Broadband Access Center for Cable Support Tools and Advanced Concepts

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

This chapter discusses these topics:

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

Developing Template Files

BACC uses templates to help administrators deploy dynamic PacketCable, DOCSIS, and CableHome files. Using templates, you can create a template file in an easily readable format, and edit it quickly and simply. A template is an ASCII text file that represents the PacketCable, DOCSIS, or CableHome options and values used for generating a valid PacketCable, DOCSIS, or CableHome file. BACC uses the .tmpl file extension to identify template files. You must add template files to the RDU as an external file using either the administrator's user interface or the API, before any class of service can reference it.

When installing the BACC RDU component, several sample template files are copied to the <BACC_HOME>/rdu/samples directory.

Although all that you need to create or edit a template is a simple text editor, before attempting to create your own template file, you should thoroughly familiarize yourself with this information:

- BACC provisioning flows
- DOCSIS 1.0, 1.1, and 2.0 RFI specifications
- PacketCable 1.0 and 1.1 specifications
- MTA device provisioning specification
- CableHome 1.0 specification
- SNMP MIBs for cable devices (for example, DOCS-CABLE-DEVICE-MIB)

Template Grammar

A template is comprised of four different types of statements:

- Comments, page 12-3
- Includes, page 12-3
- Options, page 12-3
- Instance Modifier, page 12-5

Comments allow you to document your templates. Includes allow you to create building block templates to be used in other templates. Options allow you to specify the PacketCable, DOCSIS, or CableHome type length value (TLV) in a descriptive manner. Table 12-1 describes the available template grammar options.

Table 12-1 Template Grammar

Option	Description
<comment>	::= #[ascii-string]
<include>	::= include “<filename.tmpl>”
<option-description>	::= option <option-num> [instance <instance-num>] <option-value>
<option-num>	::= <unsigned-byte>[.<unsigned-byte>]*
<option-value>	::= <well-defined-value> | <custom-value>
<well-defined-value>	::= <option-value-string>[,<option-value-string>]*
<custom-value>	::= <ascii-value> | <hex-value> | <ip-value> | <snmp-value>
<ascii-value>	::= ascii <ascii-string>
<hex-value>	::= hex <hex-string>
<ip-value>	::= ip <ip-string>
<instance-num>	::= <unsigned integer>
<template>	::= <template-statement>*
<template-statement>	::= <comment> | <include> | <option-description>
<snmp-value>	::= snmp <snmpvar-oid>,<snmpvar-type>,<snmpvar-value>

Comments

Comments provide information only and are always located between the pound (#) symbol and the end of a line. Example 12-1 shows example comment usage.

Example 12-1 Example Comment Usage

```
#
# Template for gold service
#

option 3 1 # enabling network access
```

Includes

Include files let you build a hierarchy of similar, but slightly different, templates. This is very useful for defining options that are common across many service classes without having to duplicate the options in several templates.

You can use multiple include statements in a single template although the location of the include statement in the template is significant; the contents of the include file are included wherever the include statement is found in the template. The included template must be added as an external file to the RDU before it can be used. The included file must not contain any location modifiers such as ../.. because the templates are stored without path information in the RDU database. Examples 12-2 and 12-3 illustrate both correct and incorrect usage of the include option.

Example 12-2 Correct Include Statement Usage

```
# Valid, including common options
include "common_options.tmpl"
```

Example 12-3 Incorrect Include Statement Usage

```
# Invalid, using location modifier
include "../common_options.tmpl"

# Invalid, using incorrect file suffix
include "common_options.common"

# Invalid, not using double quotes
include common_options.tmpl
```

Options

PacketCable, DOCSIS, and CableHome configuration files consist of properly encoded option id-value pairs. Two forms of options are supported: defined and custom.

- Well-defined options require the option number and value. The value is encoded based on the encoding type of the option number.
- Custom options require the option number, explicit value encoding type, and the value.

When using compound options, for example, option 43, you can use the instance modifier to specify the TLV groupings. See the "Instance Modifier" section on page 12-5 for additional information.

When specifying one of these well-defined options in a template, it is not necessary to specify a value encoding for the value. See the "Encoding Types for Defined Options" section on page 12-8 and the "DOCSIS Option Support" section on page 12-12 for additional information on these defined encoding types.

When specifying custom options (e.g. option 43), you must specify the encoding type for the option. The available encoding types are:

- ASCII— ASCII type encodes any given value as an ASCII string without a `NULL` terminator. If the value contains spaces, they must be double quoted.
- hex—The value must be valid hexadecimal and there must be exactly 2 characters for each octet. If `01` is specified as the value, then exactly one octet is used in the encoding. If `0001` is specified as the value, then exactly two octets are used in the encoding process.
- IP address—IP address type encodes any given value as 4 octets. For example, the IP address `10.10.10.1` is encoded as `0A0A0A01`.

Use a comma to separate multi-valued options on a given line. Each value is treated as such, so you might have to double quote one of the values, but not the others. A good example of a multi-valued option is Option 11 (SNMP Varbind). See the "SNMP Varbind" section on page 12-5 for additional information.

When specifying compound options, there is no need to specify the top level option (for example option 4 when specifying option 4.1). Examples 12-4 and 12-5 illustrate both correct and incorrect usage of the option statement.

Example 12-4 Correct Option Statement Usage

```
# Valid, specifying the number for well known option 3
option 3 1

# Valid, specifying the number for option 4 sub-option 1
option 4.1 1

# Valid, specifying a vendor option as hex
option 43.200 hex 00000C

# Valid, specifying a vendor option as ascii
option 43.201 ascii "enable log"

# Valid, specifying a vendor option as IP
option 43.202 ip 10.4.2.1
```

Example 12-5 Incorrect Option Statement Usage

```
# Invalid, using hex with incorrect hex separator
option 43.200 hex 00.00.0C

# Invalid, not using double quotes when needed
option 43.201 ascii enable log

# Invalid, not specifying IP address correctly
option 43.202 ip 10-10-10-1

# Invalid, specifying the description for option "Network Access Control"
option "Network Access Control" 1

# Invalid, specifying top level option
option 4
```

Instance Modifier

The instance modifier is used to group compound options into specific individual Tag-Length-Values (TLVs). Examples 12-6 and 12-7 illustrate both correct and incorrect methods of creating separate TLVs. These are required to enable the IOS DOCSIS modem to interpret the IOS commands as two separate commands.

Example 12-6 Correct IOS Command Line Entries

```
# Valid, each IOS command gets its own TLV
option 43.8 instance 1 00-00-0C
option 43.131 instance 1 ascii "login"
option 43.8 instance 2 00-00-0C
option 43.131 instance 2 ascii "password cable"
```

Example 12-7 Incorrect IOS Command Line Entries

```
# Invalid, IOS commands are grouped into one TLV
option 43.8 00-00-0C
option 43.131 ascii "login"
option 43.131 ascii "password cable"

# Invalid, using instance on non-compound options
option 3 instance 1 1
```

Note The encoding type for option 43.8 is an organizationally unique identifier (OUI). Unlike that shown in Example 12-4, this only accepts an 00-00-0C format.

SNMP Varbind

You must use an object identifier (OID), when specifying DOCSIS option 11, PacketCable option 64, or CableHome option 28. The MIB that contains the OID must be in one of the following MIBs loaded by the RDU. You must specify as much of the OID as needed to uniquely identify it. You can use the name or the number of the OID. The RDU automatically loads these MIBs:

- SNMPv2-SMI
- SNMPv2-TC
- CISCO-SMI
- CISCO-TC
- SNMPv2-MIB
- RFC1213-MIB
- IANAifType-MIB
- IF-MIB

DOCSIS MIBs

These DOCSIS MIBs are loaded into the RDU:

- DOCS-IF-MIB
- DOCS-BPI-MIB

- CISCO-CABLE-SPECTRUM-MIB
- CISCO-DOCS-EXT-MIB
- SNMP-FRAMEWORK-MIB
- DOCS-CABLE-DEVICE-MIB
- DOCS-CABLE-DEVICE-MIB-OBSOLETE
- CISCO-CABLE-MODEM-MIB

Two versions of the DOCS-CABLE-DEVICE MIB are loaded into the RDU:

- DOCS-CABLE-DEVICE-MIB-OBSOLETE (experimental branch)
- DOCS-CABLE-DEVICE-MIB (mib2 branch)

A fully-qualified MIB OID (.experimental...) always uniquely identifies a MIB OID.

If you use a non-fully qualified MIB OID from DOCS-CABLE-DEVICE-MIB, it will always default to DOCS-CABLE-DEVICE-MIB and not DOCS-CABLE-DEVICE-MIB-OBSOLETE.

Examples 12-8 and 12-9 illustrate using a fully-qualified MIB OID and a non-fully qualified MIB OID.

Example 12-8 Fully Qualified MIB OID

```
# Valid, uniquely identifying an OID
option 11 .experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccess
Entry.docsDevNmAccessStatus.1, Integer, 4
```

Example 12-9 Non-Fully Qualified MIB OID (Defaults to DOCS-CABLE-DEVICE-MIB)

```
# Valid, Non-Fully Qualified MIB OID.
option 11 .docsDevNmAccessStatus.1, Integer, 4
```

If no DOCSIS CMs in a deployment require DOCS-CABLE-DEVICE-MIB-OBSOLETE, you can always use the shorter form of the MIB OID.

PacketCable MIBs

These PacketCable (North American) MIBs are loaded into the RDU:

- CLAB-DEF-MIB
- PKTC-MTA-MIB
- PKTC-SIG-MIB
- PKTC-EVENT-MIB

CableHome MIBs

These CableHome MIBs are loaded into the RDU:

- CABH-CAP-MIB
- CABH-CDP-MIB
- CABH-CTP-MIB
- CABH-PS-DEV-MIB

- CABH-QOS-MIB
- CABH-SEC-MIB

These additional MIBs are needed but are not part of the BACC product:

- CABH-CTP-MIB needs RMON2-MIB, TOKEN-RING-RMON-MIB
- CABH-SEC-MIB needs DOCS-BPI2-MIB.

