

Using ProLiant Essentials Rapid Deployment Pack for scripted blade-based switch configuration

HOWTO



Abstract.....	2
Introduction.....	2
Reader requirements.....	2
Acronyms in text.....	2
Text conventions.....	3
Problem description.....	3
Solution summary.....	6
Solution details.....	8
Acquisition List.....	8
Code description.....	8
Implementation steps.....	8
Step 1: Copy and rename one of the server blade Windows scripted install jobs.....	8
Step 2: Copy the files from the appendices to the specified directory on the RDP server.....	10
Step 3: Modify the data file to match your environment.....	10
Step 4: Add a server-side script task with the provided code and properties.....	10
Step 5: Use the newly created job by dragging and dropping it onto a server blade.....	13
Customizing the solution.....	14
Understanding the supplied code.....	14
Using an alternative to Perl.....	15
Changing properties other than VLAN.....	16
Using server-side scripts outside of the RDP environment.....	16
Code appendix.....	17
SwitchConfig.pl.....	17
SwitchInfo.txt.....	18
GetSwitchInfo.pm.....	18
pGbE.pm (Switch specific code for p-Class GbE Interconnect Switch).....	19
pGbE2.pm (Switch specific code for p-Class GbE2 Interconnect Switch).....	20
e-Gbe.pm (Switch specific code for e-Class GbE Interconnect Switch).....	21
For more information.....	23
Call to action.....	23

Abstract

This HOWTO describes how to use the HP ProLiant Essentials Rapid Deployment Pack (RDP) Windows Edition v1.40 to integrate ProLiant BL interconnect switch Virtual LAN (VLAN) configuration within server deployment. RDP Windows Edition v1.40 introduces a new feature called server-side scripts. This feature gives IT administrators the ability to create a script task that executes on the RDP deployment server and integrates with other tasks within a deployment job. Using RDP Windows Edition v1.40 server-side scripting of interconnect switches is an alternative to interconnect switch scripting outside of the RDP environment. VLAN configuration can be accomplished using the scriptable command line interface included with each interconnect switch.

The intended audience for this paper is engineers and system administrators familiar with ProLiant BL interconnect switches and deploying servers using RDP Windows Edition on ProLiant BL systems. The paper will be of particular interest to readers who want to automate the configuration of interconnect switch ports as part of server blade deployment.

Introduction

The RDP Windows Edition v1.40 server-side scripts feature gives IT administrators the ability to create a script task that executes on the RDP deployment server and integrates with other tasks within a deployment job. This permits integration of ProLiant BL Interconnect Switch VLAN configuration and other scriptable switch parameters. Although this HOWTO provides an example of interconnect switch VLAN configuration, server-side scripts can also be used for any other scriptable interconnect switch configuration parameter.

Reader requirements

Readers should be familiar with the following:

- Using Rapid Deployment Pack Windows Edition v1.4 for ProLiant BL deployments
- Configuring and maintaining ProLiant BL e-Class and p-Class interconnect switches
- Performing interconnect switch configuration through its command line interface (CLI)

For more information about the topics listed above, use the links in the [For more information](#) section.

Acronyms in text

The following acronyms are used in the text of this document.

Table 1. Acronyms

Acronym abbreviation	Acronym expansion
CLI	Command Line Interface
RDP Windows Edition	Rapid Deployment Pack Windows Edition v1.4
iLO	Integrated Lights-Out
NIC	Network Interface Controller
VLAN	Virtual Local Area Network
DB	Database
VID	VLAN Identifier
OS	Operating System
PXE	Preboot eXecution Environment

Text conventions

This HOWTO uses the following conventions to distinguish elements of text:

Menu options, Command names, Dialog box names, Screen names

These elements appear in initial capital letters and may appear in boldface for emphasis.

User input (commands to be typed)

User input appears in a different typeface and is highlighted in gray.

Scripts and files

The content of the scripts and files appears in a different typeface and is highlighted in gray with a border around it.

Comments included in the scripts are listed in blue font for explanation purposes and marked with comment markers (#) so that the code can be copied and pasted.

Problem description

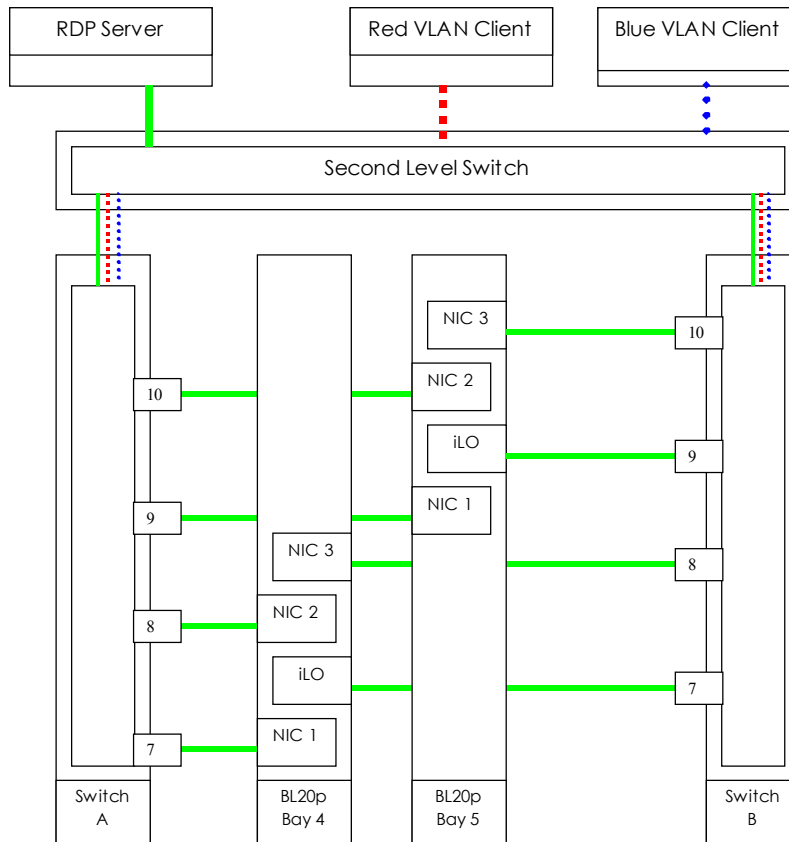
When a server deployment is initiated, often the server's network interface controllers (NIC) are associated with specific switch ports that must be configured in specific VLANs. The server being deployed cannot configure the switch; and as a result, a customer was required to log on to both switches residing in the same enclosure as the server being deployed, and then configure the proper switch ports manually.

For example, assume that your current environment uses a ProLiant BL p-Class server blade enclosure and deploys BL20p servers into one of the following configurations:

- Configuration 1: The iLO and NIC 1 (with PXE) ports are to be deployed on the management VLAN (VID=1) and NIC 2 and NIC 3 are to be deployed on the production VLAN (for our example, we will use VID=10).
- Configuration 2: This configuration is identical to the first configuration except that the production VLAN uses VID=11.

