ダイナミック ファイル設定の管理

この章は、プライム ケーブル プロビジョニング がデバイス設定およびデバイス管理をサポートする次の機能を説明します。

Groovy スクリプト

このセクションでは、プライム ケーブル プロビジョニングがデバイス設定とデバイス管理に提供している Groovy スクリプト サポートについて説明します。このセクションの機能：

概要

プライム ケーブル プロビジョニングは、設定ファイルの生成のためテンプレートとは別に Groovy スクリプトを使用します。これは、DOCSIS、PacketCable、CableHome、OpenCable STB を含むプライム ケーブル プロビジョニング対応の CableLabs 標準に、動的ファイルを展開するのに役立ちます。このスクリプトインターフェイスでは、生成する TFTP ファイルを決定するのに役立つ検出済みの DHCP データとデバイス プロパティをにアクセスできます。プライム ケーブル プロビジョニング RDU は、テンプレートまたは Groovy スクリプトを使用して設定ファイルを生成します。RDU は、拡張子 .groovy で Groovy ファイルを識別します。Groovy

サンプル スクリプト ファイルは BPR_HOME/rdu/samples/groovy ディレクトリに存在し、テストに使用できます。

Groovy スクリプト ファイルを作成するには、テンプレートを作成するための要件に加えて、Groovy スクリプト言語を十分に理解している必要があります。

Groovy スクリプト言語

Groovy スクリプトは、次のオプションを含めることができます。

Groovy スクリプト

オプション	説明	例
<comment>	// [ascii-string] /* * 複数行コマンド */	// Config File Start/End / * configFile：型 DOCSISTFTPFile * services：型 ExtensionServices * discoveredData：型タイプ DHCPDataAccess * deviceProperties：型 CSRCProperties * option：型 DOCSISoptionfactory * device：型 IPDevice * context：型 ConfigContext */
<option-description>	<option-with-no-suboptions> | <compound-option>	
<option-with-no- sub options>	configFile.add (option.createOptionValue (<custom-value>, "<option-num>","<option-value>"));	configFile.add (option.createOptionValue ("3", "1")); カスタム値： configFile.add (option.createOptionValue (OptionSyntaxHEX., "217.53","010868446146484A4737"));

オプション	説明	例

<compound-option>	def = option
	.createOptionValue("");
	<variable-name>.add(option
	.createOptionValue(<custom-value>,
	"<optionnum>","<option-value>"));
	configFile.add(<variable-name>);
		def option24 = option.createOptionValue("24");
		option24.add(option.createOptionValue("24.8", "4194304"));
		configFile.add(option24);

<custom-value> optional	OptionSyntax.HEX |OptionSyntax.ASCII |OptionSyntax.SNMP	
<option-num>	<unsigned-byte>[.<unsigned-byte>]*	24
<option-value>	<option-value-string> [、[,]< オプション値文字列 >] *	「1」または [".docsDevNmAccessCommunity.1", "Octet String", "private"] as String[]

Groovy 環境に表示されているバインディングは次のとおりです。

- configFile：型 DOCSISTFTPFile
- services：型 ExtensionServices
- discoveredData：型 DHCPDataAccess
- deviceProperties：型 CSRCProperties
- option：型 DOCSISoptionfactory
- device：型 IPDevice
- context：型 ConfigContext

（注） CLI ファイル ユーティリティから Groovy スクリプトを実行中に、デバイス オブジェクト バインディングを使用できません。

（注） groovy スクリプトを作成中のコンパイル エラーを避けるため、次を使用します。

- 文字列に単一引用符（''）。

  例：`configFile.add (option.createOptionValue (OptionSyntax.SNMP,"11]、['.iso.org.dod.internet.private.enterprises.8595.2.1.2.10.1.2.2','STRING','msopassword']));`

- 特殊文字前に文字列に二重引用符（""）。

  例：`configFile.add (option.createOptionValue (OptionSyntax.SNMP,"11]、['.iso.org.dod.internet.private.enterprises.8595.2.1.2.10.1.2.2','STRING',"ms\$opassword」 ]));`

Groovy スクリプトをプライム ケーブル プロビジョニング RDU に追加する

プライム ケーブル プロビジョニング RDU に Groovy スクリプト ファイルを追加するには：

ステップ 1 **[Configuration] > [Files]** を選択します。[Vies Files] ページが表示されます。
ステップ 2 [Add] をクリックします。[Add Files] ページが表示されます。page appears
ステップ 3 **[File Type]** ドロップダウン リストから、CableLabs 設定スクリプト オプションを選択します。
ステップ 4 送信元のファイル名を参照します。
ステップ 5 [File Name] フィールドに、 <*filename* >.*groovy* ファイルを追加します。
ステップ 6 [Submit] をクリックします。`

Groovy に設定ファイル ユーティリティを使用する

設定ファイルユーティリティは、バイナリ設定ファイルにgroovyファイルを変換できます（その逆も可能）。設定およびgroovyファイルの表示と検証にも使用できます。設定ファイルユーティリティは、BPR_HOME/rdu/binディレクトリにインストールされます。Groovyファイルおよびバイナリ ファイルは、設定ファイル ユーティリティが呼び出されるディレクトリで使用可能である必要があります。

（注） プライム ケーブル プロビジョニングは、バイナリを groovy ファイルに生成するためにのみ（その逆も可能）設定ユーティリティを使用し、その他のスクリプト言語をサポートしません。

ここでは、次の内容について説明します。

- 設定ファイル ユーティリティの実行 （5 ページ）

設定ファイル ユーティリティの実行

設定ファイルユーティリティを実行するには、*BPR_HOME/rdu/bin*ディレクトリからコマンドを実行します。

runCfgUtil.sh オプション

利用可能なオプションは下記の通りです。

- **-?**：この使用のメッセージを印刷します。
- **-e**：BACC groovy/テンプレート ファイル（デフォルト）のエンコーディングを実行します。
- **-d**：バイナリ ファイルのデコードを実行します。
- **-g**：バイナリ ファイルから groovy/テンプレート ファイルの生成を実行します。
- **-snmp**：OptionSyntax.SNMP が有効になっているバイナリ ファイルから、テンプレートまたは動的スクリプト ファイルの生成を実行します。

- **-c***shared*：BACC groovy/テンプレート ファイル（デフォルトは cisco）を解析するときに使用する CMTS 共有秘密鍵。
- **-h** *host:port*：RDU がある場所を指定します（デフォルトでは localhost:49187）。
- **-i***device ID*：groovy/テンプレートの解析中に、マクロ変数置換を使用するデバイスを指定します。
- **m***macro*：groovy/テンプレートを解析するときの代わりに使用するマクロ変数を指定します。
- **-s**：人が読める形式で解析された groovy/テンプレートまたはバイナリ ファイルの内容を表示します。
- **o-***filename*：指定したファイル名で解析された groovy/テンプレートまたは人が読める形式の出力を保存します。
- **-l***filename*：ローカル ファイル システム上に存在する入力ファイルを指定します。
- **-r***filename*：RDU でリモート入力ファイルを指定します。
- **-pkt**：PacketCable MTA 設定ファイルとして処理するファイルを指定します。
- **-t***type*：PacketCable エンコーディング タイプを指定します（デフォルトでは Secure）。
- **-loc***locale*：na、euro、ietf など PacketCable ロケールを指定します（デフォルトでは na）。デフォルトは na です。MTA が欧州仕様 MTA の場合、ロケールは euro に設定する必要があります。
- **-cablehome**：CableHome 設定ファイルとして処理するファイルを指定します。
- **-docsis**：DOCSIS 設定ファイル（デフォルト）として処理するファイルを指定します。
- **-E**：拡張 CMTS MIC（EMIC）計算を有効にして、EMIC 計算のデフォルトのオプションを特定します。デフォルトのオプションは次の通りです。
  - HMAC タイプ：MMH16
  - EMIC ダイジェスト タイプ：明示
  - Cisco としての EMIC 共有秘密鍵。
- **-Ei**：CMTS MIC ダイジェスト サブタイプの拡張のために使用される [implicit] プレゼンテーションを指定します。
- **-Eh***HMACType*：拡張 CMTS MIC を計算するために使用するハッシュ アルゴリズムを指定します。サポートされているアルゴリズムは MD5 と MMH16 です（デフォルトでは MMH16）。
- **Es-**シークレット：拡張 CMTS MIC 計算に使用する CMTS 共有秘密鍵（デフォルトでは cisco）。
- **-u***username*：RDU に接続するときに使用するユーザー名を指定します。

- **-p***password*：RDU に接続するときに使用するパスワードを指定します。
- **-v***version*：入力ファイルを処理するテクノロジーのバージョンを指定します。
- **-prop***filename*：動的に使用される変数のキーと値を持つプロパティ ファイルを指定します。