Macro Variables

Macro variables are specified as values in templates that let you specify device-specific option values. When a macro variable is encountered in the template, the properties hierarchy is searched for the macro variable name and the value of the variable is then substituted. The variable name is a custom property, which is predefined in the RDU. It must not contain any spaces.

Note When you use custom properties for macro variables, you must use DataType.STRING.

After the custom property is defined, it can be used in this property hierarchy:

- System defaults
- Technology defaults, such as PacketCable, DOCSIS, or CableHome
- DHCP criteria properties
- Class of service properties
- Device properties

The template parser works bottom up when locating properties in the hierarchy (device first, then the class of service, and so on) and converts the template option syntax. The following syntax is supported for macro variables:

- ${var-name}—This syntax is a straight substitution. If the variable is not found, the parser will generate an error.
- ${var-name, ignore}—This syntax lets the template parser ignore this option if the variable value is not found in the properties hierarchy.
- ${var-name, default-value}—This syntax provides a default value if the variable is not found in the properties hierarchy.

Examples 12-10 and 12-11 illustrate both correct and incorrect usage of option 11.

Example 12-10 Correct Macro Variables Usage

```
# Valid, using macro variable for max CPE's, straight substitution
option 18 ${MAX_CPES}

# Valid, using macro variable for max CPE's, ignore option if variable not found
# option 18 will not be defined in the DOCSIS configuration file if MAX_CPES
# is not found in the properties hierarchy
option 18 ${MAX_CPES, ignore}

# Valid, using macro variable for max CPE's with a default value
option 18 ${MAX_CPES, 1}
```

```
# Valid, using macro variable for vendor option
option 43.200 hex ${MACRO_VAR_HEX}

# Valid, using macro variable for vendor option
option 43.201 ascii ${MACRO_VAR_ASCII}

# Valid, using macro variable for vendor option
option 43.202 ip ${MACRO_VAR_IP}

# Valid, using macro variable in double quotes
option 18 "${MAX_CPES}"

# Valid, using macro variable within a value
option 43.131 ascii "hostname ${HOSTNAME}"

# Valid, using macro variables in multi-valued options
option 11 ${ACCESS_CONTROL_MIB,
.mib-2.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.docsDevNmAccess
Control.1}, Integer, ${ACCESS_CONTROL_VAL, 3}

# Valid, using macro variable in an include statement
include "${EXTRA_TEMPLATE}"

# Valid, using macro variable in an include statement with a default value
include "${EXTRA_TEMPLATE, modem_reset.tmpl}"

# Valid, using macro variable in an include statement with a default value
include "${EXTRA_TEMPLATE, modem_reset}.tmpl"

# Valid, using macro variable in an include statement with an ignore clause
include "${MY_TEMPLATE, ignore}"
```

Example 12-11 Incorrect Macro Variables Usage

```
# Invalid, using macro variable as the option number
option ${MAX_CPES} 1

# Invalid, using macro variable with space in name
option 18 ${MAX CPES}
```

Encoding Types for Defined Options

Table 12-2 identifies the options with defined encoding types.

Table 12-2 Defined Option Encoding Types

Encoding	Input	Example
Boolean	0 for false and 1 for true.	`0`
Bytes	A series of hexadecimal octets. Each octet must be exactly 2 characters in length.	`000102030405060708`
IP Address	Four unsigned integer 8, dot (`.`) separated.	`10.10.10.1`
Multiple IP Addresses	Comma separated list of IP addresses.	`10.11.12.13,10.11.12.14`

MAC Address	Six hexadecimal octets colon (`:`) or dash (`-`) separated. Each octet must be exactly 2 characters in length. Colons and dashes must not be mixed.	`00:01:02:03:04:05`
		or
		`00-01-02-03-04-05`

Table 12-2 Defined Option Encoding Types (continued)

Encoding	Input	Example

MAC Address And Mask	Twelve octets colon (`:`) or dash (`-`) separated. Each octet must be exactly 2 characters in length. Colons and dashes must not be mixed. The first six octets represent the MAC address; the last six represent the mask for the MAC address.	`00:01:02:03:04:05:06:07:08:09:0A:0B`
		or
		`00-01-02-03-04-05-06-07-08-09-0A-0B`

NVTASCII	An ASCII string. The encoded string will not be `NULL` terminated.	`This is an ASCII string`
OID	An SNMP OID string.	`sysinfo.0`
OIDCF	An SNMP OID string and an unsigned integer (`0` or `1`) comma separated.	`sysinfo.0,1`
OUI	Three hexadecimal octets colon (`:`) or dash (`-`) separated. Each octet must be exactly 2 characters in length.	`00-00-0C`
SNMPVarBind	An SNMP OID string, type, and value. Each of these is comma separated. Valid types are: • BITS • Counter • Counter32 • Counter64 • Gauge • Gauge32 • INTEGER • Integer32 • IpAddress • OCTETSTRING • OBJECTIDENTIFIER • Opaque • TimeTicks • Unsigned32 **Note** The OCTETSTRING can be either a string that will be converted to hexadecimal notation without a trailing NULL, octet string for example, or hexadecimal notation contained in single quotes, 'aa:bb:cc' for example.	`.experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.docsDevNmAccessStatus.1, INTEGER, 4`

Sub Type	One or two comma separated unsigned integer 8.	`12`
		or
		`12,14`

Table 12-2 Defined Option Encoding Types (continued)

Encoding	Input	Example
Unsigned integer 8	0 to 255	`14`
Unsigned integer 16	0 to 65535	`1244`
Unsigned integer 32	0 to 4294967295	`3455335`
Unsigned integer 8 and unsigned integer 16	One unsigned integer 8 and one unsigned integer 16, comma separated.	`3,12324`
Unsigned integer 8 pair	Two unsigned integer 8, comma separated.	`1,3`
Unsigned integer 8 triplet	Three unsigned integer 8, comma separated.	`1,2,3`
ZTASCII	An ASCII string. The encoded string will be `NULL` terminated.	`This is an ASCII string`

BITS Value Syntax

When using the BITS type, you must specify either the labels (“interval1 interval2 interval3”) or numeric bit location (“0 1 2”). Note that label values are 1-based and bit values are 0-based.

This is the syntax that uses the bit numbers:

```
option 11 .pktcSigDevR0Cadence.0,STRING,"0 1 2 3 4 5 6 7 8 9 10 11 12 13 14"
```

This is the syntax for the customer octet string (FFFE000000000000) that uses the labels:

```
option 11 .pktcSigDevR0Cadence.0,STRING,"interval1 interval2 interval3
interval4 interval5 interval6 interval7 interval8 interval9 interval10
interval11 interval12 interval13 interval14 interval15"
```

OCTETSTRING Syntax

The OCTETSTRING can be either a string that is converted to hexadecimal notation without a trailing NULL (for example, octet string), or hexadecimal notation contained within single quotes, (for example, 'aa:bb:cc').

Adding SNMP TLVs Without a MIB

You can add SNMP TLVs in dynamic configuration files (DOCSIS, PacketCable, CableHome) without requiring the MIB be loaded by the RDU. From within RDU configuration extensions, the functionality can be accessed with the DOCSISOptionFactory interface, using the following method:

```
public OptionValue createOptionValue(OptionSyntax syntax, String optionNumStr, String[]
optionValueList)
```

The public OptionSyntax.SNMP enumerated value can be used in the above method, in conjunction with the optionValueList containing the tuple: OID, Type, Value.

From RDU dynamic configuration templates, the following syntax is used to specify SNMP TLVs that are not validated against the RDU MIBs:

```
option <option-number> snmp <OID>, <Type>, <Value>
```

Examples:

```
# DOCS-CABLE-DEVICE-MIB:
option 11 snmp .docsDevNmAccessIp.1,IPADDRESS,192.168.1.1

# Arris vendor specific SNMP TLV (OID numbers only, mix names/numbers)
option 11 snmp .1.3.6.1.4.1.4115.1.3.1.1.2.3.2.0, INTEGER, 6
option 11 snmp .enterprises.4115.1.3.1.1.2.3.2.0, INTEGER, 6


# NOTE: trailing colon required for single octet
option 11 snmp .1.3.6.1.2.1.69.1.2.1.6.3, STRING, 'c0:'
```

The allowed SNMP variable type names are:

IETF standard SMI Data Type	SNMP API name
Integer32	INTEGER
Integer (Enumerated)	INTEGER
Unsigned32	UNSIGNED32
Gauge32	GAUGE
Counter32	COUNTER
Counter64	COUNTER64
Timeticks	TIMETICKS
OCTET STRING	STRING
OBJECT IDENTIFIER	OBJID
IpAddress	IPADDRESS
BITS	STRING

For example, to specify a SMI Integer32 type, the following types are accepted (regardless of case sensitivity): Integer32, INTEGER.

For OCTET STRING type, all of the following types are accepted: OCTET STRING, OCTETSTRING, or STRING.

The custom SNMP TLV template option can be used to specify any SNMP TLV, including those that are present in the RDU MIBs. The custom SNMP TLV error checking is less stringent, and will not detect incorrect scalar/columnar references (for example, .0 vs. .n in OID names).

DOCSIS Option Support

Table 12-3 describes DOCSIS options and identifies the specific version support for each option.

