CHAPTER 13

Support Tools and Advanced Concepts

This chapter contains information on, and explains the use of, tools that help you maintain Broadband Access Center (BAC) as well as speed and improve the installation, deployment, and use of this product.

This chapter discusses:

Note This section contains many examples of tool use. In many cases, the tool filenames include a path specified as *BAC_home*. This indicates the default directory location.

BAC Tools

BAC provides automated tools that you use to perform certain functions more efficiently. Table 13-1 lists the various tools that this BAC release supports.

Table 13-1 BAC Tools

Tool	Description	Refer...
Configuration File Utility	Used to test, validate, and view BAC template and configuration files.	Using the Configuration File Utility, page 8-27
BAC Process Watchdog	Interacts with the BAC watchdog daemon to observe the status of the BAC system components, and stop or start servers.	Using the BAC Process Watchdog from the Command Line, page 2-14

Table 13-1 BAC Tools (continued)

Tool	Description	Refer...
RDU Log Level Tool	Sets the log level of the RDU, and enables or disables debugging log output.	Using the RDU Log Level Tool, page 13-2
PacketCable Certificates Tool	Installs, and manages, the KDC certificates that are required by the KDC for its operation.	Using the PKCert.sh Tool, page 13-5
KeyGen Tool	Generate PacketCable service keys.	Using the KeyGen Tool, page 13-11
Changing Network Registrar Properties Tool	Used to change key configuration properties used by BAC extensions that are incorporated into the Network Registrar DHCP server.	Using the changeNRProperties.sh Tool, page 13-13
SNMP Agent Configuration Tool	Manages the SNMP agent.	Using the snmpAgentCfgUtil.sh Tool, page 13-15
Disk Space Monitoring Tool	Sets threshold values for one or more file systems. When these thresholds are surpassed, an alert is generated until additional disk space is available.	Using the disk_monitor.sh Tool, page 13-21

Using the RDU Log Level Tool

Use the RDU log level tool to change the current log level of the RDU from the command line, using the **setLogLevel.sh** command. This tool resides in the *BAC_home*/rdu/bin directory. Table 13-2 identifies the available severity levels and the types of messages written to the log file when enabled.

Table 13-2 Logging Levels

Log Level	Description
0-Emergency	System unstable. Sets the logging function to save all emergency messages.
1-Alert	Immediate action needed. Sets the logging function to save all activities that need immediate action and those of a more severe nature.
2-Critical	Critical conditions exist. Sets the logging function to save all error messages and those of a more severe nature
3-Error	Error conditions exist. Sets the logging function to save all error messages and those of a more severe nature.
4-Warning	Warning conditions exist. Sets the logging function to save all warning messages and those of a more severe nature.

Table 13-2 Logging Levels (continued)

Log Level	Description
5-Notification	A normal, but significant, condition exists. Sets the logging function to save all notification messages and those of a more severe nature.
6-Information	Informational messages. Sets the logging function to save all logging messages available.

Note Another level known as 7-DEBUG is used exclusively by Cisco for debugging purposes. Do not use this level except at the direction of the Cisco TAC.

We recommend that you keep the RDU severity level at the Warning level to help maintain a steady operations state. The Information level is recommended to be used with caution if you need to maintain steady state performance during debug operations. You should exercise caution when running with the Information level because this creates a great number of log entries, which in itself can adversely impact performance.

Note The RDU process has to be up to execute the log level tool. Also, you must be a privileged user to run this tool by using the **setLogLevel.sh** command.

Syntax Description

setLogLevel.sh -[*0..6*] [**-help**] [**-show**] [**-default**] [**-debug**]

- **-**[*0..6*]—Identifies the severity level to be used. For a list of available levels, see Table 13-2.
- **-help**—Displays help for the tool.
- **-show**—Displays the current severity level set for the RDU server.
- **-default**—Sets the RDU to the installation default level 5 (notification).
- **-debug**— Sets an interactive mode to enable or disable tracing categories for the RDU server.

Note You should only enable the debug settings that the Cisco support staff recommends.

You can also use this tool to perform these functions:

- Setting the RDU Log Level, page 13-3
- Viewing the Current Log Level of RDU, page 13-4

Setting the RDU Log Level

You can use this tool to change the logging level from one value to another value. The following example illustrates how to set the RDU logging level to the warning level, as indicated by the number 4 in the **setLogLevel.sh** command. The actual log level set is not important for the procedure, it can be interchanged as required.

The example described in this section assumes that the RDU server is up, the username for the RDU is **admin**, and the password is **changeme.**

To set the RDU logging level:

Step 1 Change directory to *BAC_home*/rdu/bin.

Step 2 Run the RDU log level tool using this command:

```
# setLogLevel.sh 4
```

This prompt appears:

```
Please type RDU username:
```

Step 3 Enter the RDU username. In this example, the default username (**admin**) is used.

```
Please type RDU username: admin
```

This prompt appears:

```
Please type RDU password:
```

Step 4 Enter the RDU password for the RDU. In this example, the default password (**changeme**) is used.

```
Please type RDU password: changeme
```

This message appears to notify you that the log level has been changed. In this example, the level was 5, for notification, and is now 4, for warning.

```
RDU Log level was changed from 5 (notification) to 4 (warning).
```

Viewing the Current Log Level of RDU

You can use this tool to view the RDU log and determine which logging level is configured before attempting to change the value.