（注） **-DDV4** または **-DDV6** のいずれかをファイル名 **-prop** とともに常に指定して、検出データを送信します。

- **-dis***filename*：フォーム キーと値ペアの動的なスクリプトで使用できる検出ータを指定します。
- **-DDv4***filename*：フォーム キーと値ペアの動的なスクリプトで使用できる検出 DHCPv4 データを指定します。
- **-DDv6***filename*：フォーム キーと値ペアの動的なスクリプトで使用できる検出 DHCPv6 データを指定します。
- **-cp***classpath*：動的スクリプトで参照される拡張 jar とスクリプト ファイルのパスを指定します。
- **b**：複数の出力ファイル生成するために、オプションの一括処理を指定します。特定のディレクトリ（-L オプションを使用）のすべてのバイナリ ファイルが処理され、生成されたファイルが -o オプションで示される出力ディレクトリで使用できます。
- **-ft**：ファイル タイプ（groovy または tmpl）を生成します。このオプションは、生成（-g オプション）生成に一括処理が有効になっているときに使用されます（-b オプションの使用）。（デフォルトのファイル タイプは tmpl です。）

runCfgUtil を使用した Groovy スクリプトの検証

プライム ケーブル プロビジョニング Groovy スクリプトをテストするための設定ファイル ユーティリティを使用するには。

ステップ 1 Groovy スクリプトを作成します。Groovy スクリプトを他の Groovy スクリプトに拡張する場合は、参照されているすべての Groovy スクリプトが同じディレクトリにあることを確認します。

ステップ 2 ローカル ファイル システムで設定ファイル ユーティリティを実行します。Groovy スクリプトの構文を確認するか、設定ファイル ユーティリティ プロセスが CRS として出力を返すように Groovy スクリプトを処理できます。

Groovy スクリプトに動的変数または検出されたデータが含まれている場合、指定された順序でこれらの操作を行います。

a) プロパティ ファイルとコマンド ライン代入をテストします。
b) RDU に追加されたデバイスでテストします。

ステップ 3 Groovy スクリプト（および使用される拡張 Groovy スクリプト）を RDU に追加します。

ステップ 4 設定ファイルユーティリティを実行して、ファイルを解析します。『外部 Groovy スクリプト ファイルの Groovy スクリプト処理をテストする』を参照してください。

Groovyスクリプトに動的変数または検出されたデータが含まれている場合、指定された順序でこれらの操作を行います。

a) プロパティ ファイルとコマンド ライン代入をテストします。
b) RDU に追加されたデバイスでテストします。

ステップ 5 すべてのテストが成功した後、Groovy スクリプトを使用するサービス クラスを設定します。

バイナリ ファイルを Groovy スクリプト ファイルに変換する

バイナリ設定メモリ ファイルを Groovy スクリプト ファイルに変換するには、runCfgUtil.sh コマンドを使用します。プライム ケーブル プロビジョニングの動的設定生成は、作成された Groovy スクリプトに基づくものです。既存のテスト済みバイナリ ファイルを Groovy スクリプトファイルに自動的に変換すると、プロセスを高速化し、エラーが発生する可能性を軽減します。

（注） **RunCfgUtil.sh** ツールを使用して、Groovy スクリプトに直接にテンプレートを変換することはできません（逆も不可能）。最初にテンプレートをバイナリ ファイルに変換して、Groovy スクリプトにバイナリ ファイルを変換する必要があります。テンプレートに Groovy スクリプトを変換する場合、最初に Groovy スクリプトをバイナリ ファイルに変換し、テンプレートにバイナリ ファイルを変換する必要があります。

構文の説明

runCfgUtil.sh -g -l*binary_file* **-o** *groovy_file*

- **-g**：Groovy スクリプト ファイルを入力バイナリ ファイルから生成する必要があることを指定します。
- **-l** *binary_file*：パス名を含むローカル入力ファイルを指定します。たとえば bronze.cm。
- **-o** *groovy_file*：パス名を含む出力 Groovy スクリプト ファイルを指定します。すべてのケースで、出力 Groovy スクリプト ファイルには .groovy ファイル拡張子があります。たとえば、test.groovy。

バイナリ ファイルを Groovy スクリプト ファイルに変換するには。

ステップ 1 ディレクトリを */opt/CSCObac/rdu/samples/docsis* に変更します。

ステップ 2 使用する Groovy スクリプト ファイルを選択します。この例では、*unprov.cm* と呼ばれる既存のバイナリ ファイルを使用します。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin # **runCfgUtil.sh -g -l unprov.cm -o test.groovy -docsis**

-docsis：入力ファイルが DOCSIS 設定ファイルになるように指定します。

MIB に依存することなくバイナリ ファイルを Groovy スクリプト ファイルに変換する

MIB に依存することなくバイナリ設定メモリ ファイルを Groovy スクリプト ファイルに変換するには、オプション -snmp を持つ**runCfgUtil.sh** コマンドを使用します。このコマンドを -snmp オプションで実行する場合、SnmpVarBind を含むすべての TLV の **OptionSyntax.SNMP** が追加されます。TLV のタイプに基づき正しい値を指定してください。例：整数値（1）。

（注） OID は数値形式であり、MIB の依存関係を完全に削除する必要があります。たとえば、.1.3.44491.1.2.3。

構文の説明

runCfgUtil.sh -docsis -g -snmp -l*binary_file* **-o** *groovy_file*

- **-g**：PacketCable MTA ファイルとして入力ファイルを特定します。
- **-snmp** ：バイナリ ファイルから生成された動的スクリプト ファイルで、OptionSyntax.SNMP が有効になることを指定します。
- **-l***binary_file*：パス名を含むローカル入力ファイルを指定します。すべてのケースで、入力バイナリ ファイル名には .cm ファイル拡張子があります。例：bronze.cm。
- **-o** *groovy_file*：パス名を含む出力 Groovy スクリプト ファイルを指定します。すべてのケースで、出力 Groovy スクリプト ファイルには .groovy ファイル拡張子があります。たとえば、test.groovy。

MIB と依存関係のない状態でバイナリ ファイルを Groovy スクリプト ファイルに変換するには。

ステップ 1 MIB をバックアップします。

ステップ 2 プロパティ `/snmp/mibs/mibList =` をディレクトリ `BPR_Home/api/conf/` にある api.properties に追加します。

（注） RDU からデバイス設定を生成する際、プロパティ `/snmp/mibs/mibList =` を `BPR_HOME/rdu/conf` ファイルにある rdu.properties に追加し、MIB の依存関係を避けます。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin # **runCfgUtil.sh -g-snmp-l example.cm -o example.cm.groovy**

- **example.cm**：入力バイナリ ファイルを特定します。

- **example.cm.groovy**：groovy ファイルの変換を特定します。

ローカル Groovy スクリプト ファイルの Groovy スクリプト処理をテストする

ローカル ファイル システムに保存されている Groovy スクリプト ファイルの処理をテストするには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -pkt -l *file*

- **-pkt**：PacketCable MTA ファイルとして入力ファイルを特定します。
- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。

ローカル ファイル システム上にある Groovy スクリプト ファイルを解析するには。

ステップ 1 */opt/CSCObac/rdu/samples/packet_cable* にディレクトリを変更します。

ステップ 2 使用する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *unprov_packet_cable.groovy* を使用します。これが PacketCable MTA Groovy スクリプトであるため、**-pkt**オプションが使用されます。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin/**runCfgUtil.sh -pkt -l unprov_packet_cable.groovy**

unprov_packet_cable.groovy：解析されない入力 t Groovy スクリプト ファイルを識別します。

外部 Groovy スクリプト ファイルの Groovy スクリプト処理をテストする

外部 Groovy スクリプト ファイルの処理をテストするには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -docsis -r *file* **-u** *username* **-p** *password*

- **-r**：RDU に追加されているファイルとして入力ファイルを識別します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*：RDU に接続するときに使用するパスワードを指定します。
- **-docsis**：DOCSIS Groovy スクリプトとしてファイルを特定します。

RDU に追加されている Groovy スクリプト ファイルを解析するには。

ステップ 1 使用する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *macro.groovy* を使用します。DOCSIS Groovy スクリプトが使用されているため、**-Docsis** オプションが使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -docsis -r unprov.groovy -u admin -p changeme
```

- **unprov.groovy**：入力ファイルを特定します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

ローカル Groovy スクリプト ファイルを処理し共有秘密鍵を追加する Groovy スクリプトのテスト

Groovy スクリプト ファイルの処理をテストし、指定した共有秘密鍵を追加するには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -docsis -l *file* **-c** *shared*

- **-e**：エンコード オプションを特定します。
- **-docsis**：DOCSIS Groovy スクリプト ファイルとして入力ファイルを識別します。
- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-c**：DOCSIS Groovy スクリプト ファイルを解析すると、CMTS 共有秘密を指定します。
- *shared*：共有秘密鍵を指定します。デフォルトの共有秘密鍵は、**cisco** です。

ローカルに保存されている Groovy スクリプト ファイルを解析するには、ユーザー指定の共有秘密鍵のを設定します。

ステップ 1 */opt/CSCObac/rdu/groovy* にディレクトリを変更します。

ステップ 2 解析する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *macro.groovy* を使用します。DOCSIS Groovy スクリプトが使用されているため、**-Docsis** オプションが使用されます。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -docsis -l unprov.groovy -c shared
```