Table 12-3 DOCSIS Options and Version Support

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
0	PAD	No length and no value	N/A	True	✓	✓	✓
1	Downstream Frequency	Unsigned integer 32	Multiples of 62500	False	✓	✓	✓
2	Upstream Channel ID	Unsigned integer 8	None	False	✓	✓	✓
3	Network Access Control	Boolean	None	False	✓	✓	✓
4	Class of Service	Compound	None	False	✓	✓	✓
4.1	Class ID	Unsigned integer 8	Between 1-16 inclusive	False	✓	✓	✓
4.2	Maximum Downstream Rate	Unsigned integer 32	None	False	✓	✓	✓
4.3	Maximum Upstream Rate	Unsigned integer 32	None	False	✓	✓	✓
4.4	Upstream Channel Priority	Unsigned integer 8	Less than 8	False	✓	✓	✓
4.5	Guaranteed Minimum Upstream Channel Data Rate	Unsigned integer 32	None	False	✓	✓	✓
4.6	Maximum Upstream Channel Transmit Burst	Unsigned integer 16	None	False	✓	✓	✓
4.7	Class-of-Service Privacy Enable	Boolean	None	False	✓	✓	✓
6	CM MIC Configuration Setting	Byte 16	None	False	✓	✓	✓
7	CMTS MIC Configuration Setting	Byte 16	None	False	✓	✓	✓
9	Software Upgrade Filename	NVTASCII	None	True	✓	✓	✓
10	SNMP Write-Access Control	OIDCF	None	True	✓	✓	✓
11	SNMP MIB Object	SNMPVarBind	None	True	✓	✓	✓
14	CPE Ethernet MAC Address	MAC Address	None	True	✓	✓	✓
15	Telephony Settings Option	NVTASCII	None	False	✓	✓	✓
15.2	Service Provider Name	NVTASCII	None	False	✓	✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
15.3	Telephone Number (1)	NVTASCII	None	False	✓	✓	✓
15.4	Telephone Number (2)	NVTASCII	None	False	✓	✓	✓
15.5	Telephone Number (3)	NVTASCII	None	False	✓	✓	✓
15.6	Connection Threshold	Unsigned integer 8	None	False	✓	✓	✓
15.7	Login Username	NVTASCII	None	False	✓	✓	✓
15.8	Login Password	NVTASCII	None	False	✓	✓	✓
15.9	DHCP Authentication	Boolean	None	False	✓	✓	✓
15.10	DHCP Server	IP Address	None	False	✓	✓	✓
15.11	RADIUS realm	NVTASCII	None	False	✓	✓	✓
15.12	PPPAuthentication	Unsigned integer 8	None	False	✓	✓	✓
15.13	Demand Dial Inactivity Timer Threshold	Unsigned integer 8	None	False	✓	✓	✓
16	SNMP IP Address (No Longer Used)	IP Address	None	False	✓	✓	✓
17	Baseline Privacy Configuration Setting	Compound	None	False	✓	✓	✓
17.1	Authorize Wait Timeout	Unsigned integer 32	Between 1 and 30 inclusive	False	✓	✓	✓
17.2	Reauthorize Wait Timeout	Unsigned integer 32	Between 1 and 30 inclusive	False	✓	✓	✓
17.3	Authorization Grace Time	Unsigned integer 32	Between 1 and 1800 inclusive	False	✓		
17.3	Authorization Grace Time	Unsigned integer 32	Between 1 and 6047999 inclusive	False		✓	✓
17.4	Operational Wait Timeout	Unsigned integer 32	Between 1 and 10 inclusive	False	✓	✓	✓
17.5	Rekey Wait Timeout	Unsigned integer 32	Between 1 and 10 inclusive	False	✓	✓	✓
17.6	TEK Grace Time	Unsigned integer 32	Between 1 and 1800 inclusive	False	✓		
17.6	TEK Grace Time	Unsigned integer 32	Between 1 and 302399 inclusive	False		✓	✓
17.7	Authorize Reject Wait Timeout	Unsigned integer 32	Between 1 and 600 inclusive	False	✓	✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
17.8	SA Map Wait Timeout	Unsigned integer 32	Between 1 and 18006 inclusive	False		✓	✓
17.9	Maximum Clock Drift	Unsigned integer 32	Between 1 and 10 inclusive	False		✓	✓
18	Maximum Number of CPEs	Unsigned integer 32	Greater than 0	False	✓	✓	✓
19	TFTP Server Timestamp	Unsigned integer 32	None	False	✓	✓	✓
20	TFTP Server Provisioned Modem Address	IP Address	None	False	✓	✓	✓
21	Software Upgrade TFTP Server	IP Address N	None	False	✓	✓	✓
22	Upstream Packet Classification Encoding	Compound	None	True		✓	✓
22.1	Classifier Reference	Unsigned integer 8	Between 1 and 255 inclusive	False		✓	✓
22.2	Classifier Identifier	Unsigned integer 16	Between 1 and 65535 inclusive	False		✓	✓
22.3	Service Flow Reference	Unsigned integer 16	Between 1 and 65535 inclusive	False		✓	✓
22.4	Service Flow Identifier	Unsigned integer 32	Greater than 0	False		✓	✓
22.5	Rule Priority	Unsigned integer 8	None	False		✓	✓
22.6	Classifier Activation State	Boolean	None	False		✓	✓
22.7	Dynamic Service Change Action	Unsigned integer 8	Less than 3	False		✓	✓
22.8	Classifier Error Encodings	Compound	None	False		✓	✓
22.8.1	Error Parameter	Sub Type	None	False		✓	✓
22.8.2	Error Code	Unsigned integer 8	Less than 26	False		✓	✓
22.8.3	Error Message	ZTAASCII	None	False		✓	✓
22.9	IP Packet Classification Encodings	Compound	None	False		✓	✓
22.9.1	IP Type of Service Range and Mask	Unsigned integer 8 triplet	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
22.9.2	IP Protocol	Unsigned integer 16	Less than 258	False		✓	✓
22.9.3	IP Source Address	IP Address	None	False		✓	✓
22.9.4	IP Source Mask	IP Address	None	False		✓	✓
22.9.5	IP Destination Address	IP Address	None	False		✓	✓
22.9.6	IP Destination Mask	IP Address	None	False		✓	✓
22.9.7	TCP/UDP Source Port Start	Unsigned integer 16	None	False		✓	✓
22.9.8	TCP/UDP Source Port End	Unsigned integer 16	None	False		✓	✓
22.9.9	TCP/UDP Destination Port Start	Unsigned integer 16	None	False		✓	✓
22.9.10	TCP/UDP Destination Port End	Unsigned integer 16	None	False		✓	✓
22.10	Ethernet LLC Packet Classification Encodings	Compound	None	False		✓	✓
22.10.1	Destination MAC Address	MAC Address and Mask	None	False		✓	✓
22.10.2	Source MAC Address	MAC Address	None	False		✓	✓
22.10.3	Ethertype/DSAP/MacType	Unsigned integer 8 and unsigned integer 16	None	False		✓	✓
22.11	IEEE 802.1P/Q Packet Classification Encodings	Compound	None	False		✓	✓
22.11.1	IEEE 802.1P User_Priority	Unsigned integer 8 pair	Less than 8	False		✓	✓
22.11.2	IEEE 802.1Q VLAN_ID	Unsigned integer 16	None	False		✓	✓
22.43	Vendor Specific Classifier Parameters	Compound	None	False		✓	✓
22.43.8	Vendor ID	OUI	None	False		✓	✓
23	Downstream Packet Classification Encoding	Compound	None	True		✓	✓
23.1	Classifier Reference	Unsigned integer 8	Between 1 and 255 inclusive	False		✓	✓
23.2	Classifier Identifier	Unsigned integer 16		False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
23.3	Service Flow Reference	Unsigned integer 16	Between 1 and 65535	False		✓	✓
23.4	Service Flow Identifier	Unsigned integer 32	Between 1 and 65535	False		✓	✓
23.5	Rule Priority	Unsigned integer 8	Greater than 0	False		✓	✓
23.6	Classifier Activation	Boolean	None	False		✓	✓
23.7	Dynamic Service Change Action	Unsigned integer 8	Less than 3	False		✓	✓
23.8	Classifier Error Encodings	Compound	None	False		✓	✓
23.8.1	Error Parameter	Sub Type	None	False		✓	✓
23.8.2	Error Code	Unsigned integer 8	Less than 26			✓	✓
23.8.3	Error Message	ZTASCII	None	False		✓	✓
23.9	IP Classification Encodings	Compound	None	False		✓	✓
23.9.1	IP Type of Service Range and Mask	Unsigned integer 8	None	False		✓	✓
23.9.2	IP Protocol	Unsigned integer 16	Less than 258	False		✓	✓
23.9.3	IP Source Address	IP Address	None	False		✓	✓
23.9.4	IP Source Mask	IP Address	None	False		✓	✓
23.9.5	IP Destination Address	IP Address	None	False		✓	✓
23.9.6	IP Destination Mask	IP Address	None	False		✓	✓
23.9.7	TCP/UDP Source Port Start	Unsigned integer 16	None	False		✓	✓
23.9.8	TCP/UDP Source Port End	Unsigned integer 16	None	False		✓	✓
23.9.9	TCP/UDP Destination Port Start	Unsigned integer 16	None	False		✓	✓
23.9.10	TCP/UDP Destination Port End	Unsigned integer 16	None	False		✓	✓
23.10	Ethernet LLC Packet Classification Encodings	Compound					✓
23.10.1	Destination MAC Address	MAC Address and Mask\	None	False		✓	✓
23.10.2	Source MAC Address	MAC Address	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
23.10.3	Ethertype/DSAP/MacType	Unsigned integer 8 and unsigned integer 16	None	False		✓	✓
23.11	IEEE 802.1P/Q Packet Classification Encodings	Compound	None	False		✓	✓
23.11.1	IEEE 802.1P User_Priority	Unsigned integer 8 pair	Less than 8	False		✓	✓
23.11.2	IEEE 802.1Q VLAN_ID	Unsigned integer 16	None	False		✓	✓
23.43	Vendor Specific Classifier Parameters	Compound	None	False		✓	✓
23.43.8	Vendor ID	OUI	None	False		✓	✓
24	Upstream Service Flow Scheduling	Compound	None	True		✓	✓
24.1	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓
24.3	Service Identifier	Unsigned integer 16	None	False		✓	✓