A ProLiant p-Class server blade enclosure is configured with the interconnect switches (switch A and B) and two non-deployed BL20p servers installed in bays 4 and 5 (Figure 1). The server blade enclosure is connected to a production network infrastructure that is configured with three VLANs: green management VLAN (VID=1), blue production VLAN (VID=10), and the red production VLAN (VID=11). An RDP for Windows deployment server (RDP Server) is connected to the management VLAN. The interconnect switch rear external ports (uplinks) have been configured to participate in all three VLANs (VIDs=1, 10, and 11). All other switch ports are in their factory default configuration. Therefore, the interconnect switch ports connected to the servers (downlinks) are included only in the default (management) VLAN (VID=1). With the interconnect switch in this configuration, the server NICs are unable able to access the two production VLANs, VLAN 10 and VLAN 11, making it impossible for these servers to access the production networks until the interconnect switch configuration is configured properly for this example network.

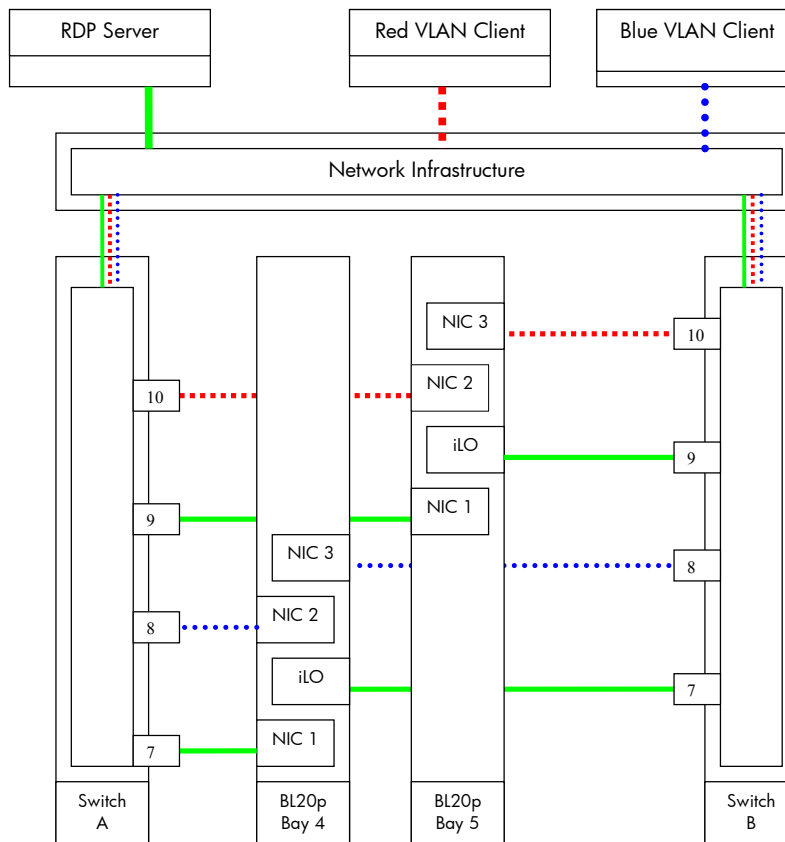
Figure 1. Initial interconnect switch configuration



In Figure 2, the configuration is the same as in Figure 1, except the servers have now been deployed with a new configuration.

- For the server in bay 4, the new configuration specifies that NICs 2 and 3 are to be placed on the blue VLAN network (VID=10). As part of this server's deployment, port 8 of both switch A and B has been configured for the blue network. Port 7 for both switches (iLO and PXE NIC 1) remain in the default management VLAN (VID=1).
- For the server in bay 5, the configuration specifies that NICs 2 and 3 are to be placed on the red VLAN network (VID=11). As part of the blade B deployment, port 10 of both switch A and B has been configured for the red network. Port 9 for both switches (iLO and PXE NIC 1) remain in the default management VLAN (VID=1).

Figure 2. Updated switch configuration after the servers have been deployed



As in this example, to deploy servers into a ProLiant BL system with the interconnect switch, the required approach had previously consisted of two main steps:

1. Automatic server deployment using RDP or other industry standard server deployment tool.
2. Manual switch configuration by directly connecting to each switch and typing in the appropriate commands.

However, as an alternative, the new server-side scripting capability within RDP Windows Edition now combines these two steps into a single, automated RDP job.

Note:

The labeling of the server NICs is performed by the operating system (OS). The example shown in figures 1 and 2 is for the BL20p deployed with a Windows operating system. For BL20p G2 deployed with a Windows operating system, the OS labels the NICs connected to switch A as NIC 2 and NIC 3, and NIC 1 to switch B. Therefore in figures 1 and 2, Switch A ports 7 and 9 would be connected to the BL20p G2 NIC labeled NIC 2 and ports 8 and 10 to NIC 3. For Switch B, ports 8 and 10 would be connected NIC 1; no labeling change occurs to the iLO NIC on switch B.

Solution summary

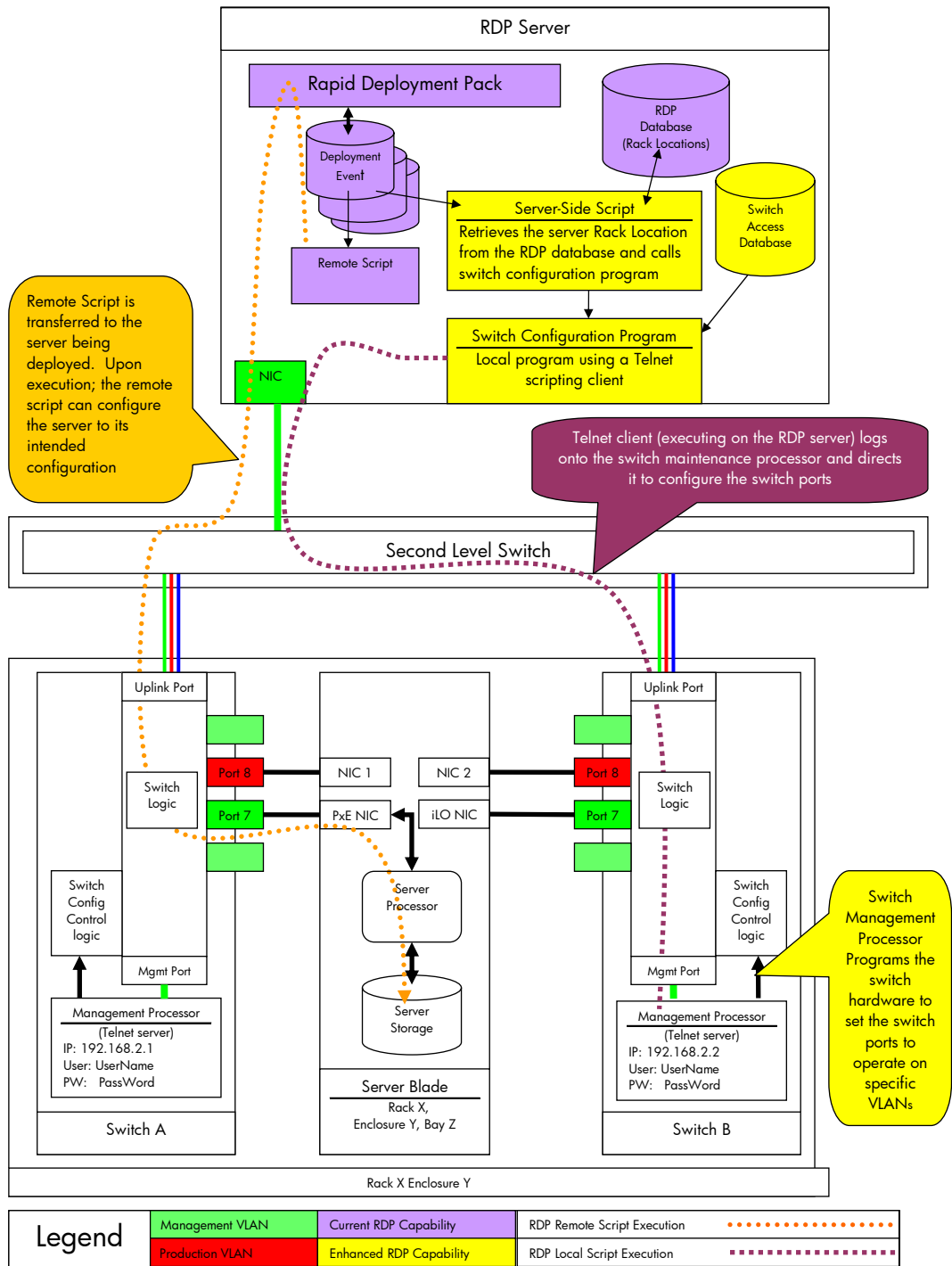
The supplied solution is designed to use RDP server-side scripts to automate the VLAN switch configuration of the interconnect switch as part of deploying an OS to an associated ProLiant BL server.