- **unprov.groovy**：ローカル ファイル システムの入力ファイルを特定します。
- **shared**：新しい共有秘密鍵を特定します。

ローカル Groovy スクリプト ファイルを処理し EMIC共有 秘密鍵 を追加する Groovy スクリプトのテスト

Groovy スクリプト ファイルの処理をテストし、指定した拡張 CMTS MIC（EMIC）共有秘密鍵を追加するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -E -docsis -l *filename*

- **-E**：EMIC 計算を有効にします。
- **-docsis**：DOCSIS Groovy スクリプト ファイルとして入力ファイルを識別します。
- **-l***filename*：パス名を含む入力 Groovy スクリプト ファイルを指定します。すべてのケースで、入力 Groovy スクリプト ファイルには *.groovy* ファイル拡張子があります。たとえば、test.groovy。

デフォルト設定で EMIC を計算するには。

ステップ 1 使用する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *macro.groovy* を使用します。**-docsis** オプションは、DOCSIS Groovy テンプレートが使用されているため使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin # runCfgUtil.sh -E-l test.groovy
```

Specifying Dynamic Variables at the Command Line

動的変数を指定するには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -l *file* **-prop** *"file"*

- **-e**：エンコード オプションを識別します。
- **-l**：入力ファイルがローカル ファイル システム上にあることを指定します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-prop**：動的スクリプトで使用される変数のキーと値を持つプロパティ ファイルを指定します。

- *"file"*：対象の動的変数を特定します。複数の動的変数が必要な場合は、各キー値のペアをその他の後に指定する必要があります。

コマンド ラインで動的変数の値を指定するには。

ステップ 1 */opt/CSCObac/rdu/groovy* にディレクトリを変更します。

ステップ 2 使用する Groovy スクリプト ファイルを選択します。

ステップ 3 Groovy スクリプトの動的変数を特定します。

ステップ 4 変数の値を特定します。

ステップ 5 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -l macro.groovy -prop prop.properties
```

- **macro.groovy**：入力ファイルを特定します。
- **prop.properties**：キー値とペアが含まれています（例：MTA_PROP = 3）

Specifying a Device for Dynamic Variables

runCfgUtil.sh コマンドを使用して、動的変数のデバイスを指定します。

構文の説明

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**：エンコードオプションを識別します。キーを承認し、記載されてない場合デフォルトキーを使用します。
- **-r**：RDU に追加されているファイルとして入力ファイルを識別します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-i**：動的変数を解析するときに使用するデバイスを指定します。
- *MAC*：そのデバイスの MAC アドレスを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。

動的変数の置換に使用するデバイスを指定するには：

ステップ 1 使用する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *macro.groovy* を使用します。

ステップ 2 Groovy スクリプトの動的変数を特定します。

ステップ 3 使用するデバイスを識別します。この例では、デバイスが RDU に存在し、プロパティとして動的変数が設定されていることを前提としています。

ステップ 4 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin# **runCfgUtil.sh -e -r macro.groovy -i "1,6,00:01:02:03:04:05" -u admin -p changeme**

（注） -i オプションを使用してデバイス MAC を指定する場合、エンコード オプション（-e）と、入力ファイル オプション（-r）を一緒に記述します。

- **macro.groovy**：入力ファイルを特定します。
- **1,6,00:01:02:03:04:05**：デバイスのMACアドレスを識別します。ここで使用されるMACアドレスは、例のみを示します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

Specifying Discovered Data at the Command Line

検出されたデータを指定するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -l *file* **-dis** *"file"*

- **-e**：エンコードオプションを識別します。キーを承認し、記載されてない場合デフォルトキーを使用します。
- **-l**：入力ファイルがローカル ファイル システム上にあることを指定します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-dis**：フォーム キーと値ペアの動的なスクリプトで使用できる検出ータを指定します。
- *"file"*：適切な検出データを特定します。

コマンド ラインで検出されたデータの値を指定するには。

ステップ 1 使用する Groovy スクリプト ファイルを選択します。
ステップ 2 Groovy スクリプトで検出されたデータを特定します。
ステップ 3 検出されたデータの値を特定します。
ステップ 4 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin# **runCfgUtil.sh -e -l macro.groovy -dis dis.properties**

- **macro.groovy**：入力ファイルを特定します。
- **dis.properties**：キー値とペアが含まれています（例：giaddr 10.1.1.9 =）。

Specifying a Device for Discovered Data

デバイスを指定し、設定ファイル生成のた検出されたデータを使用するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**：エンコードオプションを識別します。キーを承認し、記載されてない場合デフォルトキーを使用します。
- **-r**：RDU に追加されているファイルとして入力ファイルを識別します。
- *file*：解析されない入力 t Groovy スクリプト ファイルを識別します。
- **-i**：検出されたデータを解析するときに使用するデバイスを指定します。
- *MAC*：そのデバイスの MAC アドレスを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。

検出されたデータの代入に使用するデバイスを指定するには。

ステップ 1 使用する Groovy スクリプト ファイルを選択します。この例では、既存の Groovy スクリプト ファイル *macro.groovy* を使用します。

ステップ 2 Groovy スクリプトで検出されたデータを特定します。

ステップ 3 使用するデバイスを識別します。この例では、デバイスが RDU に存在し、プロパティとして検出されたデータが設定されていることを前提としています。

ステップ 4 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin# **runCfgUtil.sh -e -r macro.groovy -i "1,6,00:01:02:03:04:05" -u admin -p changeme**

- **macro.groovy**：入力ファイルを特定します。
- **1,6,00:01:02:03:04:05**：デバイスの MAC アドレスを識別します。ここで使用される MAC アドレスは、例のみを示します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

Groovy からバイナリ ファイルを生成する

runCfgUtil.sh コマンドを使用して、バイナリ ファイルとして解析された Groovy スクリプトの出力を指定します。

構文の説明

runCfgUtil.sh -l *input_file* **-o** *output_file*

- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。
- *input_file*：解析される入力 Groovy スクリプト ファイルを特定します。
- **-o**：解析された Groovy スクリプト ファイルがバイナリ ファイルとして保存されることを指定します。
- *output_file*：解析された Groovy スクリプト ファイルのバイナリの内容が保存されているファイルの名前を特定します。

Groovy スクリプトからバイナリ ファイルに解析する出力を指定します。

ステップ 1 使用する Groovy スクリプト ファイルを選択します。

ステップ 2 出力ファイル名を特定します。この例では、*unprov.cm* を使用しています。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin# **runCfgUtil.sh -l unprov.groovy -o unprov.cm**

- **unprov.groovy**：バイナリ ファイルに解析される既存の Groovy スクリプト ファイルを特定します。
- **unprov.cm**：使用される出力ファイル名を特定します。

ローカル バイナリ ファイルの表示

詳細については、『ローカル バイナリ ファイルの表示』を参照してください。

外部バイナリ ファイルの表示

詳細については、『外部バイナリ ファイルの表示』を参照してください。

PacketCable Basic フローのアクティブ化

詳細については、PacketCable Basic フローのアクティブ化 （68 ページ）を参照してください。

マルチベンダー サポート用 TLV 43s の生成

詳細については、『マルチベンダー サポート用 TLV 43s の生成』を参照してください。

動的 TFTP ファイル命名規則

TFTP ファイルの命名規則を使用して、動的 TFTP ファイル名とその順序の変数コンポーネントをカスタマイズできます。Groovy スクリプトは、公開されている DHCP 検出データおよびその他のインターフェイスなどのコンポーネントを使用して、TFTP ファイル名を生成します。

スクリプトは、サービス クラス名、検出されたベンダー名、ダウン ストリーム速度など重要な情報を含むことができます。テクノロジー デフォルトまたはシステム デフォルトのどちらかでスクリプトを設定できます。デフォルトでは 127 文字まで指定できます。