24.4	Service Class Name	ZTASCII	None	False		✓	✓
24.5	Service Flow Error Encodings	Compound	None	True		✓	✓
24.5.1	Errored Parameter	Unsigned integer 8	None	False		✓	✓
24.5.2	Error Code	Unsigned integer 8	Less than 26	False		✓	✓
24.5.3	Error Message	ZTASCII	None	False		✓	✓
24.6	Quality of Service Parameter Set Type	Unsigned integer 8	Less than 8	False		✓	✓
24.7	Traffic Priority	Unsigned integer 8	Less than 8	False		✓	✓
24.8	Upstream Maximum Sustained Traffic Rate	Unsigned integer 32	None	False		✓	✓
24.9	Maximum Traffic Burst	Unsigned integer 32	None	False		✓	✓
24.10	Minimum Reserved Traffic Rate	Unsigned integer 32	None	False		✓	✓
24.11	Assumed Minimum Reserved Rate Packet Size	Unsigned integer 16	None	False		✓	✓
24.12	Timeout for active QoS Parameters	Unsigned integer 16	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
24.13	Timeout for Admitted QoS Parameters	Unsigned integer 16	None	False		✓	✓
24.14	Maximum Concatenated Burst	Unsigned integer 16	None	False		✓	✓
24.15	Service Flow Scheduling Type	Unsigned integer 8	Between 1-6 inclusive	False		✓	✓
24.16	Request/Transmission Policy	Unsigned integer 32	Less than 512	False		✓	✓
24.17	Nominal Polling Interval	Unsigned integer 32	None	False		✓	✓
24.18	Tolerated Poll Jitter	Unsigned integer 32	None	False		✓	✓
24.19	Unsolicited Grant Size	Unsigned integer 16	None	False		✓	✓
24.20	Nominal Grant Interval	Unsigned integer 32	None	False		✓	✓
24.21	Tolerated Grant Jitter	Unsigned integer 32	None	False		✓	✓
24.22	Grants per Interval	Unsigned integer 8	Less than 128	False		✓	✓
24.23	IP Type of Service Overwrite	Unsigned integer 8 pair	None	False		✓	✓
24.24	Unsolicited Grant Time Reference	Unsigned integer 32	None	False		✓	✓
24.43	Vendor Specific PHS Parameters	Compound	None	False		✓	✓
24.43.8	Vendor ID	OUI	None	False		✓	✓
25	Downstream Service Flow Scheduling	Compound	None	True		✓	✓
25.1	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓
25.3	Service Identifier	Unsigned integer 16	None	False		✓	✓
25.4	Service Class Name	ZTASCII	None	False		✓	✓
25.5	Service Flow Error Encodings	Compound	None	True		✓	✓
25.5.1	Errored Parameter	Unsigned integer 8	None	False		✓	✓
25.5.2	Error Code	Unsigned integer 8	Less than 26	False		✓	✓
25.5.3	Error Message	ZTASCII	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
25.6	Quality of Service Parameter Set Type	Unsigned integer 8	Less than 8	False		✓	✓
25.7	Traffic Priority	Unsigned integer 8	Less than 8	False		✓	✓
25.8	Downstream Maximum Sustained Traffic Rate	Unsigned integer 32	None	False		✓	✓
25.9	Maximum Traffic Burst	Unsigned integer 32	None	False		✓	✓
25.10	Minimum Reserved Traffic Rate	Unsigned integer 32	None	False		✓	✓
25.11	Assumed Minimum Reserved Rate Packet Size	Unsigned integer 16	None	False		✓	✓
25.12	Timeout for active QoS Parameters	Unsigned integer 16	None	False		✓	✓
25.13	Timeout for Admitted QoS Parameters	Unsigned integer 16	None	False		✓	✓
25.14	Maximum Downstream Latency	Unsigned integer 32	None	False		✓	✓
25.43	Vendor Specific PHS Parameters	Compound	None	False		✓	✓
25.43.8	Vendor ID	OUI	None	False		✓	✓
26	Payload Header Suppression	Compound	None	True		✓	✓
26.1	Classifier Reference	Unsigned integer 8	Greater than 0	False		✓	✓
26.2	Classifier Identifier	Unsigned integer 16	Greater than 0	False		✓	✓
26.3	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓
26.4	Service Flow Identifier	Unsigned integer 32	Greater than 0	False		✓	✓
26.5	Dynamic Service Change Action	Unsigned integer 8	Less than 4	False		✓	✓
26.6	Payload Header Suppression Error Encodings	Compound	None	False		✓	✓
26.6.1	Errored Parameter	Unsigned integer 8	None	False		✓	✓
26.6.2	Error Code	Unsigned integer 8	Less than 26	False		✓	✓
26.6.3	Error Message	ZTASCII	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
26.7	Payload Header Suppression Field (PHSF)	Bytes	None	False		✓	✓
26.8	Payload Header Suppression Index (PHSI)	Unsigned integer 8	Greater than 0	False		✓	✓
26.9	Payload Header Suppression Mask (PHSM)	Bytes	None	False		✓	✓
26.10	Payload Header Suppression Size (PHSS)	Unsigned integer 8	None	False		✓	✓
26.11	Payload Header Suppression Verification (PHSV)	Boolean	None	False		✓	✓
26.43	Vendor Specific PHS Parameters	Compound	None	False		✓	✓
26.43.8	Vendor ID	OUI	None	False		✓	✓
28	Maximum Number of Classifiers	Unsigned integer 16	None	False		✓	✓
29	Privacy Enable	Boolean	None	False		✓	✓
32	Manufacturer CVC	Bytes	None	False		✓	✓
33	Co-signer CVC	Bytes	None	False		✓	✓
34	SnmpV3 Kickstart Value	Compound	None	False		✓	✓
34.1	SnmpV3 Kickstart Security Name	NVTASCII	None	False		✓	✓
34.2	SnmpV3 Kickstart Manager Public Number	Bytes	None	False		✓	✓
35	Subscriber Management Control	Bytes	None	False		✓	✓
36	Subscriber Management CPE IP Table	Multiple IP Addresses	None	False		✓	✓
37	Subscriber Management Filter Groups	Bytes	None	False		✓	✓
38	Configuration File Element - docsisV3 Notification Receiver	Compound	None	False		✓	✓
38.1	IP Address of Trap Receiver	IP address	None	False		✓	✓
38.2	UDP Port Number of Trap Receiver	unsigned integer 16	None	False		✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
38.3	Type of Trap Sent by the PS	unsigned integer 8	None	False		✓	✓
38.4	Timeout	unsigned integer 32	None	False		✓	✓
38.5	Number of Retries When Sending an Inform After Sending the Inform First	unsigned integer 8	None	False		✓	✓
38.6	Notification Filtering Parameters	OID	None	False		✓	✓
38.7	Security Name to Use When Sending SNMP V3 Notification	NVTASCII	None	False		✓	✓
39	Enable 2.0 Mode	Enable/Disable	None	False			✓
40	Enable Test Mode	SubOptions	None	True			✓
41	Downstream Channel List	SubOptions	None	True			✓
41.1	Single Downstream Channel	SubOptions	None	True			✓
41.1.1	Single Downstream Channel Timeout	unsigned integer 16	None	False			✓
41.1.2	Single Downstream Channel Frequency	unsigned integer 32	None	False			✓
41.2	Downstream Frequency Range	SubOptions		True			✓
41.2.1	Downstream Frequency Range Timeout	unsigned integer 16	None	False			✓
41.2.2	Downstream Frequency Range Start	unsigned integer 32	Multiples of 62500	False			✓
41.2.3	Downstream Frequency Range End	unsigned integer 32	Multiples of 62500	False			✓
41.2.4	Downstream Frequency Range Step Size	unsigned integer 32	None	False			✓
41.3	Default Scanning	unsigned integer 32	None	True			✓
42	Multicast MAC Address	MAC Address	None	True			✓
43	Vendor-Specific Information	Compound	None	True	✓	✓	✓
43.1	Static Downstream Frequency	Unsigned integer 32	None	False	✓	✓	✓
43.2	Sync Loss Timeout	Unsigned integer 32	None	False	✓	✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
43.3	Update Boot Monitor Image	NVTASCII	None	False	✓	✓	✓
43.4	Power Backoff	Unsigned integer 16	None	False	✓	✓	✓
43.8	Vendor ID	OUI	None	False	✓	✓	✓
43.9	Update Factory System Image	Boolean	None	False	✓	✓	✓
43.10	Phone Lines	Unsigned integer 8	None	False	✓	✓	✓
43.11	IP Precedence Settings	Compound	None	True	✓	✓	✓
43.11.1	IP Precedence Value	Unsigned integer 8	None	False	✓	✓	✓
43.11.2	Rate Limit	Unsigned integer 32	None	False	✓	✓	✓
43.128	IOS Configuration Filename	NVTASCII	None	False	✓	✓	✓
43.129	IOS Config File Without Console Disable	NVTASCII	None	False	✓	✓	✓
43.131	IOS CLI Command	NVTASCII	None	True	✓	✓	✓
43.132	1.0 Plus Flow Encodings	Compound	None	False	✓	✓	✓
43.132.1	1.0 Plus Flow ID	Unsigned integer 8	None	False	✓	✓	✓
43.132.2	Class ID	Unsigned integer 8	None	False	✓	✓	✓
43.132.3	Unsolicited Grant Size	Unsigned integer 16	Between 1-65535 inclusive	False	✓	✓	✓
43.132.4	Nominal Grant Interval	Unsigned integer 32	Between 1-65535 inclusive	False	✓	✓	✓
43.132.5	Grants Per Interval	Unsigned integer 8	Between 0-127 inclusive	False	✓	✓	✓
43.132.6	Embedded Voice Calls	Unsigned integer 8	Between 0-127 inclusive	False	✓	✓	✓
43.132.7	Hold Queue Length	Unsigned integer 16	Between 0-4096 inclusive	False	✓	✓	✓
43.132.8	Fair Queue	Compound	None	False	✓	✓	✓
43.132.8.1	Congestive Discard Threshold	Unsigned integer 16	Between 1-4096 inclusive	False	✓	✓	✓