The solution involves adding a server-side script task that configures the associated interconnect switches to an existing RDP deployment job. A server-side script task is necessary because the target server cannot communicate with the switch, and even if it could, it would not know with which switch to communicate. The example task calls a Perl script that uses a data file to determine with which switches to communicate and how to talk to them. Perl is an open source scripting language available on many operating systems including Win32 (a Microsoft Windows based program) and Linux. The script uses the Rack/Enclosure/Bay information of the blade server being deployed to determine which switches to configure.

See [Solution details](#) for detailed information, including an [Acquisition List](#), [Code description](#), and a list of [Implementation steps](#).

Figure 3 illustrates how server-side scripts provide the enhanced capability to integrate VLAN switch configuration within a deployment job.

Figure 3. Solution architecture using server-side scripts



Solution details

Acquisition List

You need the following to execute the supplied example solution:

- RDP Windows Edition v1.40 (or later) deployment server
- ProLiant BL e-Class or p-Class system with associated interconnect switch option
- Perl version 5.6 (or later) with the telnet module installed on the deployment server
- Code from the appendix of this document

Code description

The scripts used in this example are partitioned into five parts.

1. **Server-Side Script:** DOS script code that obtains the rack location from the RDP database and executes the Perl based interconnect switch configuration program
2. **SwitchConfig.pl:** A Perl based interconnect switch configuration program that will configure the interconnect switch ports for the proper VLAN connectivity.
3. **GetSwitchInfo.pm:** A Perl module loaded by the interconnect switch configuration program (SwitchConfig.pl). This module contains code that is not specific to any particular interconnect switch. This module contains a single sub-function called GetSwitchInfo(). This sub-function uses passed-in rack location information as a key and searches the interconnect switch information data file (SwitchInfo.txt) to return the interconnect switch address and login information.
4. ***GbE*.pm:** A Perl module loaded by the interconnect switch configuration program. This module contains code that is specific to the interconnect switch used in the enclosure. A version of this module is provided for each switch ([pGbE.pm](#), [pGbE2.pm](#), and [eGbE.pm](#)), for the p-Class GbE, p-Class GbE2, and e-Class GbE Interconnect Switches, respectively. Each version of this module contains two sub-functions:
 - GetPortNumbers(): Maps the server blade enclosure bay number to the corresponding interconnect switch port numbers.
 - ConfigVlan(): Connects to the interconnect switch and performs the VLAN configuration.
5. **SwitchInfo.txt:** An example text based database file that contains the interconnect switch address and access information.

Implementation steps

1. [Copy and rename one of the server blade Windows scripted install jobs](#)
2. [Copy the files from the appendices to the specified directory on the RDP server](#)
3. [Modify the data file to match your environment](#)
4. [Add a server-side script task with the provided code and properties](#)
5. [Use the newly created RDP job by dragging and dropping it onto a server blade](#)

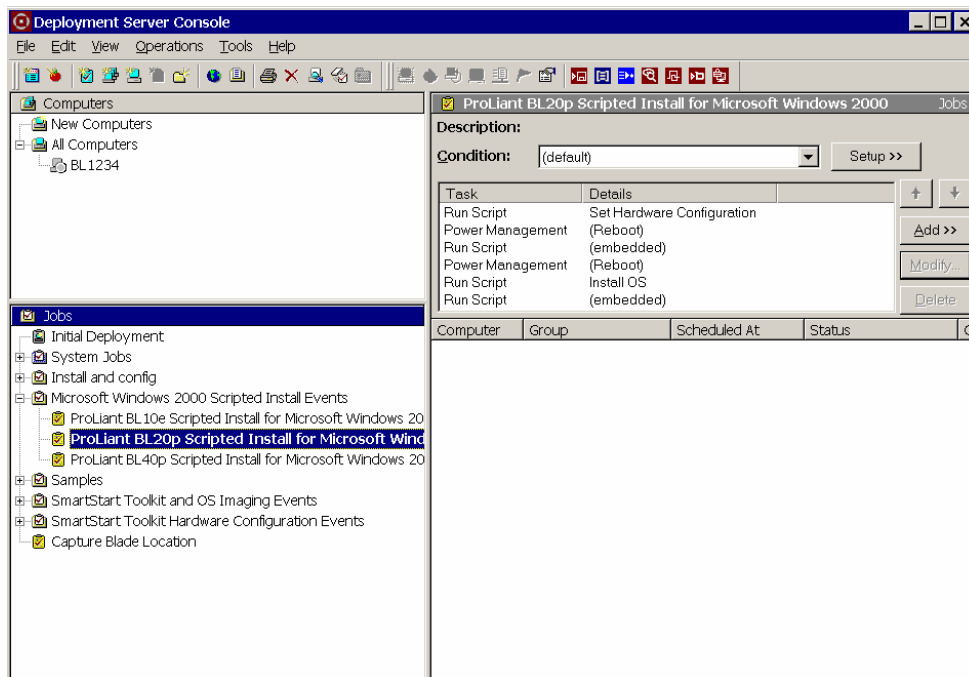
Step 1: Copy and rename one of the server blade Windows scripted install jobs

To copy a job:

1. Select the deployment job to copy.
2. Right click and select the copy operation.
3. Select the folder where the new job will be placed.
4. Right click and select **Paste**.
5. Select the new job.
6. Right click and select **Rename**.
7. Type the name for the newly created job.