CableLabs 設定ファイル名スクリプトが変更された場合は、変更を反映するため影響を受けるデバイスのリストの設定の再生成がトリガーされます。

BAC_HOME/rdu/samples/groovy で Groovy スクリプトのサンプルを確認できます。

例 *19-1* サンプル *TFTP* ファイル名 *Groovy* スクリプト

```
/**
* example_extended_filename.groovy
*
* A sample CableLabs Configuration Filename Script that demonstrates how
* to create an extended filename. This example includes the following
* strings in the extended filename: DeviceType, Selected ClassOfService,
* and provisioning group. For DOCSIS device types, the default DOCSIS
* version is included after device type. The resulting extended filename
* string is:
*
* "<device-type>_<default-docsis-version>_<selected-cos>_<pg>"
* (e.g., "cm_11_goldcos_westpg", "pc_silvercos_eastpg").
*
* A CableLabs Configuration Filename Script specifies an extended filename
* label that is appended to the standard BAC dynamic configuration filename.
* In BAC 4.2 and later releases, the dynamic configurations have a filename
* consisting of the fixed/standard prefix. The script can be configured at
* System Defaults (preferred) and/or Technology Defaults.
*
* BAC properties:
* DocsisDefaultKeys.DOCSIS_DEFAULT_VERSION
*
* Variable bindings:
* configFileName - Extended Filename of type StringBuilder
* services - of type ExtensionServices
* discoveredData - of type DHCPDataAccess
* deviceProperties - of type CSRCProperties
* device - of type IPDevice
* context - of type ConfigContext
*/
import com.cisco.provisioning.cpe.constants.DocsisDefaultKeys
import com.cisco.provisioning.cpe.extensions.constants.CNRNames
import com.cisco.provisioning.cpe.extensions.services.DeviceType
/*
* A Groovy list is used to collect the ordered list of string fields
* that will comprise the extended filename. Once all the fields have been
* added to the list, the "join" method is used to concatenate the
* fields with a underscore ('_') separator character.
*/
def label = []
/*
* Add Device Type (abbreviated).
*
* The device type string is too verbose for a filename component, so an
* abbreviation is used instead. For example, the DOCSIS device type value
* "DOCSISModem" is abbreviated as "cm". If no abbreviation is defined,
* a default abbreviation of "xx" is used.
*/
def deviceType = device.getDeviceType().getName()
def deviceTypeMap = [
(DeviceType.DOCSIS_MODEM) : "cm",
(DeviceType.PACKET_CABLE_MTA) : "pc",
```

```
(DeviceType.CABLEHOME_WAN_MAN) : "ch",
(DeviceType.STB) : "st",
(DeviceType.CUSTOM_CPE): "cu"
]
label << deviceTypeMap[deviceType] ?: "xx"
/*
* Add default DOCSIS version number (exclude embedded "dot").
*
* For DOCSIS device types, the default DOCSIS version number specifies the
* maximum DOCSIS version supported by the CM and CMTS. This version number
* indicates the DOCSIS version grammar used when constructing the dynamic
* configuration file. The embedded "dot" is stripped from the version number
* (i.e., "3.0" --> "30").
*/
if (deviceType == DeviceType.DOCSIS_MODEM)
{
label << deviceProperties.getProperty(
DocsisDefaultKeys.DOCSIS_DEFAULT_VERSION, "1.0") - "."
}
/*
* Add Selected Class of Service name.
*/
label << device.getSelectedClassOfService().getClassOfServiceName()
/*
* Add Provisioning Group name.
*/
label << device.getProvGroup().getProvGroupId()
/*
* Convert the list of filename components into a string value with underscore
* ('_') characters separating the filename components. Add the resulting
* string to the configFileName StringBuilder binding.
*/
configFileName << label.join("_")
```

拡張機能を使用した動的 TFTP ファイルの名前付け

TFTP ファイルのネーミング機能は、RDU 拡張レベルでカスタマイズをサポートするため強化されています。プライム ケーブル プロビジョニング 5.2 リリース以前は、TFTP ファイル名は TFTP ファイル名生成 Groovy スクリプトを使用して、動的にのみ名前付けが可能です。プライム ケーブル プロビジョニング 5.2 リリースでは、サービス レベルの拡張機能、設定の生成拡張機能、設定の生成スクリプトで使用可能な共有コンテキストで、ユーザーが設定ファイル名を設定できます。

共有コンテキストで設定されているファイル名は、TFTP ファイル名の生成 Groovy スクリプトで設定されているファイル名上の、より優先順位の高いテクノロジーまたはシステムデフォルト レベルで設定できます。

共有コンテキスト ファイル名

共有コンテキスト（`com.cisco.provisioning.cpe.extensions.configuration.SharedConfig`）が ConfigContext （com.cisco.provisioning.cpe.extensions.configuration.ConfigContext）に含まれており、サービス レベル拡張、設定生成拡張、設定生成 Groovy スクリプトで共有されます。新規の共有コンテキスト オブジェクトでは、ファイル名を設定するために使用可能な設定方法（`setConfigFileName (String)`）があります。次のサンプル コード スニペットでは、拡張の共有コンテキストで設定されているファイル名を説明します。

```
SharedConfig sharedConfig = configContext.getSharedConfig();
sharedConfig.setConfigFileName("<fillup_the_filename_here>");
```

ファイル名への設定と取得方法に加えて、共有コンテキストオブジェクトにはマップオブジェクトがあり（取得方法<Map<String, Object> getSharedConfigMap() を使用してアクセス可能）、データを保存し拡張間で共有するためのコンテナとして使用できます。

共有コンテキストはデバイス検出コンテキスト（com.cisco.provisioning.cpe.extensions.detection.DeviceDetectionContext）に含まれており、デバイス検出拡張機能で使用できます。デバイス検出拡張機能により入力されている共有コンテキストは、サービス レベル拡張機能と設定生成拡張機能で共有される設定コンテキストで事前入力されます。共有コンテキストがデバイス検出拡張機能により入力されている場合、設定コンテキストで事前入力され、サービス レベル拡張機能と設定生成拡張機能で使用可能です。

デバイス中断コンテキスト（com.cisco.provisioning.cpe.extensions.disruption.DeviceDisruptionContext）は、共有コンテキストをカプセル化します。ただし、これは設定コンテキストまたはファイルの名前付けには関係ありません。デバイス中断コンテキストに含まれる共有コンテキスト（com.cisco.provisioning.cpe.extensions.configuration.SharedContext）は、複数のデバイス中断拡張機能で使用できます。

動的 TFTP ファイル名の生成の基本的なフロー

ファイル名生成の基本的なフローを説明する一連の手順を次に示します。

1. RDU は設定の生成要求を受信します。
2. 生成拡張によりデバイスに割り当てる必要がある動的 TFTP ファイルを決定している間、RDU はデバイスの設定生成を実行します。ファイルはテンプレート（例：docsis.tmpl）またはスクリプト（例：docsis.groovy）になる可能性があります。（例：silver.cm）
3. すべてのカスタム拡張機能を実行後、設定エンジンは共有コンテキスト内のファイル名の可用性を検索します。ファイル名に共有コンテキストとプロビジョニング グループ機能が入力されている場合、*/provgroup/capability/dpe/tftp/filename/extensions" (ProvGroupCapabilitiesKeys.TFTP_DYNAMIC_FILENAME_USING_EXTENSIONS)* が有効になり、その後手順 5 に従います。ファイル名が共有コンテキストに入力されていない場合、手順 4 で説明したように拡張 TFTP ファイル名の Groovy スクリプト ワークフローに従ってファイル名の生成を行います。
4. 生成拡張は、テクノロジー デフォルトの CableLabs 設定ファイル名のスクリプト プロパティを検索します。見つからない場合、システムデフォルトに保存されます。このプロパティの値は、実行するスクリプトの名前です。スクリプトが実行され、動的 TFTP ファイル名に含まれる追加の文字列を返します。
5. 生成拡張はこの値を使用し、デバイス用の動的 TFTP ファイル名を作成します。ファイル名の生成が完了した後、設定は DPE に送信されキャッシュされます。

共有コンテキストを介したファイル名生成に関する基本ポインタ

- TFTP ファイル名 Groovy スクリプト機能と同様に、共有コンテキストに入力される名前には実際の TFTP 設定ファイル名の動的な部分が含まれます。
- デフォルトのファイル名の最大長は 127 文字であり、ファイル名は制限を超える場合切り捨てられます。スペースと特殊文字の削除は、既存の TFTP 動的ファイル名の生成機能と同じです。
- ファイル名がカスタム拡張子または設定生成スクリプト別に共有コンテキストに入力されなかった場合、ファイル名の生成は設定ファイル名生成 Groovy スクリプト ワークフロー経由で既存の TFTP 生成に従います。

共有コンテキストのファイル名を入力するサンプル サービス レベルの拡張機能

```
package com.cisco.provisioning.cpe.extensions.samples;

import java.util.Map;

import com.cisco.provisioning.cpe.extensions.ExtensionException;
import com.cisco.provisioning.cpe.extensions.configuration.ConfigContext;
import com.cisco.provisioning.cpe.extensions.configuration.ServiceLevel;
import com.cisco.provisioning.cpe.extensions.configuration.SharedConfig;
import com.cisco.provisioning.cpe.extensions.services.ExtensionServices;
import com.cisco.provisioning.cpe.extensions.services.IPDevice;

/**
* This class will demonstrate usage of Shared Context and the filename
* availability in the configuration context object. The PCP extensions will
* make use of the Shared Context and also have the feasibility of setting the
* filename from the extensions.
*
*/
public class SampleSharedContextServiceLevelSelectionExtension implements
com.cisco.provisioning.cpe.extensions.configuration.ServiceLevelSelector
{
@Override
public void selectServiceLevel(IPDevice device,
ConfigContext configContext, ServiceLevel serviceLevel,
ExtensionServices extensionServices) throws ExtensionException
{
final String extensionName =
SampleSharedContextServiceLevelSelectionExtension.class.getName();
com.cisco.provisioning.cpe.extensions.services.LogManager logManager =
extensionServices.getLogManager();
logManager.log(
com.cisco.provisioning.cpe.extensions.services.LogLevel.INFO,
extensionName, " Executing custom code here..");
long duration = System.currentTimeMillis();
/*
* Get the SharedConfig object from ConfigContext
*/
SharedConfig sharedConfig = configContext.getSharedConfig();
/*
* Get the sharedContext map from SharedConfig object
*/
Map<String, Object> sharedContextMap =
sharedConfig.getSharedConfigMap();
/*
* Set the generated token into the Shared Context map
```

```
*/
sharedContextMap.put("customToken_1", generateTokenXYZ());
logManager.log(
com.cisco.provisioning.cpe.extensions.services.LogLevel.INFO,
extensionName, " Shared Context Map after token generation "
+ sharedContextMap);
/*
* Setting the file name in SharedConfig object
*/
logManager.log(
com.cisco.provisioning.cpe.extensions.services.LogLevel.INFO,
extensionName, " Settinng the file name here...");
sharedConfig.setConfigFileName("SetBy" + this.getClass().getCanonicalName());
/*
*
* Set a token that is used by sample configuration script
* example_shared_context.groovy
*/
sharedContextMap.put("maxCPE","7");
logManager.log(
com.cisco.provisioning.cpe.extensions.services.LogLevel.INFO,
extensionName, " Shared Context Map after new token "
+ sharedContextMap);
logManager.log(
com.cisco.provisioning.cpe.extensions.services.LogLevel.INFO,
extensionName, " It took ["
+ (System.currentTimeMillis() - duration)
+ "] milliseconds to complete this extension ");
}
/**
* This method models a custom token generation
* @return the generated token as String
*/
private String generateTokenXYZ()
{
/*
* operation.
*/
return "dummyTokenValue";
}
}
¢
```