The example described in this section assumes that the RDU server is up, the username for the RDU is **admin**, and the password is **changeme.**

To view the current logging level of the RDU:

Step 1 Change directory to *BAC_home*/rdu/bin.

Step 2 Run this command:

```
# setLogLevel.sh -show
```

This prompt appears:

```
Please type RDU username:
```

Step 3 Enter the RDU username (**admin**) and press **Enter**.

```
Please type RDU username: admin
```

This prompt appears:

```
Please type RDU password:
```

Step 4 Enter the RDU password (**changeme**) and press **Enter**.

```
Please type RDU password: changeme
```

This message appears:

```
The logging is currently set at level: 4 (warning)

All tracing is currently disabled.
```

Using the PKCert.sh Tool

The PKCert tool creates the certificate and the corresponding private key that the KDC requires. It also installs the CableLabs service provider certificates as certificate files, similar to those shown below.

- Cablelabs_Service_Provider_Root.cer
- Service_Provider.cer
- Local_System.cer
- KDC.cer

This tool also allows you to verify certificate chains and copy and rename a certificate chain to the names required by the KDC.

Note This tool is available only when the KDC component is installed.

Running the PKCert Tool

Run the PKCert tool by executing the PKCert.sh command, which resides by default in the *BAC_home*/kdc directory.

Syntax Description

PKCert.sh *function option*

- *function*—Identifies the function to be performed. You can choose:
  - **-c**—Creates a KDC certificate. See Creating a KDC Certificate, page 13-6.
  - **-v**—Verifies and normalizes the PacketCable certificate set. See Validating the KDC Certificates, page 13-7.
  - **-z**—Sets the log level for debug output that is stored in the pkcert.log file. See Setting the Log Level for Debug Output, page 13-8.

  Note If you have trouble in using these options, specify **-?** to display available help information.

- *option*—Implements optional functions, depending on the function you selected.

Creating a KDC Certificate

To create the KDC certificate:

Step 1 Change directory to /opt/CSCObpr/kdc.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -s *dir* **-d** *dir* **-c** *cert* **-e -r** *realm* **-a** *name* **-k** *keyFile* [**-n** *serial#*] [**-o**]

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-c** *cert*—Uses the service provider certificate (DER encoded)
- **-e**—Identifies the certificate as a Euro PacketCable certificate
- **-r** *realm*—Specifies the Kerberos realm for the KDC certificate
- **-a** *name*—Specifies the DNS name of the KDC
- **-k** *keyFile*—Uses the service provider private key (DER encoded)
- **-n** *serial#*—Sets the certificate serial number
- **-o**—Overwrites existing files

When a new certificate is created and installed, the new certificate identifies the realm in the subject alternate name field. The new certificate is unique to its current environment in that it contains the:

- KDC realm.
- DNS name associated with this KDC that the Media Terminal Adapter (MTA) will use.

Examples

```
# ./PKCert.sh  -c "-s . \
> -d /opt/CSCObpr/kdc/solaris/packetcable/certificates \
> -k CLCerts/Test_LSCA_privkey.der \
> -c CLCerts/Test_LSCA.cer \
> -r PCTEST.CISCO.COM \
> -n 100 \
> -a kdc.pctest.cisco.com \
> -o"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: .
Destination Directory: /opt/CSCObpr/kdc/solaris/packetcable/certificates
Private Key File: CLCerts/Test_LSCA_privkey.der
Certificate File: CLCerts/Test_LSCA.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC_private_key.pkcs8
File written:
/opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC_private_key_proprietary.
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC_PublicKey.der
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC.cer
```

```
KDC Certificate Successfully Created at
/opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC.cer
```

Using this command creates the files /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC.cer and /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC_private_key.pkcs8. The KDC certificate will have a realm set to PCTEST.CISCO.COM, a serial number set to 100, and the fully qualified domain name (FQDN) of the KDC server set to kdc.pctest.cisco.com.

Validating the KDC Certificates

This command examines all files in the source directory specified and attempts to identify them as X.509 certificates. If legitimate X.509 certificates are found, the files are properly renamed and copied to the destination directory. An error is generated when more than one legitimate chain of certificates for a particular purpose (service provider or device) is identified. If this occurs, you must remove the extra certificate from the source directory and run the command again.

Note When you enter the **PKCert.sh -v -?** command, usage instructions for validating KDC certificates by using the PKCert tool appear.

To validate the KDC certificate:

Step 1 Change directory to /opt/CSCObpr/kdc.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -v -s *dir* **-d** *dir* **-r** *dir*

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-o**—Overwrites any existing files
- **-r** *dir*—Specifies the reference certificate directory

Verification is performed against reference certificates built into this package. If you specify the **-d** option, the certificates are installed in the target directory with name normalization.