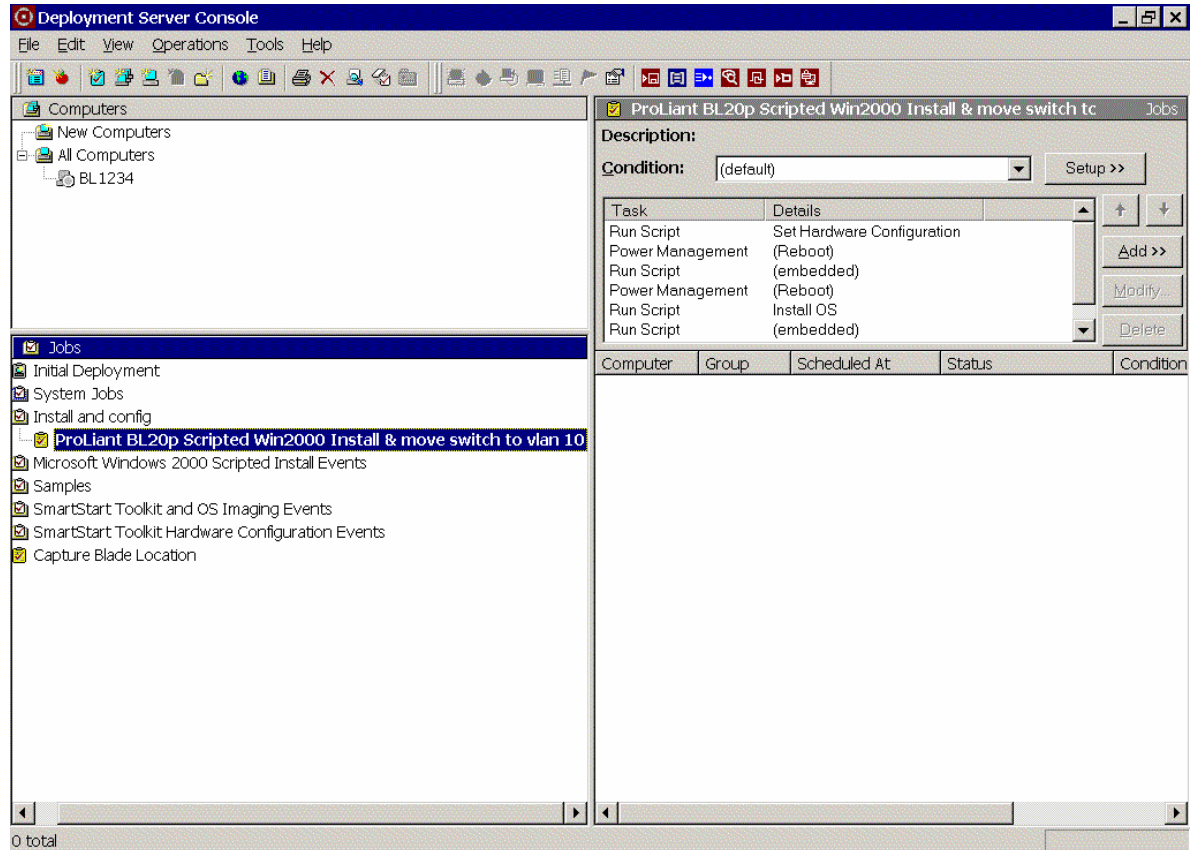
In Figure 4, *ProLiant BL20p Scripted Install for Microsoft Windows 2000* is the deployment job that will be copied for the example solution.

Figure 4. Installation job to modify is selected



In Figure 5, *ProLiant BL20p Scripted Install for Microsoft Windows 2000* job has been copied and pasted into the Install and Config folder, and then renamed to *ProLiant BL20p Scripted Win2000 Install & move switch to vlan10*.

Figure 5. Renamed job in new folder.



Step 2: Copy the files from the appendices to the specified directory on the RDP server

In order to perform the VLAN interconnect switch configuration, the appropriate code must be copied to a directory on the RDP server. The example in this paper uses `c:\SwitchVlanConfig`, but the directory can be changed to suit your needs.

All Perl files and the text switch access information file must reside in the same local directory on the RDP server. The Perl script must be executed from this directory. See Step 4 for additional changes if an alternate directory is used.

Step 3: Modify the data file to match your environment

In order for the code in the appendix to work in your environment, you will need to modify the code to reflect the appropriate configuration settings for that environment. The code in the appendices corresponds to the environment described in the Problem description section, and may not be appropriate for your environment.

Step 4: Add a server-side script task with the provided code and properties

To add a Run Script task:

1. Select the newly created job.
2. Click the **Add** button (located on the upper right hand side of the screen).
3. Select **Run Script**.
4. Paste the rack location and switch configuration call into the **Run this script** box.
5. Select the OS to run script: **Windows**.
6. Click on the **Advanced** button.

7. Select **Locally on the Deployment Server**.
8. Click **OK**.

Rack location and switch configuration call:

(The set slot, set enclosure, and set rack paragraphs within the code are three lines, not three paragraphs. A simple cut/paste will not work; cut and paste each paragraph onto one line within the Run this script box.)

```
set slot=%#"SELECT physical_bay.real_name FROM computer INNER JOIN physical_bay ON
computer.computer_id = physical_bay.computer_id INNER JOIN physical_enclosure ON
physical_bay.enclosure_id = physical_enclosure.enclosure_id INNER JOIN physical_rack
ON physical_enclosure.rack_id = physical_rack.rack_id AND computer.computer_id =
{ID} "%

set enclosure=%#"SELECT physical_enclosure.real_name FROM computer INNER JOIN
physical_bay ON computer.computer_id = physical_bay.computer_id INNER JOIN
physical_enclosure ON physical_bay.enclosure_id = physical_enclosure.enclosure_id
INNER JOIN physical_rack ON physical_enclosure.rack_id = physical_rack.rack_id AND
computer.computer_id = {ID} "%

set rack=%#"SELECT physical_rack.real_name FROM computer INNER JOIN physical_bay ON
computer.computer_id = physical_bay.computer_id INNER JOIN physical_enclosure ON
physical_bay.enclosure_id = physical_enclosure.enclosure_id INNER JOIN physical_rack
ON physical_enclosure.rack_id = physical_rack.rack_id AND computer.computer_id =
{ID} "%

REM Call a program that will configure NICs 1 and 2 to port 10 on the left and
REM right switch.

cd c:\SwitchVlanConfig

perl SwitchConfig.pl 10 >> SwitchConfig.log
```

The code reads the rack location of the server being deployed from the RDP database and writes them into environmental variables. The last line executes a Perl script that uses this rack location to properly configure a VLAN for the interconnect switch port on both the left and right switch. The number 10 is passed into the Perl script as the VLAN to be configured.

The Perl script must be executed from the directory where it resides, so the server-side script has to change directory (cd) before executing the Perl switch configuration program. The server-side script uses the command cd c:\SwitchVlanConfig to perform this task. If you use an alternate directory on the RDP server to store the switch configuration files the change directory command line must be modified to change directory into your chosen directory.

In Figure 6, the rack location and switch configuration call script has been copied and pasted into the Run this script: box and set to run in Windows.

In Figure 7, the Advanced Script Options have been accessed, and the Execution Location is chosen. The script must be set to run locally on the deployment server.