Sample Configuration Script to access Shared Context and populate filename in Shared Context

```
/**
* A sample CableLabs Configuration Script that demonstrates the Shared Context
availability in Groovy script. And also it demonstrate to populate the filename in Shared
 Context
*
*
* PCP properties:
* DocsisDefaultKeys.DOCSIS_DEFAULT_VERSION
* SNMPPropertyKeys.READ_COMMUNITY_STRING
* SNMPPropertyKeys.WRITE_COMMUNITY_STRING
*
* Variable bindings:
* configFile - of type DOCSISTFTPFile
* option - of type DOCSISOptionFactory
* services - of type ExtensionServices
* discoveredData - of type DhcpDataAccess
* deviceProperties - of type CSRCProperties
* device - of type IPDevice
* context - of type ConfigContext
*
```

```
* @see com.cisco.provisioning.cpe.extensions.configuration.DOCSISTFTPFile
* @see com.cisco.provisioning.cpe.extensions.configuration.DOCSISOptionFactory
* @see com.cisco.provisioning.cpe.extensions.services.ExtensionServices *
* @see com.cisco.provisioning.cpe.extensions.configuration.DhcpDataAccess
* @see com.cisco.provisioning.cpe.extensions.services.CSRCProperties
* @see com.cisco.provisioning.cpe.extensions.services.IPDevice
* @see com.cisco.provisioning.cpe.extensions.configuration.ConfigContext
*/
import com.cisco.provisioning.cpe.constants.SNMPPropertyKeys
import com.cisco.provisioning.cpe.extensions.configuration.SharedConfig
def TLV = option.&createOptionValue

/* 1. Accessing Shared Context */
/*
* This script tries to set the maxCPE value based on the tokens available at the Shared
 Context
* Populate the maxCPE property in Shared Context by using extensions to use this
demonstration.
/*
* Get the SharedConfig object from ConfigContext
*/
SharedConfig sharedConfigObj = context.getSharedConfig();
/*
* Get the sharedContext map from SharedConfig object
*/
Map<String, Object> sharedContextMapObj =
sharedConfigObj.getSharedConfigMap();
def maxCPE = TLV("18", "3")
if (sharedContextMapObj.containsKey("maxCPE"))
{
maxCPE = TLV("18", (String)sharedContextMapObj.get("maxCPE"))
}
configFile.add(maxCPE)

/* 2. Filename can also be set at this configuration script by using the Shared Context
*/
/** Sets the dynamic filename */
sharedConfigObj.setConfigFileName("samplenamesetByCfgScript");
```

テンプレート

このセクションでは、プライム ケーブル プロビジョニング がデバイス設定とデバイス管理をサポートするテンプレートを説明しています。このセクションの機能：

- テンプレート ファイル：概要 （23 ページ）
- テンプレート文章
  - SNMP VarBind
  - マクロ変数
  - SNMP TLV
  - 定義されたオプションのエンコーディング タイプ
- テンプレートの設定ファイルユーティリティの使用

テンプレート ファイル：概要

プライム ケーブル プロビジョニング テンプレートを使用して、動的 PacketCable、DOCSIS、および CableHome ファイルを導入するのに役立ちます。テンプレートを使用して、簡単に判読可能な形式でテンプレート ファイルを作成し、迅速かつ簡単に編集できます。テンプレートは、PacketCable、DOCSIS、または CableHome オプションと、有効な PacketCable、DOCSIS、または CableHome ファイルを生成するために使用される値を表す ASCII テキスト ファイルです。プライム ケーブル プロビジョニング はテンプレート ファイルを識別するため *.tmpl* 拡張子を使用します。サービス クラスを参照する前に、管理者ユーザー インターフェイスまたはアプリケーション プログラミング インターフェイス（API）を使用して、ファイルとして RDU にテンプレート ファイルを追加する必要があります。

プライム ケーブル プロビジョニング RDU コンポーネントをインストールする際、いくつかのサンプル テンプレート ファイルが *BPR_HOME/rdu/templates* ディレクトリにコピーされます。

テンプレートの作成や編集に必要なのはシンプルなテキスト エディタのみですが、独自のテンプレート ファイルを作成する前に、この情報を十分に熟知する必要があります。

- プロビジョニング Cisco のフロー プライム ケーブル プロビジョニング
- DOCSIS 1.0、1.1、2.0、3.0、3.1 および RFI 仕様
- DOCSIS レイヤ 2 仮想プライベート ネットワーク仕様
- PacketCable 1.0、1.5 および 2.0 仕様
- マルチメディア終端アダプタ（MTA）デバイス プロビジョニング仕様
- CableHome 1.0 の仕様
- ケーブルのデバイスの SNMP MIB（例：DOCS-CABLE-DEVICE-MIB）

テンプレート文章

テンプレートには、次のタイプのステートメントで構成されています。

- 説明（Comments） （24 ページ）
- Includes （24 ページ）
- オプション（Options） （25 ページ）
- インスタンス修飾子 （27 ページ）
- OUI 修飾子 （27 ページ）

コメントでテンプレートを文書化することができます。他のテンプレートで使用される構成要素テンプレートを作成することができます。わかりやすい方法で PacketCable、DOCSIS、または CableHome タイプの長さの値（TLV）を指定するため、オプションを使用します。特定の個別 TLV に複合オプションをグループ化するインスタンス修飾子を使用することができます。OUI 修飾子を使用すると、ベンダー固有の情報を含めることができます。使用できるテンプレート文章オプションは次のとおりです。

表 *1:* テンプレート文章

オプション	説明
<comment>	::= #[ascii-string]
<include>	::= include “<filename.tmpl>”
<option-description>	::= option <option-num> [instance <instance-num>] [oui <oui>] <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>	::= <snmpvar-oid>,<snmpvar-type>,<snmpvar-value>

説明（Comments）

コメントは情報のみを入力し、シャープ（#）記号と行の最後の間に常に存在します。次の例では、サンプル コメントの使用を示します。

使用例

```
#
# Template for gold service
#

option 3 1 # enabling network access
```

Includes

ファイルを含めることで同様の階層を構築できますが、若干異なるテンプレートになります。これは複数のテンプレートのオプションが重複することなく、多くのサービス クラスで一般的なオプションを定義するのに非常に便利です。

単一のテンプレートには複数の含有ステートメントを使用できますが、テンプレートの含有ステートメントの場所は重要です。含有ステートメントがテンプレートで確認される場合は、常に含有ファイルの内容が含まれます。含有テンプレートは使用する前に RDU にファイルとして追加される必要があります。テンプレートは RDU データベースでパス情報がない状態で保存されるため、含有ファイルには ../.. などの場所の修飾子を含まないようにしてください。Example 19-3 and Example 19-4 では、含有オプションの正しい使用法と誤った使用法の両方を説明しています。

正しい含有ステートメントの使用法

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

誤った含有ステートメントの使用法

```
# 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
```

オプション（Options）

PacketCable、DOCSIS、および CableHome 設定ファイルは、適切にエンコードされたオプション ID と値のペアで構成されています。2 つのオプション形式がサポートされます。 定義済みおよびカスタム。

- 明確に定義されたオプションは、オプションの数と値が必要です。オプション番号のエンコーディング タイプに基づいて、値がエンコードされています。
- カスタム オプションはオプション番号、明示的な値のエンコーディング タイプ、および値が必要です。

オプション 43 など複合オプションを使用すると、TLV グループを指定するためにインスタンス修飾子を使用できます。インスタンス修飾子 （27 ページ）を参照してください。