Examples

```
# ./PKCert.sh  -v \
> "-s /opt/CSCObpr/kdc/TestCerts
> -d /opt/CSCObpr/kdc/solaris/packetcable/certificates \
> -o"
Pkcert Version 1.0
Logging to pkcert.log
Output files will overwrite existing files in destination directory

Cert Chain(0)    Chain Type: Service Provider
[Local File]                          [Certificate Label]                    [PacketCable
Name]
CableLabs_Service_Provider_Root.cer   CableLabs_Service_Provider_Root.cer
Service_Provider.cer                  Service_Provider.cer
Local_System.cer                      Local_System.cer
KDC.cer                               KDC.cer
```

```
Cert Chain(1)    Chain Type: Device
[Local File]                          [Certificate Label]                    [PacketCable
Name]
MTA_Root.cer                           MTA_Root.cer
File written:
/opt/CSCObpr/kdc/solaris/packetcable/certificates/CableLabs_Service_Provider_Root.cer
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/Service_Provider.cer
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/Local_System.cer
File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC.cer

Service Provider Certificate Chain Written to Destination Directory
/opt/CSCObpr/kdc/solaris/packetcable/certificates


File written: /opt/CSCObpr/kdc/solaris/packetcable/certificates/MTA_Root.cer

Device Certificate Chain Written to Destination Directory
/opt/CSCObpr/kdc/solaris/packetcable/certificates
```

Setting the Log Level for Debug Output

This command enables you to set the log level for debug output that is logged in pkcert.log, which resides in *BAC_home*/kdc. You can use the data in the log file to troubleshoot any problems that may have occurred while performing the requested tasks.

To set the log level for debug output:

Step 1 Change directory to /opt/CSCObpr/kdc.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -s *dir* **-d** *dir* **-k** *keyFile* **-c** *cert* **-r** *realm* **-a** *name* **-n** *serial#* **-o** {**-z error | info | debug}**

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-k** *keyFile*—Uses the service provider private key (DER encoded)
- **-c** *cert*—Uses the service provider certificate (DER encoded)
- **-r** *realm*—Specifies the Kerberos realm for the KDC certificate
- **-a** *name*—Specifies the DNS name of the KDC
- **-n** *serial#*—Sets the certificate serial number
- **-o**—Overwrites existing files
- **-z**—Sets the log level for debug output that is stored in the pkcert.log file. The values you can choose are:
  - **error**—Specifies the logging of error messages.
  - **info**—Specifies the logging of informational messages.
  - **debug**—Specifies the logging of debug messages. This is the default setting.

Examples

Example 1

In this example, the log level is set for collecting error messages.

```
# ./PKCert.sh  -c "-s /var/certsInput
> -d /var/certsOutput
> -k /var/certsInput/Local_System.der
> -c /var/certsInput/Local_System.cer
> -r PCTEST.CISCO.COM
> -n 100
> -a kdc.pctest.cisco.com
> -o -z error"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
Setting debug to error
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e.
/opt/CSCObpr/kdc/solaris/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObpr/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObpr/kdc/solaris)
```

Example 2

In this example, the log level is set for collecting information messages.

```
# ./PKCert.sh -c "-s /var/certsInput
> -d /var/certsOutput
> -k /var/certsInput/Local_System.der
> -c /var/certsInput/Local_System.cer
> -r PCTEST.CISCO.COM
> -n 100
> -a kdc.pctest.cisco.com
> -o -z info"
 INFO [main] 2007-05-02 06:32:26,280 (PKCert.java:97) - Pkcert Version 1.0
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
Setting debug to info
 INFO [main] 2007-05-02 06:32:26,289 (PKCCreate.java:69) - PKCCreate startup
WARNING - Certificate File will be overwritten
 INFO [main] 2007-05-02 06:32:26,291 (PKCCreate.java:341) - WARNING - Certificate File
will be overwritten
```

```
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e.
/opt/CSCObpr/kdc/solaris/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObpr/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObpr/kdc/solaris)
```

Example 3

In this example, the log level is set for debugging.