Figure 6. Rack location and switch configuration call set to run in Windows OS

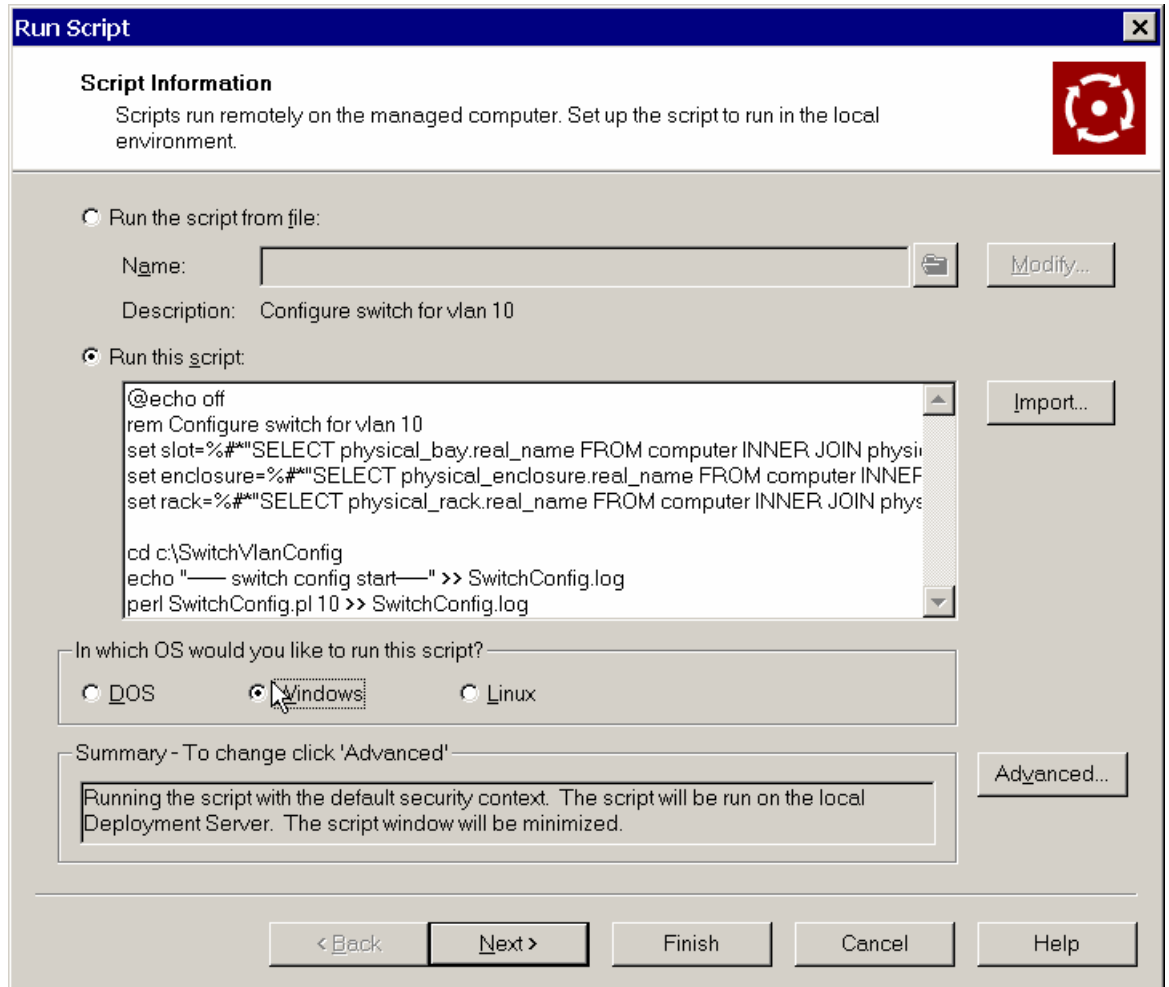
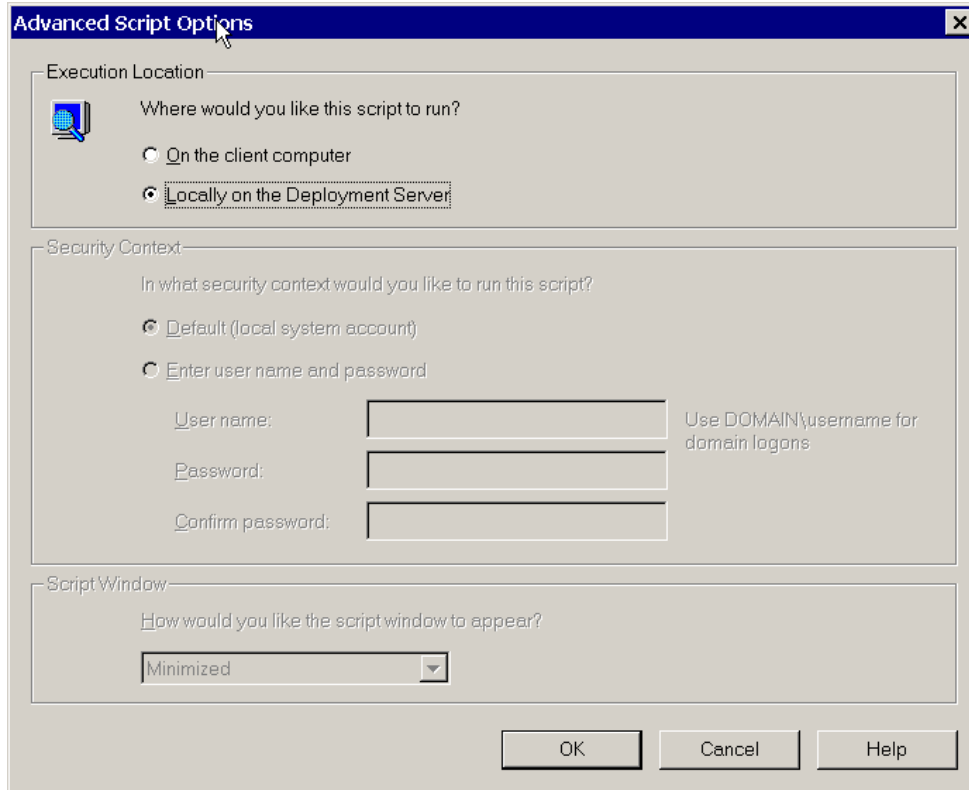


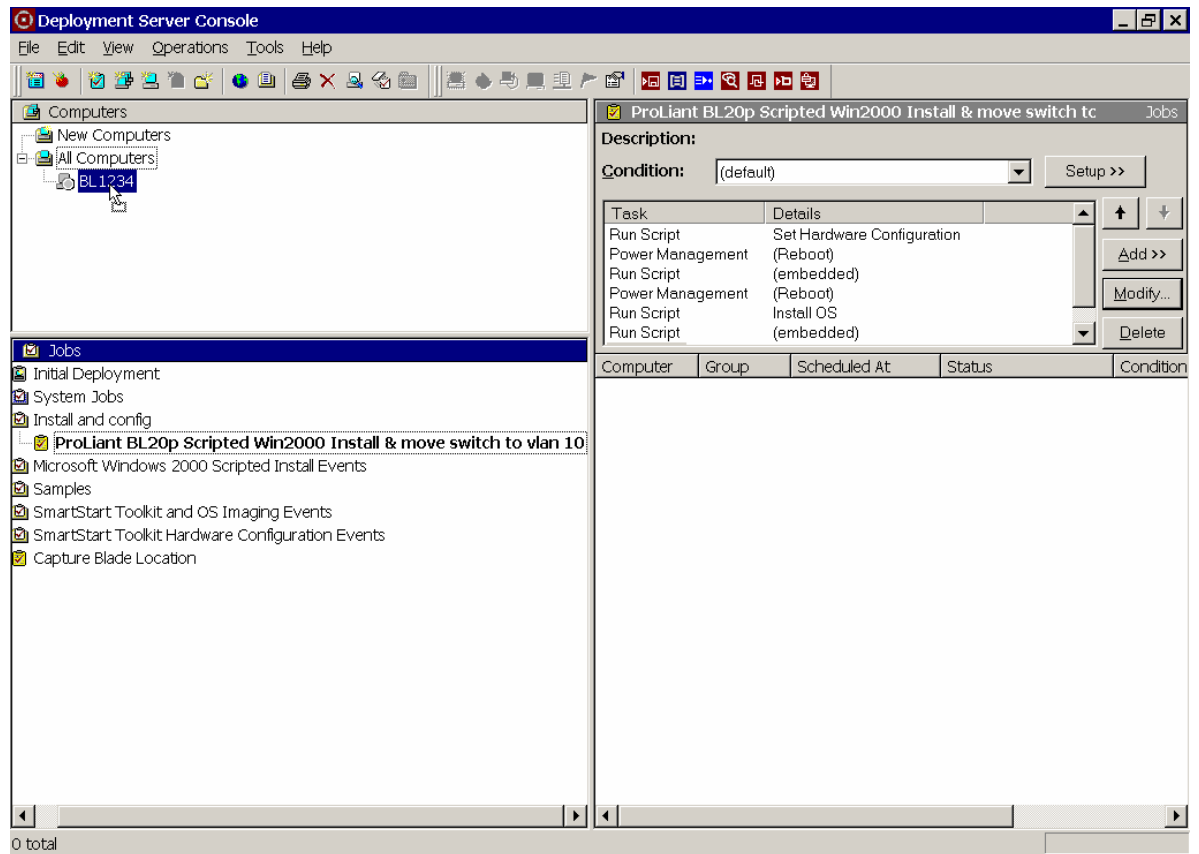
Figure 7. Advanced Script Options set to execute locally on the deployment server