テンプレートでこれらの明確に定義されたオプションのいずれかを指定すると、値のエンコーディング値を指定する必要はありません。これらの定義されたエンコーディング タイプの詳細については、定義されたオプションのエンコーディング タイプ （36 ページ） および テクノロジー オプション サポート を参照してください。

カスタム オプション（例：オプション 43）を指定するときに、オプションのエンコーディングのタイプを指定する必要があります。使用可能なエンコーディング タイプを示します。

- ASCII：NULL ターミネータを必要とせず、ASCII 文字列として ASCII タイプは特定の値をエンコードします。値にスペースを含む場合、二重引用符で囲みます。

- 16 進数：値は有効な 16 進数であり、各オクテットに 2 文字である必要があります。01 を値として指定する場合、1 個のオクテットがエンコーディングで使用されます。0001 を値として指定する場合、2 つのオクテットがエンコード プロセスで使用されます。
- IP アドレス：IP アドレス タイプは、4 つのオクテットとして特定の値をエンコードします。たとえば、IP アドレス 10.10.10.1 は 0A0A0A01 としてエンコードされます。
- SNMPVarBind：SNMP OID 文字列、タイプ、および値。これらそれぞれはカンマ区切りです。

特定のラインで複数のオプションにカンマを使用します。各値は各自処理されるため、二重引用符で値のいずれかを囲む必要があります（他は必要ありません）。複数値を持つオプションの良い例は、オプション 11（SNMP VarBind）です。詳細については、SNMP VarBind （29 ページ）を参照してください。

複合オプションを指定するときに、最上位レベル オプションを指定する必要はありません（例：オプション 4.1 を指定するときにオプション 4 など) を指定する必要はありません。**Correct Option Statement Usage** および **Incorrect Option Statement Usage** は、オプション ステートメントの正しい使用法と誤った使用法の両方を示しています。

正しいオプション ステートメントの使用法

```
# 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
```

誤ったオプション ステートメントの使用法

```
# 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
```

インスタンス修飾子

インスタンス修飾子は特定の個別の Type-Length-Values（TLV）に復号オプションをグループ化するために使用します。Example と Example は、別の TLV の作成に正しい方法と誤った方法の両方を示しています。これらは、2 つの独立したコマンドとして IOS コマンドを解釈する IOS DOCSIS モデムを有効にする必要があります。

例：正確な *IOS* コマンド ライン エントリ

```
# 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"
```

例：不正な *IOS* コマンド ライン エントリ

```
# 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
```

（注） オプション 43.8 のエンコーディング タイプは、組織的固有識別子（OUI）です。Example で示すものとは異なり、00-00-0 C 形式のみ受け入れられます。

OUI 修飾子

OUI 修飾子により、オプション 43 およびそのサブオプションを使用した複数ベンダーのサポート機能が向上します。

プライム ケーブル プロビジョニング では、多くのベンダーからさまざまな TLV 43 を指定する単一テンプレートを使用することができます。Correct OUI Modifier Usage の例では、XX-XX-XX として次で OUI 形式を指定します。

- FF-FF-FF：DOCSIS の一般的な拡張にエンコーディングを指定するには、ベンダー ID を特定します。
- 00-00-0C：Cisco 固有のケーブル モデム オプション 43 とそのサブオプションを指定する Cisco ベンダー ID を特定します。

Correct OUI Modifier Usage の例では、ケーブル モデム設定ファイルを使用した L2VPN の プライム ケーブル プロビジョニング サポートを説明し、L2VPN のアップストリーム トラフィックを分類します。このテンプレートの内容を使用して、subTLV を生成することができます。

- OUI=FF-FF-FF を使用した、DOCSIS の一般的な拡張エンコーディングからの 43.5.1 および 43.5.2.2。
- OUI=00-00-0C を使用した、Cisco 固有のオプション 43 からの 43.1。

ただし、DOCSIS 仕様を遵守するためには、TLV 43 の最初の subTLV として挿入する必要があります。

- 一般的な拡張情報をエンコードする DOCSIS 拡張機能フィールドを使用する場合は 0xFFFFFF。
- Cisco 固有の subTLV の生成時に 0x00000C。

正しい *OUI* 修飾子の使用法

```
# Upstream L2VPN Classifier Example

# This example shows how to classify upstream traffic from a specific CPE
# onto an upstream L2VPN service flow, in which other CPE attached to
# the cable modem forward to the non-L2VPN forwarder, as depicted below.

# This example also demonstrates that when using the DOCSIS extension
# field (TLV 43) to encode general extension information (GEI), you do
# not need to specify oui=FF-FF-FF. You only need to specify the OUI tag when
# general extension encoding is not used and vendor-specific encoding is used.

# Upstream L2VPN Classifier Cable Modem Config File

# (43) Per-CM L2VPN Encoding
# GEI (43.8) Vendor ID : 0xFFFFFF for GEI
option 43.8 instance 1 ff-ff-ff

# GEI (43.5) for L2VPN Encoding
# GEI (43.5.1) VPNID Subtype
option 43.5.1  instance 1 0234560003

# GEI (43.5) for L2VPN Encoding
# GEI (43.5.2) IEEE 802.1Q Format Subtype
# VLAN ID 25
option 43.5.2.2 instance 1 25

# Cisco Specific Vendor Option Encodings
# (43.8) Vendor ID : 00-00-0C (Cisco Vendor ID)
option 43.8 instance 2 00-00-0C

# Cisco Vendor Specific option (43.1)
# Static Downstream Frequency
# Frequency 402750000
option 43.1 instance 2 oui 00-00-0C 402750000

# Cisco Specific Vendor Option Encodings
# (43.8) Vendor ID : 00-00-0C (Cisco Vendor ID)
option 43.8 instance 3 00-00-0C

# Cisco Vendor Specific option (43.3)
# Update Boot Monitor Image
# image name (boot_monitor_image.bin)
option 43.3 instance 3 oui 00-00-0C boot_monitor_image.bin
```

次の例は、OUI 修飾子の間違った使用方法を説明しています。

例 *1*

```
# Invalid, OUI tag needs to be present for each 43 suboption if/when general extension
# encoding is not used and vendor-specific encoding is used.
```

```
option 43.8 00-00-0C

option 43.3 boot_monitor_image.bin
```

例 *2*

```
# Invalid, when both OUI and instance modifier are used in authoring a template,
# "instance" modifier needs to occur before "oui" modifier.

option 43.8 instance 1 00-00-0C

option 43.3 oui 00-00-0C instance 1 boot_monitor_image.bin
```

SNMP VarBind

DOCSIS オプション 11、PacketCable オプション 64、または CableHome オプション 28 を指定する場合は、オブジェクト ID（OID）を使用する必要があります。OID を含む MIB は、RDU によりロードされている次の MIB のいずれかに存在する必要があります。識別のために、必要に応じて OID の大半を指定する必要があります。OID の名前または番号を使用することができます。RDU は自動的にこれらの MIB をロードします。

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

eRouter Mib

ipNetToPhysicalTable [RFC 4293]	**IP-MIB**
vacmAccessTable [RFC 3415]	SNMP-VIEW-BASED-ACM-MIB
vacmSecurityToGroupTable [RFC 3415];	
vacmViewTreeFamilyTable [RFC 3415];	
vacmAccessReadViewName [RFC 3415];	
vacmAccessWriteViewName [RFC 3415];	
snmpCommunityTable [RFC 3584];	SNMP-COMMUNITY-MIB
snmpTargetAddrTMask [RFC 3584];	
snmpTargetAddrExtTable [RFC 3584];	

snmpTargetAddrTable [RFC 3413]	SNMP-TARGET-MIB
snmpTargetAddrTAddress [RFC 3413];	
esafeErouterInitModeControl [eDOCSIS].	**ESAFE-MIB.mib(upgrade)**

DOCSIS Mib

これらの DOCSIS Mib は RDU に読み込まれます。

- DOCS-IF-MIB
- DOCS-BPI-MIB
- CISCO-CABLE-SPECTRUM-MIB
- CISCO-DOCS-EXT-MIB
- SNMP-FRAMEWORK-MIB
- DOCS-CABLE-DEVICE-MIB
- CISCO-CABLE-MODEM-MIB

（注） Cisco BAC 4.1 で、DOCS-CABLE-DEVICE-MIB（mib2 branch）を廃止するため、DOCSIS MIB、DOCS-CABLE-DEVICE-MIB-OBSOLETE（実験ブランチ）は、RDUのデフォルトでロードされたMIBリストから削除されます。

Cisco BAC 4.2 で、CLAB-DEF-MIB and DOCS-BPI2-MIB が個別に重複しているため、CL-SP-MIB-CLABDEF-I02-020920 and DOCS-BPI2-MIB-ipcdn-08 MIBs は、RDU のデフォルトでロードされた MIB リストから削除されます。

上記から削除された MIB から完全修飾 MIB OID への参照は、顧客テンプレートとスクリプトで新しい MIB から適切な OID を置換する必要があります。ただし、これらの実験OIDを含めるカスタム MIB オプションを使用できます。カスタム MIB の追加についての詳細は、『ベンダー固有の MIB を持つ SNMP TLV の追加』を参照してください。