Note The sample output has been trimmed for demonstration purposes.

```
# ./PKCert.sh -c "-s /var/certsInput
> -d /var/certsOutput
> -k /var/certsInput/Local_System.der
> -c /var/certsInput/Local_System.cer
> -r PCTEST.CISCO.COM
> -n 100
> -a kdc.pctest.cisco.com
> -o -z debug"
 INFO [main] 2007-05-02 06:32:06,029 (PKCert.java:97) - Pkcert Version 1.0
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: IPFONIX.COM
Serial Number: 100
DNS Name of KDC: bacdev3-dpe-4.cisco.com
Setting debug to debug
 INFO [main] 2007-05-02 06:32:06,038 (PKCCreate.java:69) - PKCCreate startup
WARNING - Certificate File will be overwritten
 INFO [main] 2007-05-02 06:32:06,039 (PKCCreate.java:341) - WARNING - Certificate File
will be overwritten
DEBUG [main] 2007-05-02 06:32:06,054 (PKCert.java:553) - Characters Read: 1218
DEBUG [main] 2007-05-02 06:32:06,056 (PKCert.java:583) - Binary File:
/var/certsInput/Local_System.der Read. Length: 1218
DEBUG [main] 2007-05-02 06:32:06,062 (PKCert.java:553) - Characters Read: 943
DEBUG [main] 2007-05-02 06:32:06,063 (PKCert.java:583) - Binary File:
/var/certsInput/Local_System.cer Read. Length: 943
DEBUG [main] 2007-05-02 06:32:06,064 (PKCert.java:455) - Jar File Path:
/opt/CSCObpr/lib/pkcerts.jar
DEBUG [main] 2007-05-02 06:32:06,065 (PKCert.java:456) - Opened jar file:
/opt/CSCObpr/lib/pkcerts.jar
DEBUG [main] 2007-05-02 06:32:06,067 (PKCert.java:460) - Jar entry unfiltered:
Tag_Packetcable_Tag/
DEBUG [main] 2007-05-02 06:32:06,068 (PKCert.java:460) - Jar entry unfiltered:
Tag_Packetcable_Tag/CableLabs_Service_Provider_Root.cer
...
DEBUG [main] 2007-05-02 06:32:06,115 (PKCert.java:472) - File:
Tag_Packetcable_Tag/Manu.cer
DEBUG [main] 2007-05-02 06:32:06,116 (PKCert.java:472) - File:
Tag_Packetcable_Tag/Service_Provider.cer
```

```
DEBUG [main] 2007-05-02 06:32:06,121 (PKCCreate.java:91) - Found 7 files in jar.
DEBUG [main] 2007-05-02 06:32:06,827 (KDCCert.java:98) - SP Cert subject name:
C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs Local System CA
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
DEBUG [main] 2007-05-02 06:32:07,687 (KDCCert.java:293) - Setting issuer to:
C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs Local System CA
DEBUG [main] 2007-05-02 06:32:07,699 (KDCCert.java:231) - DERVisibleToGeneral
org.bouncycastle.asn1.DERGeneralString@bd0b4ea6

DEBUG [main] 2007-05-02 06:32:07,700 (KDCCert.java:231) - DERVisibleToGeneral
org.bouncycastle.asn1.DERGeneralString@5035bc0

DEBUG [main] 2007-05-02 06:32:07,701 (KDCCert.java:231) - DERVisibleToGeneral
org.bouncycastle.asn1.DERGeneralString@5035bc0

DEBUG [main] 2007-05-02 06:32:07,703 (KDCCert.java:210) - DERCombineTagged [0] IMPLICIT
    DER ConstructedSequence
        ObjectIdentifier(1.3.6.1.5.2.2)
        Tagged [0]
            DER ConstructedSequence
                Tagged [0]
                    org.bouncycastle.asn1.DERGeneralString@5035bc0
                Tagged [1]
                    DER ConstructedSequence
                        Tagged [0]
                            Integer(2)
                        Tagged [1]
                            DER ConstructedSequence
                                org.bouncycastle.asn1.DERGeneralString@bd0b4ea6
                                org.bouncycastle.asn1.DERGeneralString@5035bc0

File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e.
/opt/CSCObpr/kdc/solaris/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObpr/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObpr/kdc/solaris)
```

Using the KeyGen Tool

The KeyGen tool is used to generate PacketCable service keys. The service keys are symmetric triple data encryption standard (triple DES or 3DES) keys (shared secret) required for KDC communication. The KDC server requires service keys for each of the provisioning FQDNs of the DPE. Any changes made to the DPE provisioning FQDN from the DPE CLI requires a corresponding change to the KDC service key filename. This change is necessary since the KDC service key uses the DPE provisioning FQDN as part of its filename.

The KeyGen tool, which resides in the *BAC_home*/kdc directory, uses command-line arguments for the DPE provisioning FQDN, realm name, and a password, and generates the service key files.

Note When running this tool, remember to enter the same password that you used to generate the service key on the DPE (by using the **packetCable registration kdc-service-key** command from the DPE CLI). For information on setting this password, refer to the *Cisco Broadband Access Center DPE CLI Reference, 2.7.1*.

The KDC server reads the service keys on startup. Any modification to the service keys requires that you restart the KDC server.

Syntax Description

keygen *options fqdn realm password*

- *options* are:
  - **-?**—Displays this usage message and exits the command.
  - **-v** or **-version**—Displays the version of this tool and exits the command.
  - **-q** or **-quiet**—Implements a quiet mode whereby no output is created.
  - **-c** or **-cms**—Creates a service key for the CMS system.
- *fqdn*—Identifies the FQDN of the DPE and is a required entry.
- *realm*—Identifies the Kerberos realm and is a required entry.
- *password*—Specifies the password to be used. This is also a required field. The password must be from 6 to 20 characters.

Three service key files are written in the KDC keys directory using this filename syntax:

mtafqdnmap,*fqdn@REALM*

mtaprovsrvr,*fqdn@REALM*

krbtgt,*REALM@REALM*

- *fqdn*—Identifies the FQDN of the DPE.
- *REALM*—Identifies the Kerberos realm.

The service key file always contains a version field of 0x0000.

Examples

```
# keygen dpe.cisco.com CISCO.COM changeme
```

When this command is implemented, these KDC service keys are written to the *BAC_home*/kdc/solaris/keys directory:

```
mtafqdnmap,dpe.cisco.com@CISCO.COM
mtaprovsrvr,dpe.cisco.com@CISCO.COM
krbtgt,CISCO.COM@CISCO.COM
```

Restart the KDC, so that the new keys are recognized. Use this BAC process watchdog command to restart the KDC:

```
# /etc/init.d/bprAgent restart kdc
```

This example illustrates the generation of a CMS service key:

```
# keygen -c cms-fqdn.com CMS-REALM-NAME changeme
```

When this command is implemented, this CMS service key is written to the *BAC_home*/kdc/solaris/keys directory.

```
cms,cms-fqdn.com@CMS-REALM-NAME
```

Verifying the KDC Service Keys

Once you generate the service keys on the KDC and the DPE, verify if the service keys match on both components.