Step 5: Use the newly created job by dragging and dropping it onto a server blade

The new job can be used by dragging and dropping onto a server blade within the environment as illustrated in Figure 8.

Figure 8. The new event is dragged and dropped onto server BL 1234 to execute VLAN interconnect switch configuration during server deployment



Customizing the solution

Understanding the supplied code

For our example solution, we add a final operation that configures both interconnect switches using server-side scripts. Normally, a RDP Windows Edition script executes on the server being deployed, but server-side scripts are executed locally on the RDP server. The following operations are included in the server-side script.

1. Retrieve the rack location of the server to be deployed.
2. Execute a server-side script to configure the interconnect switches.

A detailed description of the two operations is given below, including the sub-functions for operation 2 (identified by the indented bulleted items).

1. Retrieve the rack location of the server to be deployed.

In order to configure the interconnect switches, the server-side script must know which interconnect switches are attached to the server being deployed and how to log onto them. The interconnect switches that need to be configured are located in the same rack and server blade enclosure as the server being deployed. By determining the rack and server blade enclosure identifiers of the server being deployed, the script can associate these with the proper interconnect switches. The script also needs to know in which bay the deployed server resides, because it will want to configure the interconnect switch ports associated with that particular bay.

When the RDP Windows Edition boot image is downloaded onto the server being deployed, it returns the server's rack, server blade enclosure, and bay ID to the RDP server. The RDP server stores that data in its database. The location information can be retrieved from within a server-side script by executing a specific set of commands. These specific server-side script commands place the location information in environment variables that can be retrieved by a locally executed program when executed within the same context.

2. Execute a local program to configure the interconnect switches.

This local program executes on the deployment server and communicates with both interconnect switches. The preferred mechanism for remotely configuring the interconnect switches is through the use of a scriptable Telnet client. The p-Class GbE2 switches and e-Class GbE/p-Class GbE switches (version 2.0 and later) support command line interfaces that can be used for scripted Telnet operation. Scripted Telnet is script execution to a remote device via a telnet connection to that device. In the example solution Perl based Telnet scripting is used because it is easily configured and accessible to most IT professionals.

- a. Using the rack and server blade enclosure identifiers from the previously created environmental variables, the program obtains access information for the left and right interconnect switches from a database created for this solution.

In order to connect with each interconnect switch, the switch IP and login information must be known. The mechanism used in the example is to create a text file which relates a rack/enclosure combination to the left and right interconnect switch host, user, and password information. This text file needs to be generated prior to server deployment. If this mechanism is used, set the access information for this text file to limit access to everything except the interconnect switch configuration program.

- b. Next, the program finds the interconnect switch port numbers associated with the server being deployed from the server blade enclosure bay number.

Depending on which NICs need to be placed on specific VLANs, the bay number needs to be mapped into the appropriate interconnect switch port numbers. Bay to port mapping varies: e-Class interconnect switches have a different bay to port mapping than p-Class systems. Additionally, within the p-Class system, BL20p and BL40p server NIC to interconnect switch mapping will also vary.

- c. The program then logs onto the left interconnect switch using scripted Telnet. (The interconnect switch access information is required as part of the logon process.)

The logon process is slightly different depending on the type of interconnect switch type used. The interconnect switches also need to be configured to operate in a non-interactive mode. See the code examples for an explanation how this is accomplished.

- d. The program places the ports on the proper VLAN.

It executes the proper CLI commands to place the switch ports on the appropriate VLANs. The CLI syntax is different between p-Class GbE2 and e-Class GbE/p-Class GbE.

- e. Finally, the program repeats the last three steps for the side B switch. For e-Class switches, we do not configure the B side switch.

Using an alternative to Perl

In order to customize a solution based on the provided example solution, a Win32 based program capable of Telnet scripting must be used. While it is not the only option for users, Perl was used in the example solution because it is widely accessible, easily configured, and commonly used in IT environments. Other Win32 programs may be used, but have not been tested in this environment.

Changing properties other than VLAN

Server-side scripts can be used to accomplish more than VLAN interconnect switch configuration. As users become comfortable with this feature, server-side script tasks can be created to operate with other tasks within an RDP for Windows deployment job and automate otherwise manually performed functions. For example, a newly deployed server can be added into membership with your load balance.

Note:

Due to the nature of the OS installation task, RDP executes the next task without waiting for the installation to complete. This execution allows interconnect switch configuration to start while the server OS is in the process of being installed. Care should be taken if the actions within the server-side script could interfere with the completion of OS installation.

Using server-side scripts outside of the RDP environment

While the configuration scripts provided in this HOWTO describe how to integrate interconnect switch configuration in RDP for Windows server deployment, interconnect switch configuration can also be scripted without RDP. This can be executed using the scriptable command line interface (CLI) provided with each interconnect switch.

Without RDP Windows Edition integration, the script still needs to determine the interconnect switch login and port information. The interconnect switch login information can be determined by passing in the rack location as command line options to the interconnect switch configuration program, instead of obtaining them through the environmental variables.

For example:

```
local($enclosure) = $_[1];  
local($rack) = $_[2];  
local($slot) = $_[3];
```

Any valid interconnect switch command line can be executed. Therefore, a complete or partial interconnect switch configuration can be accomplished through the scripting mechanism.

The entire interconnect switch configuration can also be captured into an editable file using the interconnect switch configuration file capture and restore capability. On p-Class GbE2 Interconnect Switches, the command to save the complete configuration file is:

```
/cfg/dump  
/cfg/ptcfg <TFTP server> <filename>
```

The command to restore the switch configuration file is:

```
/cfg/gtcfg <TFTP server> <filename>  
apply
```

Equivalent configuration file save and restore commands exist for p-Class GbE and e-Class GbE Interconnect Switches; the command to save the complete configuration file is:

```
tftp upload type cfg [path <path_filename>] [ip <ip_addr>] [port <port_num>]
```

The command to restore the switch configuration file is:

```
tftp download type cfg [ip <ip_addr>] [port <port_num>]  
cfg save
```

For more information on using the interconnect switch CLI, refer to the user documentation supplied with the interconnect switch.