PacketCable MIB

これら PacketCable（北米）MIB は、RDU にロードされます。

- CLAB-DEF-MIB
- PKTC-MTA-MIB
- PKTC-SIG-MIB
- PKTC-イベント-MIB

CableHome MIB

これら CableHome MIB は、RDU にロードされます。

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

これらの追加 MIB は必要ですが、プライム ケーブル プロビジョニング製品の一部ではありません。

- 16 CABH-CTP-MIB には RMON2-MIB、TOKEN-RING-RMON-MIB が必要です
- CABH-SEC-MIB には DOCS-BPI2-MIB が必要です

マクロ変数

マクロの変数は、デバイス固有のオプション値を指定するテンプレートの値として指定されます。テンプレートでマクロ変数が発生した場合、プロパティ階層によりマクロ変数名が検索され、変数の値が置換されます。変数名は、RDU で事前に定義されているカスタム プロパティです。スペースを含めることはできません。

カスタム プロパティが定義されると、このプロパティ階層で使用できます。

- デバイス プロパティ
- プロビジョニング グループ プロパティ
- サービス クラス プロパティ
- DHCP 条件プロパティ
- PacketCable、DOCSIS、または CableHome などのテクノロジー デフォルト
- システム デフォルト（System defaults）

階層にプロパティが存在する（最初にデバイス、次にサービス クラスなど）場合、テンプレート パーサー動作はボトムアップ式になり、テンプレート オプション構文を変換します。次の構文はマクロ変数でサポートされています。

- ${var name}：この構文はそのまま置換されます。変数が見つからない場合、パーサーはエラーを生成します。
- ${var-name, ignore}：変数値がプロパティ階層で見つからない場合、構文によりテンプレート パーサーはこのオプションを無視します。

- ${var-name, default-value}：この構文は、変数がプロパティ階層で見つからない場合にデフォルト値を提供します。

Correct Macro Variables Usage および Incorrect Macro Variables Usage の例では、オプション 11 の正しい使用法と誤ったの使用法を示しています。

正しいマクロ変数の使用法

```
# 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.docsDevNmAccessControl.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}"
```

間違ったマクロ変数の使用法

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

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

SNMP TLV

プライム ケーブル プロビジョニングは、オプション 11 および 64 を使用して、動的テンプレート ファイルの SNMP TLV をサポートします。

- DOCSIS：ケーブル バージョン 2,0 以降のプライム ケーブル プロビジョニングから。
- PacketCable：プライム ケーブル プロビジョニング バージョン 2.5 以降から。
- CableHome：プライム ケーブル プロビジョニング バージョン 2.6 以降。

これらのテンプレート ファイルの SNMP TLV 構文を検証するには、プライム ケーブル プロビジョニングは SNMP TLV で参照されるう対応する SNMP OID を含む MIB ファイルを要求します。テンプレートに MIB にない SNMP OID を持つ SNMP TLV が含まれる場合、SNMP TLV は構文エラーを生成します。

次のセクションでは、MIB がない状態や、ベンダー固有の MIB がある状態で SNMP TLV を追加する方法について説明します。

MIB を使用しない SNMP TLV の追加

RDU によって MIB がロードされる必要なく、動的設定ファイル（DOCSIS、PacketCable、CableHome）で SNMP TLVs を追加することができます。RDU 設定の拡張機能内で、次の方法で DOCSISOptionFactory インターフェイスの機能にアクセスできます。

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

tuple: OID、タイプ、値 を含む optionValueList とともに、.パブリック optionValueList 列挙値は上記の方法で使用可能です。

RDU 動的設定テンプレートから次の構文を使用して、RDU MIB に対して検証されない SNMP TLVs を指定します。

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

例：

```
# 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:'
```

以下の表は、許可される SNMP 変数タイプ名を説明します。

表 *2 : SNMP* 変数タイプ

IETF 標準 SMI データ タイプ	SNMP API 名
Integer32	INTEGER
Integer（列挙）	INTEGER
Unsigned32	UNSIGNED32
Gauge32	GAUGE
Counter32	COUNTER
Counter64	COUNTER64
Timeticks	Timeticks
OCTET STRING	STRING
OBJECT IDENTIFIER	OBJID
IpAddress	IPADDRESS
BITS	STRING

たとえば、SMI Integer32 タイプを指定するため、次のタイプが承認されます（大文字小文字に関係なく）：Integer32、INTEGER。

OCTET STRING タイプについては、次のタイプすべてが承認サされます：OCTET STRING、OCTETSTRING、STRING。

カスタム SNMP TLV テンプレート オプションは、RDU MIB に存在するものを含め、任意の SNMP TLV を指定するために使用できます。カスタム SNMP TLV エラーのチェックは厳密性が低く、不正なスカラー/カラムの参照を検出しません（たとえば、OID 名で .9 に対して .n）。

ベンダー固有の MIB を持つ SNMP TLV の追加

RDU へ MIB を追加することで、SNMP TLV 値で使用されるマクロ変数を許可しながら、テンプレートが人が読める形式の SNMP OID を使用できます。

使用する SNMP OID に対応する MIB がある場合、MIB ファイルをプライム ケーブル プロビジョニング RDU に追加できます。MIB を追加後、新しい MIB で参照されている SNMP OID を使用する SNMP TLV が認識されます。

プライム ケーブル プロビジョニング RDU に新しい MIB を追加するには。

ステップ 1 プライム ケーブル プロビジョニング管理者ユーザー インターフェイスを起動します。

ステップ 2 ナビゲーション バーで、**[Configuration（設定）] > [Defaults（デフォルト）]** をクリックします。

ステップ 3 表示される [Configure Defaults（デフォルトの設定）] ページで、左ペインにある [System Defaults（システム デフォルト）] リンクをクリックします。

ステップ 4 [MIB List（MIB リスト）] フィールドの最後に新しい MIB の内容を貼り付けます。

ステップ 5 [Submit] をクリックします。`

MIB のロード順序のデバッグ

通常ベンダーは、内部 MIB の依存関係を満たすため、特定のロード順序を必要とするいくつかの MIB を提供します。ベンダーでは頻繁に適切なロード順序が使用されていないため、適切なロード順序をユーザーが決定する必要があります。このセクションでは、プライム ケーブル プロビジョニング デバッグ情報を使用して、MIB ロード順序の問題を解決する方法について説明します。

（注） プライム ケーブル プロビジョニングの MIB ロード順序は、*/snmp/mibs/MibList* プロパティで一覧表示されている MIB の順序によって設定されます。

RunCfgUtil.sh ツールを使用して、*api.properties* ファイルで指定されたプロパティの適切なロード順序を決定できます。runCfgUtil.sh ツールは、 *BPR_HOME/rdu/bin* ディレクトリに存在します。

ステップ 1 この手順で説明されている内容と同様の設定内容を使用して、*api.properties* ファイルを設定します。*Api.properties* ファイルによって、プライム ケーブル プロビジョニング トレーシングがユーザーのコンソールに MIB デバッグ方法をリダイレクトできます。

```
#
# Enable logging to the console
#
/server/log/1/level=Info
/server/log/1/properties=level
/server/log/1/service=com.cisco.csrc.logging.SystemLogService
/server/log/1/name=Console
#
# Enable trace categories
#
/server/log/trace/rduserver/enable=enabled
#
# The list of MIBs to be added.
#
/snmp/mibs/MibList=arrishdr.mib,arris_cm_capability.mib,arris_mta_device.mib,arris_sip.mib,arris_cm.mib,
pp.mib,blp2.mib,dev0.mib,docs_evnt.mib,qos.mib,test.mib,usb.mib,snmpv2_conf.mib,rfc1493.mib,rfc1907.mib,
rfc2011.mib,rfc2013.mib,rfc2233.mib,rfc2571.mib,rfc2572.mib,rfc2573.mib,rfc2574.mib,rfc2575.mib,rfc2576
.mib,rfc2665.mib,rfc2669.mib,rfc2670.mib,rfc2786.mib,rfc2851.mib,rfc2933.mib,rfc 3083.mib
```

ステップ 2 RunCfgUtil.sh が十分に設定されている状態で、オプション 11 またはオプション 64（SNMP エンコーディング）が含まれている任意のテンプレートをエンコードするツールを実行します。ツールが /snmp/mibs/MibList 内で指定した MIB のロードを試行し、MIB ロードエラーとともに完全なデバッグ情報をユーザー コンソールにリダイレクトします。

ステップ 3 エラーが発生せず完全なMIBのセットがロードされ、ファイルが正常にエンコードされるまで、エラー情報を使用して /snmp/mibs/MibList 内で指定した MIB 順序を変更します。

ステップ 4 正常なロード順序を決定したら、使用しているプライム ケーブル プロビジョニング バージョンに基づき、この手順で説明されている手順を完了します。

1. 管理者ユーザー インターフェイスから、**[Configuration（設定）] > [Defaults（デフォルト）]** をクリックして [System Defaults（システム デフォルト）] リンクをクリックします。
2. MIB リスト フィールドでロード順序の情報をコピーします。