Table 12-3 DOCSIS Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	DOCSIS Version		
					1.0	1.1	2.0
43.132.8.2	Number of Dynamic Conversation Queues	Unsigned integer 16	Between 16-4096 inclusive	False	✓	✓	✓
43.132.8.3	Number of Reservable Conversation Queues	Unsigned integer 16	Between 0-1000 inclusive	False	✓	✓	✓
43.132.9	Custom Queue List Length	Unsigned integer 8	Between 1-16 inclusive	False	✓	✓	✓
43.132.10	Random Detection	Boolean	None	False	✓	✓	✓
43.132.11	Priority Group	Unsigned integer 8	Between 1-16 inclusive	False	✓	✓	✓
43.132.12	Service Policy File	NVTASCII	None	False	✓	✓	✓
43.132.13	Inactivity Timer	Unsigned integer 16	Between 1-10080 inclusive	False	✓	✓	✓
43.132.14	COS Tag	NVTASCII	None	False	✓	✓	✓
43.133	Downstream Sub Channel ID	Unsigned integer 8	Between 0-15 inclusive	False	✓	✓	✓
43.134	SU Tag	NVTASCII	None	False	✓	✓	✓
255	End-of-Data Marker	No length and no value	N/A	False	✓	✓	✓

PacketCable Option Support

Table 12-4 identifies the PacketCable 1.0 MTA options supported by BACC.

Table 12-4 PacketCable MTA 1.0 Options

Number	Description	Encoding	Validation	Multi-valued	PacketCable Version	
					1.0	1.1
11	SNMP MIB Object	SNMPVarBind with 1 byte length	None	True	✓	✓
38	SNMPv3 Notification Receiver	SubOptions	None	True	✓	✓
38.1	SNMPv3 Notification Receiver IP Address	IPAddress	None	False	✓	✓
38.2	SNMPv3 Notification Receiver UDP Port Number	Unsigned integer 16	None	False	✓	✓

Table 12-4 PacketCable MTA 1.0 Options (continued)

Number	Description	Encoding	Validation	Multi-valued	PacketCable Version	
					1.0	1.1
38.3	SNMPv3 Notification Receiver Trap Type	SNMPTrapType	From 1 to 5	False	✓	✓
38.4	SNMPv3 Notification Receiver Timeout	Unsigned integer 16	None	False	✓	✓
38.5	SNMPv3 Notification Receiver Retries	Unsigned integer 16	From 0 to 255	False	✓	✓
38.6	Notification Receiver Filtering Parameters	OID	None	False	✓	✓
38.7	Notification Receiver Security Name	NVTASCII	None	False	✓	✓
43	Vendor-Specific Information	SubOptions	None	True	✓	✓
43.8	Vendor ID	OUI	None	False	✓	✓
64	SNMP MIB Object	SNMPVarBind with 2 byte length	None	True	✓	✓
254	Telephony Config File Start/End	Unsigned integer 8	Must be 1 or 255	False	✓	✓

Non-Secure CableHome Option Support

Table 12-5 identifies the non-secure CableHome options supported by BACC.

Table 12-5 Non-secure CableHome Options and Version Support

Option Number	Description	Encoding	Validation	Multi-valued	CableHome Version
					1.0
0	PAD	No length and no value	None	True	✓
9	Software Upgrade Filename	NVTASCII	None	False	✓
10	SNMP Write-Access Control	OIDCF	None	True	✓
12	Modem IP Address	IP Address	None	False	✓
14	CPE Ethernet MAC Address	MACAddress	None	True	✓
21	Software Upgrade TFTP Server	IPAddress	None	False	✓
28	SNMP MIB Object	SNMPVarBind	None	True	✓
32	Manufacturer CVC	Bytes	None	False	✓
33	Co-signer CVC	Bytes	None	True	✓
34	SnmpV3 Kickstart Value	SubOptions	None	False	✓

Table 12-5 Non-secure CableHome Options and Version Support (continued)

Option Number	Description	Encoding	Validation	Multi-valued	CableHome Version 1.0
34.1	SnmpV3 Kickstart Security Name	NVTASCII	None	False	✓
38	SNMPv3 Notification Receiver	SubOptions	None	True	✓
38.1	SNMPv3 Notification Receiver IP Address	IPAddress	None	False	✓
38.2	SNMPv3 Notification Receiver UDP Port Number	Unsigned integer 16	None	False	✓
38.3	SNMPv3 Notification Receiver Trap Type	SNMPTrapType	From 1 to5	False	✓
38.4	SNMPv3 Notification Receiver Timeout	Unsigned integer 16	None	False	✓
38.5	SNMPv3 Notification Receiver Retries	Unsigned integer 16	None	False	✓
38.6	Notification Receiver Filtering Parameters	OID	None	False	✓
38.7	Notification Receiver Security Name	NVTASCII	None	False	✓
43	Vendor-Specific Information	SubOptions	None	True	✓
43.1	Vendor ID	OUI	None	False	✓
53	PS MIC. A 20 octet SHA-1 hash of PS config file	Bytes	None	False	✓
255	End-of-Data Marker	No length and no value	None	False	✓

Using the Configuration File Utility

You use the configuration file utility to test, validate, and view PacketCable 1.0, DOCSIS 1.0/1.1/2.0, and CableHome template and configuration files. These activities are critical to successful deployment of individualized configuration files. See the "Developing Template Files" section on page 12-1 for more information on templates.

The configuration file utility is available only when the RDU is installed and the utility is installed in the <BACC_HOME>/rdu/bin directory.

Both the template file being encoded and the binary file being decoded must reside in the directory from which the configuration file utility is invoked.

All examples found in this section assume that the RDU is operating and that these conditions apply:

- The BACC application is installed in the home directory (/opt/CSCObpr).
- The RDU login name is admin.
- The RDU login password is changeme.

Note Some of the examples provided in this section have been truncated whenever the omitted formation is of no consequence to the example of its outcome. Instances where this occurs are identified by a series of periods (...) that are located just prior to the example summary.

This section discusses these topics:

Running the Configuration File Utility

In subsequent procedures and examples, the phrase "run the configuration file utility" means to enter the **runCfgUtil.sh** command from the directory specified. To run the configuration file utility, run this command from the home directory:

```
runCfgUtil.sh (options)
```

Where the available *(options)* include:

- -c <secret>—Specifies the CMTS shared secret when parsing a DOCSIS template file. To specify the default shared secret, enter **-c cisco**.
- -cablehome—Identifies the input file as a CableHome portal service configuration file. Do not use this with either the -docsis or -pkt options.
- -d—Decodes the binary input file. Do not use this with the -e option.
- -docsis—Specifies the input file is a DOCSIS configuration file. Do not use this default with the -pkt option.
- -e—Encodes the template input file. Do not use this default with the -d option.
- -g—Generates a template file from either a docsis, packetcable or cablehome binary file.
- -h<host:port>—Specifies the host and port number. The default port number id 49187.
- -i <device id>—Specifies the device to use when parsing macro variables. For example, if your device is ID is 1,6,00:00:00:00:00:01, enter **-i 1,6,00:00:00:00:00:01**. When using this option, you must also use the -u and -p options, respectively, to specify the username and password. Do not use this with the -m option.
- -l <filename>—Identifies the input file as being on the local file system. For example, if your input file is called any_file, enter -l **any_file**. Do not use this with the -r option.
- -loc—Specifies the PacketCable locale, na (North America) or euro (Europe). The default is na. If the MTA is euro-MTA, then the locale should be set to euro.
- -m <*macros*>—Specifies key value pairs for macro variables. The format is key=value. If you require multiple macro variables, use a double comma separator between the key value pairs, for example, key_1=value_1,,key_2=value_2. Do not use this with the -i option.

- -p <password>—Specifies the password to use when connecting to the RDU. For example, if your password is 123456, enter **-p 123456**.
- -o <filename>—Saves parsed template file as a binary file. For example, if you want the output to be found in a file call op_file, enter **-o op_file**.
- -pkt—Identifies the input file as a PacketCable MTA configuration file. Do not use this with the -docsis option.
- -r <filename>—Identifies the input file as an external file that has been added to the RDU. For example, if your file is called file25 enter **-r file25**. When using this option you must also use the -u and -p options, respectively, to specify the username and password. Do not use this with the -l option.
- -s—Displays the parsed template or the contents of the binary file in a human readable format.
- -t—Specifies the PacketCable encoding type, secure or basic (default is secure).
- -u <username>—Specifies the username to use when connecting to the RDU. For example, if your username is admin, enter **-u admin**.
- -v <version>—Specifies the DOCSIS version being used. For example, if you are using DOCSIS 1.1, enter **-v 1.1**. If you do not specify the version number, the command defaults to use DOCSIS 2.0.

Note The configuration file utility does not include option 19 (TFTP server timestamp) and option 20 (TFTP server provisioned modem address) in the template file, however the BACC TFTP mixing does. Also, options 6 (CM MIC) and 7 (CMTS MIC) are both automatically inserted into the encoded template file. Therefore, you do not have to specify these message integrity checks (MIC).

Using the Configuration File Utility

To use the configuration file utility to test BACC templates:

Step 1 Develop the template as described in the "Developing Template Files" section on page 12-1.

Step 2 Run the configuration file utility on the local file system. If the template contains macro variables, perform these operations in the order specified:

a. Test with command line substitution.

b. Test with a device that has been added to your RDU.

Step 3 Add the template (and any included templates that are used) to the RDU.

Step 4 Run the configuration file utility as an external file. If the template contains macro variables, perform these operations in the order specified:

a. Test with command line substitution.

b. Test with a device that has been added to your RDU.

Step 5 Configure a class of service to use the template.