Code appendix

This section includes sample scripts as a reference for examples provided throughout the HOWTO.

SwitchConfig.pl

SwitchConfig.pl is a Perl based interconnect switch configuration program that configures the interconnect switch ports for the proper VLAN connectivity.

```
# Load the p-GbE2 specific telnet module.
use pGbe2; # Module to communicate with the switch using telnet. # When
using p-GbE or e-GbE, change the use line to the accurate module.
use GetSwitchInfo; # Used to get switch details.

# Obtain the rack location from the environmental variables set by RDP.
# The VLAN to use is passed as the first argument.
local($enc) = $ENV{'enclosure'};
local($rack) = $ENV{'rack'};
local($slot) = $ENV{'slot'};
$exit status = 0;
local($vlan) = @ARGV[0];

if(!$vlan){
    die "usage: $0 vlan_id\n";
}

# Get the switch port numbers that connect to the server in the given slot.
%ports = GetPortNumbers($slot);

# Obtain the switch access information for the switch A.
# The information is returned as a name-value list.
# This name-value list contains all parameters required to connect
# to the switch.
%switch A info = GetSwitchInfo($enc, $rack, "switchA");
# Set the switch A's VLAN on the specific ports or print an error
# message if not found.
if($switch A info{'host'}){
    print "Configuring $switch A info{'host'}...\n";
    @A output = &ConfigVlan(*switch A info,*ports, $vlan);
} else {
    print "Error: Unable to find switch A information in DB file.\n";
    $exit_status = 1;
}

# Print the output captured in the named value list.
foreach $line (@A_output)
{
    print "$line\n";
}

# Repeat for switch B.
%switch B info = GetSwitchInfo($enc, $rack, "switchB");
if($switch_B_info{'host'}){
    print "Configuring $switch_B_info{'host'}...\n";
    @B output = &ConfigVlan(*switch B info,*ports, $vlan);
} else {
    print "Error: Unable to find switch B information in DB file.\n";
    $exit_status = 1;
}

foreach $line (@B output)
{
    print "$line\n";
}

exit($exit_status);
```

SwitchInfo.txt

SwitchInfo.txt is an example text based database file containing the interconnect switch address and access information.

```
# The following record represents switch login information.
# Lines not beginning with the switch location are ignored
# including blank lines.

# Each line is of the following form:
# EnclosureName-RackName-switchAorB:SwitchType:NameOrIP:User:Password.
# Rack name is rack1, enclosure name is enclosure0.

# IP address is 192.168.10.15. User name is root, password is admin.
enc0-rack1-switchA:p-GbE2:192.168.10.15:root:admin
# IP address is 192.168.10.16. There is no user name, password is admin.
enc0-rack1-switchB:p-GbE2:192.168.10.16::admin
```

GetSwitchInfo.pm

GetSwitchInfo.pm is a Perl module loaded by the interconnect switch configuration program.

```
sub GetSwitchInfo
{
    # Set switch location from passed in parameters.
    local($enc) = @_ [0];
    local($rack) = @_ [1];
    local($side) = @_ [2]; # A or B

    # Define local variables for the return values and the line buffer.
    local($line);
    # Define the search key which is the rack location and
    # the switch side separated by `-'`.
    local($key) = $enc . "-$rack-" . $side;
    local(%switch_params);

    # Set the length of the search key.
    local($keylen) = length($key);

    # Open the switch information file and read each line until
    # there are no more lines to read.
    open(filehandle, "<SwitchInfo.txt")
        or die "error opening SwitchInfo.txt\n";
    while(1){
        $line = <filehandle>;
        if(!$line){
            last;
        }

        # Remove the newline from the end of the line
        # and check for a search match.
        chomp($line);

        if(substr($line, 0, $keylen) eq $key){ #found the line matching the key
            # Split the line on colons, the first entry will be the key
            @tmp = split(/:/, substr($line, $keylen + 1), -1);
            $switch_params{'switch-type'} = $tmp[0];
            $switch_params{'host'} = $tmp[1];
            $switch_params{'user'} = $tmp[2];
            $switch_params{'password'} = $tmp[3];
            $switch_params{'side'} = $side;
            return %switch_params;
        }
    }
    # Return the values.
    return %switch_params;
}

1;
```

pGbE.pm (Switch specific code for p-Class GbE Interconnect Switch)

Note: Change the use line in switchconfing.pl to match the module used.

```
use Net::Telnet;
# Connect to the switch using the passed in switch access information
# configure the port specified by the passed in port number to the passed
# in vlan ID.
sub ConfigVlan
{
    local(*switch_info) = @_ [0]; # Switch login information
    local(*ports) = @_ [1];      # List of ports for this slot
    local ($vlan) = @_ [2];
    local(@lines);
    # Obtain switch login information.
    $host = $switch_info{'host'};
    $password = $switch_info{'password'};
    $user = $switch_info{'user'};

    # For this example we are changing NIC 2 and NIC3.
    # The first port is the iLO (switch B) and PXE (switch A)
    # NIC that we are not changing.
    $port = $ports{'nic_2-3'};

    # Set prompt to use.
    my $prompt = "/>+/";

    # Create Telnet access object.
    $telnet = new Net::Telnet(Host => $host, Prompt => $prompt);
    # Connect to the switch.
    $ok = $telnet->open();

    # We wait for the switch to prompt for the user/password.
    $ok = $telnet->waitfor(Match => '/username:*/i');
    # Log onto the switch using the user and password.
    $ok = $telnet->print($user);
    $ok = $telnet->waitfor(Match => '/password:*/i');
    $ok = $telnet->print($password);

    # Send a Return. This will be a no-operation if already in CLI mode
    # and will switch to CLI if in menu mode as "switch to CLI"
    # is the first item in the menu.
    $ok = $telnet->print("");
    $ok = $telnet->waitfor(Match => $prompt);

    # Set paging mode off so the CLI is not looking for interactive commands.
    $ok = $telnet->cmd("paging off");

    # Set the port to operate on the specified vlan.
    @tmp = $telnet->cmd("vlan create id $vlan name vlan $vlan");
    push(@lines, @tmp);

    @tmp = $telnet->cmd("vlan add port id $vlan eg untagged $port");
    push(@lines, @tmp);

    @tmp = $telnet->cmd("vlan set pvid port $port id $vlan");
    push(@lines, @tmp);

    # Save the configuration change. It will be in effect even after a reboot.

    @tmp = $telnet->cmd("cfg save");
    push(@lines, @tmp);
    $ok = $telnet->print("logout");

    $telnet->close();
    return @lines;
}

# Using the passed in slot number, calculate the switch port numbers
# assigned to:
# iLO (Right Switch) or PXE (Left Switch). This is nic_0-1.
# NIC3 (Right Switch) or NIC2 (Left Switch). This is nic_2-3.
sub GetPortNumbers
{
    local(%switch_ports);
    local($slot) = @_ [0]; # Get passed in slot number
```

```

if($slot > 0 && $slot <= 8){ #Insure a slot number was passed in
    $switch_ports{'nic_0-1'} = (2 * $slot) - 1;
    $switch_ports{'nic_2-3'} = (2 * $slot);
} else {
    die "Invalid slot number $slot\n";
}
return %switch ports;
}
1;