The KeyGen tool requires you to enter the same password that you used to generate the service key on the DPE using the **packetCable registration kdc-service-key** command. Once you set this password on the DPE, you can view the service key from the dpe.properties file, which resides in the *BAC_home*/dpe/conf directory. Look for the value against the `/pktcbl/regsvr/KDCServiceKey=` property.

For example:

```
# more dpe.properties
/pktcbl/regsvr/KDCServiceKey=2e:d5:ef:e9:5a:4e:d7:06:67:dc:65:ac:bb:89:e3:2c:bb:
71:5f:22:bf:94:cf:2c
```

Note The output of this example has been trimmed for demonstration purposes.

To view the service key generated on the KDC, run the following command from the *BAC_home*/kdc/solaris/keys directory:

od -Ax -tx1 mtaprovsrvr,*fqdn@REALM*

- *fqdn*—Identifies the FQDN of the DPE.
- *REALM*—Identifies the Kerberos realm.

The output that this command generates should match the value of the `/pktcbl/regsvr/KDCServiceKey=` property in the dpe.properties file.

For example:

```
# od -Ax -tx1 mtaprovsrvr,dpe.cisco.com@CISCO.COM
0000000 00 00 2e d5 ef e9 5a 4e d7 06 67 dc 65 ac bb 89
0000010 e3 2c bb 71 5f 22 bf 94 cf 2c
000001a
```

In the examples shown here, note that the service key generated at the KDC matches the service key on the DPE.

Using the changeNRProperties.sh Tool

The BAC installation program establishes values for configuration properties used by BAC extensions that are incorporated into the Network Registrar DHCP server. You use the **changeNRProperties.sh** command, which is found in the *BAC_home*/cnr_ep/bin directory, to change key configuration properties.

Invoking the script without any parameters displays a help message listing the properties that can be set.

To run this command:

Step 1 Change directory to *BAC_home*/cnr_ep/bin.

Step 2 Run the **changeNRProperties.sh** command using this syntax:

changeNRProperties.sh *options*

Where *options* are:

- **-help**—Displays this help message. The -help option must be used exclusively. Do not use this in conjunction with any other option.
- **-e enabled | disabled**—Sets the PacketCable enable property. Enter **-e enabled** to enable the property, and **-e disabled** to disable it.
- **-d**—Displays the current properties. The -d option must be used exclusively. Do not use this in conjunction with any other option.
- **-s** *secret*—Identifies the BAC shared secret. For example, enter **-s secret**, if the shared secret is the word *secret*.
- **-f** *fqdn*—Identifies the RDU FQDN. For example, enter **-f rdu.cisco.com**, if you use rdu.cisco.com as the fully qualified domain name.
- **-p** *port*—Identifies the RDU port you want to use. For example, enter **-p 49187**, if you wanted to use port number 49187.
- **-r** *realm*—Identifies the PacketCable realm. For example, enter **-r CISCO.COM,** if your PacketCable realm is CISCO.COM.

  Note The realm must be entered in uppercase letters.

- **-g** *prov_group*—Identifies the provisioning group. For example, enter **-g group1**, if you are using provisioning group called group1.
- **-t 00 | 01**—Identifies whether or not the PacketCable TGT is set to off or on. For example, enter **-t 00** to set this to off, or **-t 01** to set it to on.
- **-a** *ip*—Identifies the PacketCable primary DHCP server address. For example, enter **-a 10.10.10.2** if the IP address of your primary DHCP server is 10.10.10.2.
- **-b** *ip*—Identifies the PacketCable secondary DHCP server address. For example, enter **-b 10.10.10.4**, if the IP address of your secondary DHCP server is 10.10.10.4. You can also enter **-b null** to set a null value, if appropriate.
- **-y** *ip*—Identifies the PacketCable primary DNS server address. For example, enter **-y 10.10.10.6**, if the IP address of the PacketCable primary DNS server is 10.10.10.6.
- **-z** *ip*—Identifies the PacketCable secondary DNS server address. For example, enter **-z 10.10.10.8**, if the IP address of your secondary DNS server is 10.10.10.8. You can also enter **-z null** to set a null value, if appropriate.
- **-o** *prov_ip man_ip*—Sets the management address to use for communication with the DPE identified by the given provisioning address. For example, **-o 10.10.10.7 10.14.0.4**, if the IP address of your provisioning group is 10.10.10.7. You can also enter a null value, if appropriate; for example, **-o 10.10.10.7 null**.

Step 3 Restart the DHCP server.

Examples

This is an example of changing the Network Registrar extensions by using the NR Extensions Properties tool:

```
# /opt/CSCObpr/cnr_ep_bin/changeNRProperties.sh -g primary1
Current NR Properties:
RDU Port: 49187
RDU FQDN: rdu.acme.com
Provisioning Group: primary1
Shared Secret: fggTaLg0XwKRs
PacketCable Enable: enabled
CableLabs client TGT: 01
CableLabs client Realm: ACME.COM
CableLabs client Primary DHCP Server: 10.10.1.2
CableLabs client Secondary DHCP Server: NOT SET
CableLabs client Primary DNS Server: 10.10.1.2
CableLabs client Secondary DNS Server: NOT SET
```

Note You must restart your NR DHCP server for the changes to take effect.