Converting a Binary File Into a Template File

Usage Guidelines

Use the **runCfgUtil.sh** command to convert binary configuration memory files into template files. BACC dynamic configuration generation is based on templates that are created. Automatically converting existing, tested, binary files to template files speeds the process and reduces the possibility of introducing errors.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to convert an existing binary file into a template file.

```
runCfgUtil.sh -g -l <binary_file> -o <template_file>
```

Where:

- -g—specifies that a template file needs to be generated from an input binary file
- -l <binary_file>—specifies the local input file, including the path name. In all cases, the input binary file name will have a *.cm file extension; bronze.cm for example.
- -o <template_file>—specifies the output template file, including the path name. In all cases, the input binary file name will have a *.tmpl file extension; test.tmpl for example.

Parsing a Local Template File

Usage Guidelines

Use the **runCfgUtil.sh** command to parse external template files.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to parse a local template file:

```
runCfgUtil.sh -pkt -l <file>
```

Where:

- -pkt—identifies the input file as a packetcable MTA file
- -l—specifies the input file is on the local file system
- <file>—identifies the input template file being parsed

Parsing the File

To parse a template file that is on the local file system:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/packet_cable.

Step 2 Select a template file to use.

Note This example uses an existing template file called unprov_packet_cable.tmpl. The -pkt option is used because this is a PacketCable MTA template.

Step 3 Run the configuration file utility using this command:

```
runCfgUtil.sh -pkt -l unprov_packet_cable.tmpl
```

Where:

- -pkt—identifies the input file as a packetcable MTA file
- -l—specifies the input file is on the local file system
- unprov_packet_cable.tmpl—identifies the input template file being parsed

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                     Value
0       FE0101          254       Telephony Config File Start/End  1
3       0B153013060E    11        SNMP MIB Object                  .iso.org.dod.internet.
        2B06010401A30B                                             private.enterprises.ca
        0202010101                                                 bleLabs.clabProject.cl
        0700020102                                                 abProjPacketCable.pktc
                                                                   MtaMib.pktcMtaMibObjec
                                                                   ts .pktcMtaDevBase.
                                                                   pktcMtaDevEnable
                                                                   d.0,INTEGER,false(2)
.....................................

0 error(s), 0 warning(s) detected. Parsing of unprov_packet_cable.tmpl was successful.
The file unprov_packet_cable.tmpl was parsed successfully in 434 ms.
The parser initialization time was 92 ms.
The parser parse time was 342 ms.
```

Parsing an External Template File

Usage Guidelines

Use the **runCfgUtil.sh** command to parse external template files.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to parse an external template file:

```
runCfgUtil.sh -r <file> -u <username> -p <password>
```

Where:

- -r <file>—identifies the input file as an external file that has been added to the RDU
- -u <username>—specifies the username to use when connecting to the RDU
- -p <password>— specifies the password to use when connecting to the RDU

Parsing the File

To parse a template file that has been added to the RDU:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/docsis.

Step 2 Select a template file to use.

Note This example uses an existing template file called unprov.tmpl. The -docsis option is used because a DOCSIS template is being used.

Step 3 Run the configuration file utility using this command:

```
runCfgUtil.sh -r <unprov.tmpl> -u <admin> -p <changeme> -docsis
```

Where:

- unprov.tmpl—identifies the input file
- admin—identifies the user name
- changeme—identifies the password
- -docsis—identifies the file as a DOCSIS template

After running the utility, results similar to these should appear:

Note The results shown here are for illustration only and have been truncated for brevity.

```
Off     File Bytes      Option    Description                     Value
0       030101          3         Network Access Control          On
3       041F            4         Class of Service
5       010101          4.1       Class ID                        1
8       02040000FA00    4.2       Maximum Downstream Rate         128000 bits/sec
14      03040000FA00    4.3       Maximum Upstream Rate           64000 bits/sec
20      040101          4.4       Upstream Channel Priority       1
. . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . .
252     06108506547F    6         CM MIC Configuration Setting    8506547FC9152B44
        C9152B44DB95                                              DB955420843EF6FE
        5420843EF6FE
270     0710644B675B    7         CMTS MIC Configuration Setting  644B675B70B7BD3E
        70B7BD3E09AC                                              09AC210F794A1E8F
        210F794A1E8F
288     FF              255       End-of-Data Marker
289     00              0         PAD
290     00              0         PAD
291     00              0         PAD
0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Parsing a Template File and Adding a User-Specified Shared Secret

Usage Guidelines

Use the **runCfgUtil.sh** command to parse a template file and add a shared secret that you specify.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to parse a template file and add a shared secret:

```
runCfgUtil.sh -e -docsis -l <file> -c <secret>
```

Where:

- -e—identifies the encode option.
- -docsis—Identifies the input file as a DOCSIS template file.
- -l—Specifies the input file is on the local file system.
- <file>—Identifies the input template file being parsed.
- -c <secret>—Specifies the CMTS shared secret when parsing a DOCSIS template file. The default shared secret is -**c cisco**.

Parsing the File

To parse a locally saved template file, and set a user specified shared secret:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/docsis.

Step 2 Select a template file to parse.

Note This example uses an existing template file called unprov.tmpl. The -docsis option is used because this is a DOCSIS template.

Step 3 Run the configuration file utility using this command:

```
runCfgUtil.sh -e -docsis -l unprov.tmpl -c shared
```

Where:

- -e—identifies the encode option.
- -docsis—Identifies the input file as a DOCSIS template file.
- -l—Specifies the input file is on the local file system. The locally saved file used in this example is unprov.tmp.
- -c—Indicates that a shared secret is being set and shared identifies that new shared secret.

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                    Value

0       030100          3         Network Access Control          Off

3       041F            4         Class of Service

5       010101          4.1       Class ID                        1
```

```
8      02040001F400   4.2       Maximum Downstream Rate          128000 bits/sec

14     03040000FA00   4.3       Maximum Upstream Rate            64000 bits/sec

20     040101         4.4       Upstream Channel Priority        1

. . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . .

252    06108506547F   6         CM MIC Configuration Setting      8506547FC9152B44
       C9152B44DB95                                             DB955420843EF6FE
       5420843EF6FE

270    0710644B675B   7         CMTS MIC Configuration Setting   644B675B70B7BD3E
       70B7BD3E09AC                                             09AC210F794A1E8F
       210F794A1E8F

288    FF             255       End-of-Data Marker

289    00             0         PAD

290    00             0         PAD

291    00             0         PAD

0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Specifying Macro Variables at the Command Line

Usage Guidelines

Use the **runCfgUtil.sh** command to specify macro variables.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command when specifying macro variables at the command line:

```
runCfgUtil.sh -e -l <file> -m <"macros">
```

Where:

- -e—identifies the encode option.
- -l—Specifies the input file is on the local file system.
- <file>—Identifies the input template file being parsed.
- -m—Specifies the macro variables to be substituted when parsing a template.
- <"macros">—Identifies the desired macros. When multiple macro variables are required, insert a double comma separator between each macro.

To specify values for macro variables at the command line:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/templates.

Step 2 Select a template file to use.

Step 3 Identify the macro variables in the template. In this example, the macro variables are macro1 (option 3) and macro11 (option 4.2).

Step 4 Identify the values for the macro variables. The value for macro1 will be set to 1, and the value for macro11 to 64000.

Step 5 Run the configuration file utility using this command:

```
runCfgUtil.sh -e -l macro.tmpl -m "macro1=1,,macro11=64000"
```

Where:

- -e—identifies the encode option.
- -unprov.tmpl—Identifies the input file.
- macro1=1,,macro11=64000—Identifies the key value pairs for macro variables. Since multiple macro variables are necessary, a double comma separator is inserted between the key value pairs.

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                     Value
 0       030101          3         Network Access Control           On
 3       041F            4         Class of Service
 5       010101          4.1       Class ID                         1
 8       02040000FA00   4.2       Maximum Downstream Rate          64000 bits/sec
 14      03040000FA00   4.3       Maximum Upstream Rate            64000 bits/sec
 20      040101          4.4      Upstream Channel Priority        1
.....................................

0 error(s), 0 warning(s) detected. Parsing of macro.tmpl was successful.
The file macro.tmpl was parsed successfully in 854 ms.
The parser initialization time was 76 ms.
The parser parse time was 778 ms.
```

Specifying a Device for Macro Variables

Usage Guidelines

Use the **runCfgUtil.sh** command to specify a device for macro variables.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command when specifying a device for macro variables:

```
runCfgUtil.sh -e -l <file> -i <MAC> -u <username> -p <password>
```

Where:

- -e—identifies the encode option
- -l—specifies the input file is on the local file system
- <file>—identifies the input template file being parsed
- -i—specifies the device to use when parsing macro variables
- <MAC>—identifies the MAC address of the device
- -u <username>—specifies the username to use when connecting to the RDU
- -p <password>— specifies the password to use when connecting to the RDU

To specify a device to be used for macro variable substitution:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/templates.

Step 2 Select a template file to use. This example will use the existing template file, macro.tmpl.

Step 3 Identify the macro variables in the template. In this example, the macro variables are macro1 (option 3) and macro11 (option 4.2).

Step 4 Identify the device to use. This example will assume that the device exists in the RDU and has the macro variables set as properties. The value for macro1 will be set to 1, and the value for macro11 to 64000.

Step 5 Run the configuration file utility using this command:

```
runCfgUtil.sh -l macro.tmpl -i "1,6,00:01:02:03:04:05" -u admin -p changeme
```

Where:

- macro.tmpl—Identifies the input file.
- 1,6,00:01:02:03:04:05—Identifies the MAC address of the device. The MAC address used here is for example purposes only.
- admin—Identifies the default username being used in this example.
- changeme—identifies the default password being used in this example

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                     Value
0        030101          3        Network Access Control            On
3        041F            4        Class of Service
5        010101          4.1      Class ID                          1
8        02040000FA00    4.2      Maximum Downstream Rate           64000 bits/sec
14       03040000FA00    4.3      Maximum Upstream Rate             64000 bits/sec
20       040101          4.4      Upstream Channel Priority         1
....................................

0 error(s), 0 warning(s) detected. Parsing of macro.tmpl was successful.
The file macro.tmpl was parsed successfully in 823 ms.
The parser initialization time was 102 ms.
The parser parse time was 803 ms.
```

Specifying Output to a Binary File

Usage Guidelines

Use the **runCfgUtil.sh** command to specify a device for macro variables.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command when specifying output to a binary file:

```
runCfgUtil.sh -l <input_file> -o <output_file>
```

Where:

- -l—specifies the input file is on the local file system
- <input_file>—identifies the input template file being parsed
- -o—specifies that file name where the binary contents of the parsed template file are stored
- <output_file>—specifies the output filename to be used

To specify the output from parsing a template to a binary file:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/templates.