```

pGbE2.pm (Switch specific code for p-Class GbE2 Interconnect Switch)

Note: Change the use line in switchconfg.pl to match the module used.

```

use Net::Telnet;

# Connect to the switch using the passed in switch access information
# configure the port specified by the passed in port number to the
# passed in vlan ID.
sub ConfigVlan
{
    local(*switch info) = @ [0]; # Switch login information.
    local(*ports) = @ [1];      # List of ports for this slot.
    local($vlan) = @ [2];
    local(@lines);

    # Obtain switch login information.
    $host = $switch_info{'host'};
    $password = $switch_info{'password'};

    # For this example, we are changing NIC 2 and NIC3.
    # The first port is the iLO (switch B) and PXE (switch A)
    # NIC that we are not changing.
    $port = $ports{'nic_2-3'};

    # Set prompt to use.
    my $prompt = '/\>\>\>.*#/';

    # Create Telnet access object.
    $telnet = new Net::Telnet(Host => $host,
        Prompt => $prompt);
    # Open the telnet session, wait for login prompt, and login.
    $ok = $telnet->open();
    $ok = $telnet->waitfor(Match => "/password[: ]*/i");
    $ok = $telnet->print($password);

    # After the prompt, set to not page or ask questions.
    $ok = $telnet->waitfor( Match => $prompt);
    $ok = $telnet->cmd("lines 0"); # no 'press any key' stuff
    $ok = $telnet->cmd("verbose 0"); # no questions

    # Add the port to the defined VLAN with commands and collect output.
    @tmp = $telnet->cmd("/cfg/vlan $vlan/add $port");
    push(@lines, @tmp);
    @tmp = $telnet->cmd("/cfg/port $port/pvid $vlan");
    push(@lines, @tmp);
    # Apply the change and save to take effect if the switch is restarted.
    @tmp = $telnet->cmd("apply");
    push(@lines, @tmp);
    @tmp = $telnet->cmd("save");
    push(@lines, @tmp);
    # Logoff and return the output.
    $telnet->close();
    return @lines;
}

# Using the passed in slot number, calculate the switch port numbers
# assigned to:
# iLO (Right Switch) or PxE (Left Switch). This is nic 0-1.
# NIC3 (Right Switch) or NIC2 (Left Switch). This is nic 2-3.
sub GetPortNumbers
{
    local(%switch ports);
    local($slot) = @_[0]; # Get passed in slot number.
    if($slot > 0 && $slot <= 8){ # Ensure a slot number was passed in.

```

```

    $switch ports{'nic i0-1'} = (2 * $slot) - 1;
    $switch_ports{'nic_2-3'} = (2 * $slot);
  } else {
    die "Invalid slot number $slot\n";
  }
}

return %switch ports;
}

1;

```

e-Gbe.pm (Switch specific code for e-Class GbE Interconnect Switch)

Note: Change the use line in switchconfging.pl to match the module used.

```

use Net::Telnet;
# Connect to the switch using the passed in switch access information
# configure the port specified by the passed in port number to the passed # in vlan
ID.
sub ConfigVlan
{
  local(*switch_info) = @_ [0]; # Switch login information.
  local(*ports) = @_ [1]; # List of ports for this slot.
  local($vlan) = @_ [2];
  local(@lines);
  if($switch_info{'side'} eq 'switchA'){
    print "Warning: Ignorning configure request for e-GbE switchA\n";
    return;
  }

  # Obtain switch login information.
  $host = $switch_info{'host'};
  $password = $switch_info{'password'};
  $user = $switch_info{'user'};

  # For this example, we are changing NIC 2 and NIC3.
  # The first port is the iLO (switch B) and PXE (switch A)
  # NIC that we are not changing.
  $port = $ports{'nic_0-1'};

  # Set prompt to use.
  my $prompt = "/>+/";

  # Create Telnet access object.
  $telnet = new Net::Telnet(Host => $host, Prompt => $prompt);
  # Connect to the switch.
  $ok = $telnet->open();

  # We wait for the switch to prompt for the user/password.
  $ok = $telnet->waitfor(Match => '/username:*/i');
  # Log onto the switch using the user and password.
  $ok = $telnet->print($user);
  $ok = $telnet->waitfor(Match => '/password:*/i');
  $ok = $telnet->print($password);

  # Send a Return. This will be a no-operation if already in CLI mode
  # and will switch to CLI if in menu mode as "switch to CLI"
  # is the first item in the menu.
  $ok = $telnet->print("");
  $ok = $telnet->waitfor(Match => $prompt);

  # Set paging mode off so the CLI is not looking for interactive commands.
  $ok = $telnet->cmd("paging off");

  # Set the port to operate on the specified vlan.
  @tmp = $telnet->cmd("vlan create id $vlan name vlan $vlan");
  push(@lines, @tmp);

  @tmp = $telnet->cmd("vlan add port id $vlan eg untagged $port");
  push(@lines, @tmp);

  @tmp = $telnet->cmd("vlan set pvid port $port id $vlan");
  push(@lines, @tmp);

  # Save the configuration change. It will be in effect even after a reboot.

```

```

@tmp = $telnet->cmd("cfg save");
push(@lines, @tmp);
$ok = $telnet->print("logout");

$telnet->close();
return @lines;
}

# Using the passed in slot number, calculate the switch port numbers
# assigned to:
# iLO (Right Switch) or PxE (Left Switch). This is nic 0-1.
# NIC3 (Right Switch) or NIC2 (Left Switch). This is nic 2-3.

# Using the passed in slot number, calculate the switch port numbers
# assigned to:
# Data(Right Switch) or PxE (Left Switch). This is nic 0-1.
sub GetPortNumbers
{
    local(%switch ports);
    local($slot) = $_[0]; # Get passed in slot number.
    if($slot > 0 && $slot <= 10){ # Ensure a slot number was passed in.
        $switch ports{'nic 0-1'} = $slot;
    } else {
        die "Invalid slot number $slot\n";
    }
}

return %switch ports;
}

1;

```

For more information

For additional information, refer to the resources detailed below.

Table 3. Web resources

Resource description	Web address
ProLiant Essentials Rapid Deployment Pack	www.hp.com/servers/rdp
e-Class GbE Interconnect Switch	http://h18004.www1.hp.com/products/servers/proliant-bl/e-class/interconnect-switch.html
p-Class GbE Interconnect Switch	http://h18004.www1.hp.com/products/servers/proliant-bl/p-class/20p/bl-p-interconnect-switch.html
p-Class GbE2 Interconnect Switch	http://h18004.www1.hp.com/products/servers/proliant-bl/p-class/20p/bl-p-interconnect-switch2.html
Interconnect switch user documentation	http://h71025.www7.hp.com/support/home/selectproduct.asp?destination=reflib

Call to action

To help us better understand and meet your needs for ISS technology information, please evaluate this paper by completing the short survey at

www.zoomerang.com/survey.zgi?JA35NV4DV4L3HQDCW7PP9QD2.

Note: This URL will be active through 30 November 2003. Please send questions and further comments about this paper to: TechCom@HP.com

© 2003 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Microsoft, Windows, and Windows NT are US registered trademarks of Microsoft Corporation.

TC030905HT, 09/2003