This is an example of viewing the current properties:

```
# /opt/CSCObpr/cnr_ep_bin/changeNRProperties.sh -d
Current NR Properties:
RDU Port: 49187
RDU FQDN: rdu.acme.com
Provisioning Group: primary1
Shared Secret: fggTaLg0XwKRs
PacketCable Enable: enabled
CableLabs client TGT: 01
CableLabs client Realm: ACME.COM
CableLabs client Primary DHCP Server: 10.10.1.2
CableLabs client Secondary DHCP Server: NOT SET
CableLabs client Primary DNS Server: 10.10.1.2
CableLabs client Secondary DNS Server: NOT SET
```

Using the snmpAgentCfgUtil.sh Tool

You can use the **snmpAgentCfgUtil.sh** tool to manage the SNMP agent installed on a Solaris computer. Using this tool, which resides in the *BAC_home*/snmp/bin directory, you can add (or remove) your host to a list of other hosts that receive SNMP notifications, and start and stop the SNMP agent process. This tool should be run from the local directory.

Note The default port number of an SNMP agent running on a Solaris computer is 8001.

You can use the RDU SNMP agent to:

- Adding a Host, page 13-16
- Deleting a Host, page 13-16
- Adding an SNMP Agent Community, page 13-17
- Deleting an SNMP Agent Community, page 13-17
- Starting the SNMP Agent, page 13-18
- Stopping the SNMP Agent, page 13-18
- Changing the SNMP Agent Location, page 13-19

- Setting Up SNMP Contacts, page 13-19
- Displaying SNMP Agent Settings, page 13-20

Adding a Host

You use this command to add the host address to the list of hosts that receive SNMP notifications from the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh add host *ip-addr* **community** *community* [**udp-port** *port*]

- *ip-addr*—Specifies the IP address of the host to which notifications are sent.
- *community*—Specifies the community (read or write) to be used while sending SNMP notifications.
- *port*—Identifies the UDP port used for sending the SNMP notifications.

Examples

```
# ./snmpAgentCfgUtil.sh add host 10.10.10.5 community trapCommunity udp-port 162
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Deleting a Host

You use this command to remove a host from the list of those receiving SNMP notifications from the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh delete host *ip-addr*

ip-addr—Specifies the IP address of the host that you want to delete from the list of hosts.

Examples

```
# ./snmpAgentCfgUtil.sh delete host 10.10.10.5
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Adding an SNMP Agent Community

You use this command to add an SNMP community string to allow access to the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh add community *string* [**ro** | **rw**]

- *string*—Identifies the SNMP community.
- **ro**—Assigns a read-only (**ro**) community string. Only **get** requests (queries) can be performed. The ro community string allows **get** requests, but no **set** operations. The NMS and the managed device must reference the same community string.
- **rw**—Assigns a read-write (**rw**) community string. SNMP applications require read-write access for **set** operations. The **rw** community string enables write access to OID values.

Note The default **ro** and **rw** community strings are `baccread` and `baccwrite`, respectively. We recommend that you change these values before deploying BAC.

Examples

```
# ./snmpAgentCfgUtil.sh add community fsda54 ro
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Deleting an SNMP Agent Community

You use this command to delete an SNMP community string to prevent access to the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh delete community *string* [**ro** | **rw**]

- *string*—identifies the SNMP community
- **ro**—assigns a read only (ro) community string
- **rw**—assigns a read write (rw) community string

Note See Adding an SNMP Agent Community, page 13-17, for additional information on the **ro** and **rw** community strings.

Examples

```
# ./snmpAgentCfgUtil.sh delete community fsda54 ro
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Starting the SNMP Agent

You use this command to start the SNMP agent process on a Solaris computer on which BAC is installed.

Note You can also start the SNMP agent by invoking the BAC process watchdog using the **/etc/init.d/bprAgent start snmpAgent** command. For more information, see Using the BAC Process Watchdog from the Command Line, page 2-14.

Examples

```
# ./snmpAgentCfgUtil.sh start
Process snmpAgent has been started
```

Stopping the SNMP Agent

You use this command to stop the SNMP agent process on a Solaris computer on which BAC is installed.

Note You can also start the SNMP agent by invoking the BAC process watchdog using the **/etc/init.d/bprAgent stop snmpAgent** command. For more information, see Using the BAC Process Watchdog from the Command Line, page 2-14.

Examples

```
# ./snmpAgentCfgUtil.sh stop
Process snmpAgent has stopped
```

Configuring an SNMP Agent Listening Port

You use this command to specify the port number that the SNMP agent will listen to. The default port number used by RDU SNMP agent is 8001.

Syntax Description

snmpAgentCfgUtil.sh udp-port *port*

port identifies the port number that the SNMP agent will listen to.

Examples

```
# ./snmpAgentCfgUtil.sh udp-port 8001
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Changing the SNMP Agent Location

You use this command to enter a string of text that indicates the location of the device running the SNMP agent. This could, for example, be used to identify the physical location of the device. You can enter any character string that is fewer than 255 characters.

Syntax Description

snmpAgentCfgUtil.sh location *location*

location is the character string identifying the agent's location.

Examples

In this example, the physical location of the SNMP agent is in an equipment rack identified as rack 5D:

```
# ./snmpAgentCfgUtil.sh location "equipmentrack5D"
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Setting Up SNMP Contacts

You can use this command to enter a string of text that identifies the contact person for the SNMP agent, together with information on how to contact this person. This could, for example, be used to identify a specific person including that person's telephone number. You can enter any character string that is fewer than 255 characters.