Step 2 Select a template file to use.

Step 3 Identify the name of the output file. This example will use unprov.cm.

Step 4 Run the configuration file utility using this command:

```
runCfgUtil.sh -l unprov.tmpl -o unprov.cm
```

Where:

- unprov.tmpl—identifies the existing template file being parsed into a binary file
- unprov.cm—specifies the output filename to be used

After running the utility, results similar to these should appear:

```
# /opt/CSCObpr/rdu/bin/runCfgUtil.sh -l unprov.tmpl -o unprov.cm
Broadband Provisioning Registrar Configuration Utility
Version: 2.7

0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 595 ms.
The parser initialization time was 262 ms.
The parser parse time was 333 ms.
```

Viewing a Local Binary File

Usage Guidelines

Use the **runCfgUtil.sh** command to view a local binary file.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to view a local binary file:

```
runCfgUtil.sh -d -l <file>
```

Where:

- -d—specifies that the command is going to decode a binary input file for viewing
- -l—identifies that the input file is located on the local file system
- <file>—identifies the existing binary input file to be viewed

To view a binary file that is on the local file system:

Step 1 Change directory to /opt/CSCObpr/rdu/samples/packet_cable.

Step 2 Select a binary file to view.

Step 3 Run the configuration file utility using this command:

```
runCfgUtil.sh  -d -l unprov_packet_cable.bin
```

Where:

- unprov_packet_cable.bin—identifies the existing binary input file to be viewed

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                     Value
  0      FE0101          254      Telephony Config File Start/End  1
        0B153013060E    11         SNMP MIB Object                 .iso.org.dod.internet.
3       2B06010401A30B                                             private.enterprises.ca
        02020101010700                                             bleLabs.clabProject.
        020102                                                     clabProjPacketCa
                                                                   ble.pktcMtaMib.pktcMta
                                                                   MibObjects
                                                                   .pktcMtaDevBase.
                                                                   pktcMtaDevEnable
                                                                   d.0,INTEGER,fals e(2)
. . . . . . . . . . . . . . . . .
```

Viewing an External Binary File

Usage Guidelines

Use the **runCfgUtil.sh** command to view an external binary file.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to view external binary files:

```
runCfgUtil.sh -d -r <file> -u <username> -p <password>
```

Where:

- -d—specifies that the command is going to decode a binary input file for viewing
- -r—identifies the input file an external file that has been added to the RDU
- <file>—identifies the existing external binary file in the RDU
- -u <username>—specifies the username to use when connecting to the RDU
- -p <password>— specifies the password to use when connecting to the RDU

To view a binary file that has been added to the RDU:

Note This example assumes that the RDU is localhost:49187.

Step 1 Select a binary file to view. This example will use the existing binary file unprov.cm.

Step 2 Run the configuration file utility using this command:

```
runCfgUtil.sh -d -r unprov.cm -u admin -p changeme
```

Where:

- unprov.cm—identifies the existing external binary file in the RDU
- admin—identifies the username being used in this example
- changeme—identifies the password being used in this example

After running the utility, results similar to these should appear:

```
Off     File Bytes      Option     Description                     Value
0       030100          3         Network Access Control           Off
3       041F            4         Class of Service
5       010101          4.1       Class ID                         1
8       02040001F400    4.2       Maximum Downstream Rate          128000 bits/sec
14      03040000FA00    4.3       Maximum Upstream Rate            64000 bits/sec
20      040101          4.4       Upstream Channel Priority        1
. . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . .
252     06108506547F    6         CM MIC Configuration Setting     8506547FC9152B44
        C9152B44DB95                                               DB955420843EF6FE
        5420843EF6FE
```

```
270     0710644B675B    7         CMTS MIC Configuration Setting    644B675B70B7BD3E
        70B7BD3E09AC                                                09AC210F794A1E8F
        210F794A1E8F
288     FF              255       End-of-Data Marker
289     00              0         PAD
290     00              0         PAD
291     00              0         PAD
0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Activating PacketCable BASIC Flow

Usage Guidelines

Use the **runCfgUtil.sh** command to support the generation and insertion of the PacketCable BASIC Flow integrity hash into a BASIC flow static configuration file.

Syntax Description

You must use this syntax when using the **runCfgUtil.sh** command to support the PacketCable BASIC Flow.

```
runCfgUtil.sh -t <basic/secure>
```

Where:

- -basic—This option calculates and inserts a PacketCable BASIC flow integrity hash into an MTA static configuration file.
- -secure—This option does not insert the PacketCable BASIC flow integrity hash into an MTA static configuration file. This is the default setting.

The RDU Log Level Tool

You use the RDU log level tool to change the current log level of the RDU (on a local computer) from the command line. Although this tool is only available in an RDU installation, it is located in the <BACC_HOME>/rdu/bin directory. Table 12-6 identifies the available log levels and the types of message written to the log file when enabled.

Table 12-6 Logging Levels

Log Level	Description
Emergency	System unstable.
Alert	Immediate action needed.
Critical	Critical conditions exist.
Error	Error conditions exist.

Table 12-6 Logging Levels (continued)

Log Level	Description
Warning	Warning conditions exist.
Notification	This is a normal, but significant, condition.
Information	This is used only for informational messages.

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

Using the RDU Log Level Tool

All examples assume that the user name for the RDU is admin, the password for the RDU is changeme, and the RDU is running.

Enter this command to run the RDU log level tool:

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

Where:

- *-[0..6]*—Identifies the logging level to be used. These are the available levels:
  - 0—This is the emergency level. It sets the logging function to save all emergency messages.
  - 1—This is the alert level. It sets the logging function to save all activities requiring immediate action and those of a more severe nature.
  - 2—This is the critical level. It sets the logging function to save all unusual conditions and those of a more severe nature.
  - 3—This is the error level. It sets the logging function to save all error messages and those of a more severe nature.
  - 4—This is the warning level. It sets the logging function to save all emergency messages and those of a more severe nature.
  - 5—This is the notification level. It sets the logging function to save all notification level messages and those of a more severe nature.
  - 6—This is the info level. It sets the logging function to save all available logging messages.
- *-help*—Displays help for the tool.
- *-show*—Displays the current log 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 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 12-40
- Viewing the RDU's Current Log Level, page 12-40

Setting the RDU Log Level

You can use this tool to change the logging level from one value to any other 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.

To set the RDU logging level:

Step 1 Change directory to <BACC_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 at the prompt. 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 at the prompt. 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 RDU's Current Log Level

You can use this tool to view the RDU log and determine which logging level is configured before attempting to change the value. This procedure illustrates how to use the tool to view the RDU's current logging level, and assumes that you have a computer already connected to the RDU.

To view the RDU's current logging level:

Step 1 Change directory to <BACC_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 to Manage KDC Certificates

The PKCert tool is used to install, and then manage, the KDC certificates that are required by the KDC for its operation. This tool takes the CableLabs service provider certificates that you obtain and converts them into the series of certificate files, similar to those shown below, that the KDC needs to operate.

- 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 for use only when the KDC component is installed.

Running the PKCert Tool

Run the PKCert tool by executing this command, the default location of which is in the <BACC__HOME>/kdc directory:

PKCert.sh *[function] [option]*

Where:

- *[function]*—identifies which function will be performed. For example, enter one of these switches to choose the appropriate function:
  - -c—Creates a KDC certificate. See the "Creating a KDC Certificate" section on page 12-42 for additional information.
  - -v—Verifies and normalizes the PacketCable certificate set. See the "Validating the KDC Certificates" section on page 12-43 for additional information.

Note If you encounter difficulty using any of these options, you can specify a -? option to display all available help information on your computer screen.

- *[option]*—Implements optional functions that are dependent on the function selected above.

When you run the PKCert command, it will print a list of all errors encountered while performing the requested activities. You can use this printout to troubleshoot any problems that may have occurred.

Creating a KDC Certificate

Enter this command, from the /opt/CSCObpr/kdc directory, to create the KDC certificate:

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

Where:

- -a <name>—specifies the DNS name of KDC
- -c <Cert File>—uses the service provider certificate (DER encoded)
- -d <directory>—specifies the destination directory
- -e—identifies the certificate as being a Euro-PacketCable certificate
- -k <Key File>—uses the service provider private key (DER encoded)
- -n <Serial#>—set the certificate serial number
- -o—overwrite existing files
- -r <Realm>—specifies the Kerberos realm for KDC certificate
- -s <directory>—specifies the source directory

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
- The DNS name associated with this KDC that the MTA will use.

  For example:

```
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"
```

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 KDC server's FQDN is set to kdc.pctest.cisco.com.

Note A console message is displayed after the successful completion of the command indicating that the file /opt/CSCObpr/kdc/solaris/packetcable/certificates/KDC_private_key.pkcs8 must be copied to /opt/CSCObpr/kdc/solaris/KDC_private_key.pkcs8. The command line option **-o** tells the utility that it should overwrite any pre-existing files.

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 whenever 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 Usage instructions for PKCert are displayed on the screen when you enter the **PKCert.sh -v -?** command.

