

mm=match.group(1)       #assigns that number to primary mm


os.system("ssh " + name + " write -config file -i " +ip+ " -l "
+directory+ " -T mm[" +mm+"]") #executes command for read config

exitcodecheck=[]
exitcodecheck.append(os.popen("echo $?").read());
value=exitcodecheck[0];

if int(value) != 0:     #Checks Exit Code of ssh command
    if int(value) == 75:
        print "Operation was not performed because the device or MM
was";
        print "not in the correct state.";
    if int(value) == 70:
        print "Internal Software Error";
    if int(value) == 64:
        print "Command Line Usage Error";
    if int(value) == 127:
        print "Command Not Found";
```

```
if int(value) == 126:
    print "User does not have permission";
if int(value) == 128:
    print "This value is strictly internal to AMM";
```

EXAMPLE

python WriteConfig.py –u username –h 9.37.123.423 –i 9.32.123.235 –d /filename/file.cfg