Syntax Description

snmpAgentCfgUtil.sh contact *contact-info*

contact-info is the character string identifying the individual to contact concerning the SNMP agent.

Examples

In this example, the contact name is Terry and the telephone extension is 1234:

```
# ./snmpAgentCfgUtil.sh contact "Terry-ext1234"
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Displaying SNMP Agent Settings

You use this command to display all current SNMP settings.

Examples

```
# ./snmpAgentCfgUtil.sh show
Location                      : equipmentrack5D
Contact                       : Terry-ext1234
Port Number                   : 8001
Notification Type             : trap
Notification Recipient Table  :
        [ Host IP address, Community, UDP Port ]
        [ 10.10.10.5 , trapCommunity , 162 ]
Access Control Table          :
        Read Only Communities
                baccread
        Read Write Communities
                baccwrite
```

Specifying SNMP Notification Types

You use this command to specify the types of notifications (traps or informs) that will be sent from the SNMP agent. By default, traps are sent, though you can set this to send SNMP informs instead.

Syntax Description

snmpAgentCfgUtil.sh inform [**retries timeout**] | **trap**

Where the parameter is the backoff timeout between retries.

Examples

```
# ./snmpAgentCfgUtil.sh inform retries 3 timeout 1000
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see BAC Process Watchdog, page 2-14.

Use the **snmpAgentCfgUtil.sh show** command to verify your configuration settings.

```
# ./snmpAgentCfgUtil.sh show
Location                      : equipmentrack5D
Contact                       : Terry-ext1234
Port Number                   : 8001
Notification Type             : inform
Notification Retries          : 3
Notification Timeout          : 1000
Notification Recipient Table  :
        [ Host IP address, Community, UDP Port ]
        [ 10.10.10.5 , trapCommunity , 162 ]
Access Control Table          :
        Read Only Communities
                baccread
        Read Write Communities
```

```
baccwrite
```

Using the disk_monitor.sh Tool

Monitoring available disk space is an important system administration task. You can use a number of custom written scripts or commercially available tools to do so.

The **disk_monitor.sh** command, which resides in the *BAC_home*/rdu/samples/tools directory, sets threshold values for one or more file systems. When these thresholds are surpassed, an alert is generated through the Solaris syslog facility, at 60-second intervals, until additional disk space is available.

Note We recommend that, at a minimum, you use the **disk_monitor.sh** script to monitor the *BAC_data* and *BAC_dblog* directories.

Syntax Description

disk_monitor.sh *filesystem-directory x [filesystem-directory* x*]*

- *filesystem-directory*—Identifies any directory in a file system to monitor.
- *x*—Identifies the percentage threshold applied to the specified file system.
- *filesystem-directory**—Identifies multiple file systems.
- *x**—Specifies percentage thresholds to be applied to multiple file systems.

Examples

Example 1

This example specifies that a notification be sent out when the */var/CSCObpr* file system reaches 80 percent of its capacity.

```
# ./disk_monitor.sh /var/CSCObpr 80
```

When the database logs disk space reaches 80-percent capacity, an alert similar to the following one is sent to the syslog file:

```
Dec 7 8:16:06 perf-u80-1 BPR: [ID 702911 local6.warning] File system /var/bpr usage is 81%
(threshold is 80%)
```

Example 2

This example describes how you can run the disk_monitor.sh tool as a background process. Specifying an ampersand (&) at the end of the command immediately returns output while running the process in the background.

```
# ./disk_monitor.sh /var/CSCObpr 80 &
1020
```

Troubleshooting Devices by MAC Address

You can use this feature to collect detailed diagnostics about one or more specific devices. Troubleshooting information includes all server interactions related to a given device or a group of devices. This information includes administrator user interface operations, RDU API operations, DPE interactions with devices, and interserver DPE-to-RDU interactions.

You can enable or disable troubleshooting via node management for one or more specific devices without turning logging on, and without searching through log files for specific device information.

BAC maintains a list of devices, based on MAC addresses, for which detailed diagnostics are collected. Troubleshooting information is stored centrally at the RDU and is maintained on a per-device basis. Neither DPEs nor Network Registrar extensions store this data. Rather, they forward this information to the RDU, which, upon receiving information, writes it to the appropriate device log file. If the connection from the DPE or Network Registrar extension to the RDU is lost, any new troubleshooting events occurring on the DPE or Network Registrar extension are discarded. The logging of troubleshooting information resumes only after the connection to the RDU is restored.

The DPE maps all device MAC addresses to its IP address mapping for that device, and the Network Registrar extensions send the IP update to the DPE whenever the extensions determine that device troubleshooting is enabled.

Any modifications to the device tracking list, such as the addition of a new device or a group, take place immediately at all servers; you do not have to reboot the RDU or the DPE. The log files on the respective servers list the current list of devices in the troubleshooting mode.

Caution Additional memory and disk space is required whenever the device troubleshooting feature is used. As the number of tracked devices increases, so does the amount of memory and disk space that is required to support the number of logs that are created.

The device troubleshooting feature is disabled until one or more devices are set in troubleshooting mode.