Enter this command, from the /opt/CSCObpr/kdc directory, to validate the KDC certificate:

PKCert.sh -v -s <dir> -d <dir> -o -r <dir>

Where:

- -s <directory>—specifies the source directory
- -d <directory>—specifies the destination directory
- -o—overwrites any existing files
- -r <directory>—specifies the reference certificate directory

Verification is performed against reference certificates built into this package. If the '-d' option is specified then certificate(s) are installed in the target directory with name normalization.

For example:

```
PKCert.sh  -v \
"-s . \
-d /opt/CSCObpr/kdc/solaris/packetcable/certificates \
-o"
```

Using the changeNRProperties.sh Tool

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

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

To run this command:

Step 1 Change directory to <BACC_HOME>/cnr_ep/bin.

Step 2 Run the changeNRProperties.sh command:

changeNRProperties.sh *<options>*

Where <*options*> can include:

- -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 BACC shared secret. For example, enter **-s secret**, if the shared secret is the word *secret*.
- -f <fqdn>—Identifies the RDU's 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.

Step 3 Restart the DHCP server.

Examples

This is an example of changing the Network Registrar extensions with 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
PacketCable TGT: 01
```

```
PacketCable Realm: ACME.COM
PacketCable Primary DHCP Server: 192.168.1.2
PacketCable Secondary DHCP Server: NOT SET
PacketCable Primary DNS Server: 192.168.1.2
PacketCable 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
PacketCable TGT: 01
PacketCable Realm: ACME.COM
PacketCable Primary DHCP Server: 192.168.1.2
PacketCable Secondary DHCP Server: NOT SET
PacketCable Primary DNS Server: 192.168.1.2
PacketCable Secondary DNS Server: NOT SET
```

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 communications. The KDC server requires service keys for each of the DPE's provisioning FQDNs.

The KDC server reads the service keys on startup. Any modification to the service keys requires the KDC server to be restarted. Any changes made to the DPE provisioning FQDN through the DPE CLI requires a corresponding change to the KDC service key filename. This applies since the KDC service key uses the DPE provisioning FQDN as part of its filename.

This tool, which is located in the <BACC_HOME>/kdc directory, uses command-line arguments for the DPE provisioning FQDN, realm name, and a password, and generates the service key files.

Syntax Description

Use this syntax when using the Keygen tool:

keygen *[options]* <fqdn> <realm> <password>

Where options include:

- -?—Displays this usage message and exits the command.
- -v, -version—Displays the version of this tool and exits the command.
- -q, -quiet—Implements a quiet mode whereby no output is created.
- -c, -cms—Creates a service key for the CMS system.
- <*fqdn*>—Identifies the DPE's fully qualified domain name 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 between 6 and 20 characters in length. For example:

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

```
mtafqdnmap,<fqdn>@<REALM>
mtaprovsrvr,<fqdn>@<REALM>
krbtgt,<REALM>@<REALM>
```

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

Examples

This example illustrates how to generate service keys used for KDC to DPE communications. Enter this command:

```
bash-2.05b$ keygen dpe.cisco.com CISCO.COM changeme
```

When this command is implemented, these KDC service keys are written to the <BACC_HOME>/kdc/solaris/keys directory:

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

Note The KDC must be restarted before the new keys are recognized.

Use this BPR agent command to restart the KDC:

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

This example illustrates the generation of a CMS service key. Enter this command:

```
bash-2.05b$ keygen -c cms-fqdn.com CMS-REALM-NAME changeme
```

When this command is implemented, this CMS service key is written to the <BACC_HOME>/kdc/solaris/keys directory.

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

Note When running this tool, enter the same password used for the PacketCable registration, kdc-service-key command. See the *Cisco Broadband Access Center for Cable Command Line Reference* for additional information.

Using the snmpAgentCfgUtil.sh Command

You can use the **snmpAgentCfgUtil.sh** command to manage the SNMP agent installed on a Solaris computer. Using this command, which is located in the <BACC_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 command 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 12-47
- Deleting a Host, page 12-47

Adding a Host

This command adds the host address to the list of hosts that receive SNMP notifications from the SNMP agent.

Syntax Description

To add a host, enter:

```
> snmpAgentCfgUtil.sh add host <host-addr> community  <community> [udp-port <port>]
```

Where:

- *<host-addr>*—specifies either the IP address or FQDN of the host that you want to add to the list of hosts
- *<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

This example shows how to use the **snmpAgentCfgUtil.sh** command to add a host:

```
> snmpAgentCfgUtil.sh add host test.cisco.com community trapCommunity udp-port 162
OK
Please restart [stop and start] SNMP agent.
```

Deleting a Host

You can remove a host from the list of those receiving SNMP notifications from the SNMP agent.

Syntax Description

To delete a host, enter:

```
snmpAgentCfgUtil.sh delete host <host-addr>
```

Where:

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

Examples

This example shows how to use the **snmpAgentCfgUtil.sh** command to delete a host:

```
> ./snmpAgentCfgUtil.sh delete host test.cisco.com
OK
Please restart [stop and start] SNMP agent.
```

Adding an SNMP Agent Community

You can add an SNMP community string to allow access to the SNMP agent.

Syntax Description

To add the community string, enter:

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

Where:

- <*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 bprread and bprwrite respectively. Cisco recommends that you change these values before deploying BACC.

Examples

This example shows how to add an SNMP agent community string:

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

Deleting an SNMP Agent Community

You can delete an SNMP community string to prevent access to the SNMP agent.

Syntax Description

To delete the community string, enter:

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

Where:

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

Note See the "Adding an SNMP Agent Community" section on page 12-48 for additional information on the ro and rw community strings.

Examples This example shows how to delete an SNMP agent community string:

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

Starting the SNMP Agent

This command starts the SNMP agent process on any Solaris computer having BACC already installed.

Syntax Description To start the SNMP agent process, enter:

```
snmpAgentCfgUtil.sh start
```

Stopping the SNMP Agent

This command stops the SNMP agent process on any Solaris computer having BACC already installed.

Syntax Description To stop the SNMP agent process, enter:

```
snmpAgentCfgUtil.sh stop
```

Configuring an SNMP Agent Listening Port

This command specifies the port number that the SNMP agent will listen to. The default port number used by RDU SNMP agent is 8001.

Syntax Description To establish a port for the SNMP agent to listen to, enter:

```
snmpAgentCfgUtil.sh udp-port <port>
```

Where <*port*> identifies the port number that the SNMP agent will listen to.

Changing the SNMP Agent Location

This command lets you enter a string of text that you can use to indicate 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 anything that you like provided that the location is less than 255 characters long.

Syntax Description

To enter the location of the SNMP agent, enter:

```
snmpAgentCfgUtil.sh location <location>
```

Where <*location*> is the character string identifying the agents location.

Examples

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

```
snmpAgentCfgUtil.sh location "equipment rack 5D"
```

Setting Up SNMP Contacts

This command lets you enter a string of text that you can use to identify 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 anything that you like up to 255 characters long.

Syntax Description

Use the following syntax to enter the specific SNMP agent contact information:

```
snmpAgentCfgUtil.sh contact <contact-info>
```

Where <*contact-info*> is the character string identifying the individual to contact concerning the SNMP agent.

Examples

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

```
snmpAgentCfgUtil.sh contact "Ace Duffy - ext 1234"
```

Listing SNMP Agent Settings

Usage Guidelines

This command lets you display all current SNMP settings.

Syntax Description

To display all current SNMP settings, enter:

```
snmpAgentCfgUtil.sh show
```

Examples

This sample output could be displayed after running this command.

```
# ./snmpAgentCfgUtil.sh show
Location : Washington_1
Contact : John
Port Number : 8001
Notification Type : trap
Notification Recipient Table :
[ Host IP address, Community, UDP Port ]
[ 10.10.10.1, public , 162 ]
Access Control Table :
Read Only Communities
baccread
Read Write Communities
baccwrite
```

Specifying SNMP Notification Types

Usage Guidelines

This command lets you specify which types of notifications (traps or informs) will be sent from the SNMP agent. By default, traps are sent although you can set this to send SNMP informs instead.

Syntax Description

Use this syntax to run this command:

snmpAgentCfgUtil.sh inform [retries timeout] |trap

Where the parameter is the back off timeout between retries.

Using the disk_monitor.sh Tool to Monitor Available Disk Space

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, located in the <BACC_HOME>/rdu/sample/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 Cisco recommends that, at a minimum, you use the disk_monitor.sh script to monitor the <BACC_DATA> and <BACC_DBLOG> directories.

To monitor available disk space:

 # ./disk_monitor.sh *(file system-directory) (x)*

Where:

- *(file system-directory)*—identifies any directory in a file system to monitor.
- *(x)*—identifies the percentage threshold applied to the specified file system.

Examples

Assume that you want to be notified when a file system (/var/CSCObpr for example) with database logs reaches 80% of its capacity. Enter the command using this syntax:

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

When the database logs disk space reaches 80% capacity, an alert is sent to the syslog file:

```
Dec 7 8:16:03 perf-u80-1 BPR: [ID 702911 local6.warning] File system /var/bpr usage is 81%
(threshold is 80%)
```

Troubleshooting Devices by MAC Address

You can use node management to troubleshoot a specific device, or a group of devices without turning logging on, and without searching through log files for device- or group-specific information.

BACC maintains a list of devices, based on MAC address, that contains detailed device information for troubleshooting problems. Troubleshooting information is stored centrally at the RDU and is maintained on a per-device basis; neither the DPE or 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 either the DPE or Network Registrar extension to the RDU is lost, this information is discarded until that connection is restored.

Note Any modifications to this list, such as the addition of a new device or group, take place immediately; there is no need to reboot the device.

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.

You can manage the device tracking list using the administrator's GUI to configure the number of devices that can be tracked at any given time; the default value is 100 devices. The tracking feature is considered to be turned off until a device is listed. See Device Management, page 9-3 for more information.

Caution Additional memory and disk space is required whenever this 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.