Note To enable diagnostics for a device, the device must be preregistered in the BAC RDU. If the device is not yet preregistered, add the device from the Manage Devices page by clicking the Add button. For information on adding devices, see Adding Device Records, page 10-14.

You can configure a maximum number of devices in diagnostics mode to prevent inadvertently putting too many devices in this mode and thus diminishing server performance. By default, this number is set at 100. To configure the maximum number of devices allowed in troubleshooting mode from the administrator user interface, click the Systems Defaults page via the **Configuration > Defaults** tabs. Enter a value in the Maximum Diagnostics Device Count field.

Relating a Device to a Node

You can troubleshoot a device by relating it to a specific node. Use the Relate function to associate a device, using its MAC address, to a specific node, which is in turn associated with a specific node type. (See Relating and Unrelating Devices, page 10-16.) Doing so records an extraordinarily large volume of information for a device; you can then use the information to troubleshoot potential problems.

Table 13-3 identifies a possible workflow using the Relate and Unrelate functions.

Table 13-3 Sample Relate/Unrelate Process

Step	Action
1.	Determine whether or not a problem exists and identify which devices are affected.
2.	Relate the devices to a node.
3.	Wait a few minutes to ensure that device traffic is passing, or perform a hard boot of the device.

Table 13-3 Sample Relate/Unrelate Process (continued)

Step	Action
4.	Open the rdu.log file in a word processing application and locate the entries for the MAC address of the specific device.
5.	Identify, correct, test, and verify the problem.
6.	Unrelate the device from the node.

Viewing a List of the Devices in Troubleshooting Mode

When you enable troubleshooting for a device, the device is automatically added to a special device node that contains a list of devices in troubleshooting mode. The node type is **system** and the node name is **diagnostics**. You can access the list of devices in this group from the API or the administrator user interface.

To view a list of devices currently enabled for troubleshooting:

Step 1 From the Manage Devices page, click the Search Type drop-down list and select Node Search.

Step 2 From the Node Name (Node Type) drop-down list, select the diagnostics (system) option to view all the devices in troubleshooting mode.

Step 3 Click **Search**.

Note An alternative way to view the list of devices in troubleshooting mode is to consult the RDU log (rdu.log) and the DPE log (dpe.log) files. The list of devices is logged whenever the server is started and whenever there is a change in the list of devices enabled for diagnostics.

The devices enabled for troubleshooting appear in the log files with the log level of 5-notification. For details on log files, see Logging, page 2-15.

Examples

This example features log output while troubleshooting an MTA:

```
bac-test.cisco.com:2005 03 04 18:38:24 EST:%BPR-DIAGNOSTICS-3-4055:[##MTA-9a Unconfirmed
FQDN Request Received from [/10.10.10.5 ['kdcquery']. Client with IP Address [10.10.20.2]
and MAC Address [1,6,00:00 :ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:24 EST:%BPR-DIAGNOSTICS-3-4082:[Results of BACC
Lookup. FQDN: [1-6-00-00-ca-b7-7e-91.cisco.com MAC: 1,6,00:00:ca:b7:7e:91. Client with IP
Address [10.10.20.2] and MAC Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:24 EST:%BPR-DIAGNOSTICS-3-4070:[##MTA-9b FQDN Reply
Sent to [/10.10.20.2(41142) for MTA 1,6,00:00:ca:b7:7e:91. Client with IP Address
[10.10.20.2] and MAC Address [1,6, 00:00:ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-4132:[##MTA-13 Incoming
APREQ received from [/10.10.20.2:1293. Client with IP Address [10.10.20.2] and MAC
Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-4141:[##MTA-13 APREP sent to
[/10.10.20.2(1293) For MTA 1,6,00:00:ca:b7:7e:91. Client with IP Address [10.10.20.2] and
MAC Address [1,6,00:00: ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-0764:[[##MTA-15 SNMPv3
INFORM Received From 10.10.20.2. Client with IP Address [10.10.20.2] and MAC Address
[1,6,00:00:ca:b7:7e:91]]]
```

```
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-0764:[[##MTA-19 SNMPv3 SET
Sent to 10.10.20.2. Client with IP Address [10.10.20.2] and MAC Address
[1,6,00:00:ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-1092:[Received a TFTP [read]
request from [10.10.20.2:1271] for [bpr01060000cab77e910002]; Client with MAC Address
[1,6,00:00:ca:b7:7e:91] and IP Address [10.10.20.2]]
bac-test.cisco.com:2005 03 04 18:38:26 EST:%BPR-DIAGNOSTICS-3-1155:[##MTA-23 Finished
handling [read] request from [10.10.20.2:1271] for [bpr01060000cab77e910002]; Transferred
[236] bytes to Client with MAC Address [1,6,00:00:ca:b7:7e:91] and IP Address
[10.10.20.2]]
bac-test.cisco.com:2005 03 04 18:38:27 EST:%BPR-DIAGNOSTICS-3-0764:[[##MTA-25 SNMP
Provisioning State INFORM Received from 10.10.20.2. Client with IP Address [10.10.20.2]
and MAC Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.cisco.com:2005 03 04 18:38:27 EST:%BPR-DIAGNOSTICS-3-0764:[[MTA Configuration
Confirmed, Returned 'pass' as the final MTA provisioning state for 10.10.20.2. Client with
IP Address [10.10.20.2] and MAC Address [1,6,00:00:ca:b7:7e:91]]]
```