   RDU は、ベンダー提供の MIB を使用してテンプレートをエンコードするように設定されています。

（注） RDU を再起動する必要はありません。

Api.properties ファイルと MIB リスト フィールドで、 */snmp/mibs/mibList* 文字列を使用していることを確認します。

定義されたオプションのエンコーディング タイプ

次の表では、定義済みのエンコーディング タイプのオプションを特定します。

表 *3:* 定義されたオプションのエンコーディング タイプ

エンコーディング	入力	例
認証アクション	符号なし 8 ビット整数または説明文字列です。 値は許可を許可するための、値は次のとおりです。 • 0 • permit 認証を拒否するには、値は次のとおりです。 • 1 • deny	0 1 permit deny

エンコーディング	入力	例
ビューのアクセス制御	符号なしの 8 ビット整数または文字列。 アクセス ビューから SNMPv3 アクセス ビュー サブツリーを含めるための値は、次のとおりです。 • 1 • included アクセス ビューから SNMPv3 アクセス ビュー サブツリーを除外するための値は、次のとおりです。 • 2 • excluded	1 2 included excluded
アクセス ビューのタイプ	符号なしの 8 ビット整数または文字列。 読み取り専用アクセスを有効にするための値は、次のとおりです。 • 1 • 読み取り専用 読み書きアクセスを有効にするための値は、次のとおりです。 • 2 • 読み取りと書き込み	1 2 Read-only Read-write

エンコーディング	入力	例
ActInact	符号なしの 8 ビット整数または文字列。 TLV を無効にするための値は、次のとおりです。 • 0 • Inactive 値は、TLV を有効にするための値は、次のとおりです。 • 0 • アクティブ	0 1 Inactive Active
BitFlag8	符号なし 8 ビット整数出力は、値の 16 進数の文字列表現です。	0xFE
BitFlag32	符号なし 32 ビット整数出力は、値の 16 進数の文字列表現です。	0xFFFF0000
ブール	0 は false true の場合は 1 です。	0 1
Byte16	16 バイトは 32 文字の 16 進数の文字列として指定します。一般的に、ケーブル モデムと CMTS の MIC オプションを表すために使用されます。0x プレフィックスは使用できません。	なし。 プライム ケーブル プロビジョニング 自動的に、ケーブル モデムと CMTS MIC オプションのハッシュを計算します。
バイト	16 進数オクテットのシリーズ。各オクテットは、2 文字にする必要があります。	000102030405060708

エンコーディング	入力	例
CPE アクセス制御	符号なしの 8 ビット整数または文字列。 デバイス アクセス制御を無効にするための値は、次のとおりです。 • 0 • ディセーブル デバイス アクセス制御を有効にするための値は、次のとおりです。 • 1 • イネーブル	0 1 Disabled Enabled
DSCClassifier	8 ビットの符号なし整数または DSCClassifier 文字列の名前。符号なし整数は、次のとおりです。 • 0: DSC 分類子の追加 • 1: DSC 置換分類子 • 2: DSC 削除分類子	0
EnableDisable	符号なしの 8 ビット整数または文字列。 値は、無効にするのには。 • 0 • ディセーブル 有効にするための値は、次のとおりです。 • 1 • イネーブル	0 1 Disabled Enabled

エンコーディング	入力	例
Erouter Init モード	eRouter 初期化モードを説明する8 ビットの符号なしの整数 0 = 無効 1: IPv4 プロトコルの有効化 2:IPv6 プロトコルの有効化 3: デュアル IP プロトコルの有効化	0 1 2 3 無効化 IPv4 IPv6 デュアル
eRouter 初期化モードのオーバーライド	eRouter 初期化モードをオーバーライドするための符号なし 8 ビットの整数 1 = eRouter 初期化モード TLV を無視し、eRouter を無効のまま保持する 0 = eRouter 初期化をフォロー	0 1 eRouter 初期化をフォロー - モード eRouter - 無効化
Inet アドレス ピア	1 バイト InetAddressTypeCode: • IPv4 の 1 • IPv6 の 2 この値の後に、IPv4 または IPv6 インターネット アドレスが続きます。 その結果、この長さは、IPv4 の場合は 5 バイト (1 + 4) 、IPv6 の場合は 17 バイト (1 + 16) になります。	`1,10.112.125.111` `2,0:0:0:0:0:ffff:8190:3426`
IP アドレス	4 つ符号なし整数 8、ドット (.) 区切り。	`10.10.10.1`
IPv6 アドレス	x:x:x:x:x:x:x:x, where the x*:x:x:x:x:x:x:x,* という形式の IPv6 アドレス。*x* はアドレスの 8 つの 16 ビット部分の 1 ～ 4 桁の 16 進数を表します。	`2001:db8:0:0:8:800:200c:417a`

エンコーディング	入力	例
IPv4 アドレスまたは IPv6 アドレス	IPv4 または IPv6 アドレスの文字列表現。	`10.112.125.111` `0:0:0:0:0:ffff:8190:3426`

IP Mode	符号なしの 8 ビット整数または文字列。
	IPv4 モードでは、値は次のとおりです。
	• 0
	• IPv4
	IPv6 モードの値では、値は次のとおりです。
	• 1
	• IPv6
		`0`
		`1`
		`IPv4`
		`IPv6`

複数の IP アドレス	IP アドレスのカンマ区切りリスト。	`10.11.12.13,10.11.12.14`
複数の IPv6 アドレス	IPv6 アドレスのカンマ区切りリスト。	`2001:db8:0:0:8:800:200c:417a,ff01:0:0:0:0:0:0:101`
MAC アドレス	6 つの 16 進数オクテット、コロン (:) またはハイフン (-) 区切り。各オクテットは、ちょうど 2 文字にする必要があります。コロン、ハイフンを混在させる必要があります。	`00:01:02:03:04:05` `00-01-02-03-04-05`
MAC アドレスとマスク	12 個のオクテット、コロン (:) またはハイフン (-) 区切り。各オクテットは、2 文字にする必要があります。コロン、ハイフンを混在させる必要があります。最初の 6 個のオクテットは、MAC アドレスを表し。最後の 6 個のオクテットは、MAC アドレスのマスクを表します。	`00:01:02:03:04:05:06:07:08:09:0A:0B` `00-01-02-03-04-05-06-07-08-09-0A-0B`
NoLV	タイプのみ。値または長さは含まれません。	`null`

エンコーディング	入力	例
NVTASCII	ASCII 文字列。エンコードされた文字列では、NULL で終了できません。	`This is an ASCII string`
OID	SNMP OID 文字列。	`sysinfo.0`
OIDCF	カンマ区切りされた SNMP OID 文字列と符号なしの整数 (0 または 1) 。	`sysinfo.0,1`

オン/オフ	符号なし 8 ビット整数
	TLV を切り替えるための値は、次のとおりです。
	• 0
	• On
	TLV をオフに切り替えるための値は、次のとおりです。
	• 1
	• 消灯
		`0`
		`1`
		`On`
		`Off`

OUI	3 つの 16 進数オクテット、コロン (:) またはハイフン (-) 区切り。各オクテットは、2 文字にする必要があります。	`00-00-0C`

RFC868Time	RFC868 時間を示す符号なしの 32 ビット整数。出力は、*MM/dd/yyyy HH:mm:ss*の形式を使用する日時文字列です。。	`0`
		(「12/31/1899 19:00:00」を表す)
		`4294967295`
		(「02/07/2036 01:28:15」を表す)

エンコーディング	入力	例
ServiceFlow	8 ビットの符号なし整数またはサービス フロー説明文字列。出力は次を示すサービス フローです。 • 0：予約済 • 1: 未定義 (CMTS の実装に依存) • 2—ベスト エフォート • 3—非リアルタイム ポーリング サービス • 4—リアルタイム ポーリング サービス • 5—アクティビティ検出による非送信請求許可サービス • 6—非送信請求許可サービス	0

エンコーディング	入力	例
SNMPVarBind	SNMP OID ストリング、タイプ、および値。これらそれぞれはカンマ区切りです。有効なタイプは、次のとおりです。 • BITS • カウンタ • Counter32 • Counter64 • ゲージ • Gauge32 • INTEGER • Integer32 • IpAddress • OCTETSTRING • OBJECTIDENTIFIER • Opaque • TimeTicks • Unsigned32 （注） OCTETSTRINGは、後に続くNULL（たとえば、オクテット文字列）なしで16進数表記に変換される文字列か、単一引用符（「aa:bb:cc」など）内に含まれる16進数表記の文字列のいずれかを使用できます。	`.experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.docsDevNmAccessStatus.1, INTEGER, 4`

エンコーディング	入力	例
SrvChangeAct	0 ~ 3 の範囲に制限される 8 ビットの符号なし整数または SrvChangeAct の記述。記述の文字列の出力は次のとおりです。 • 0: PHS ルールの追加 • 1: PHS ルールの設定 • 2: PHS ルールの削除 • 3: すべてのコール ルールの削除	`0`
Subtype	1 個または 2 個のカンマで区切られた符号なし整数 8。	`12` `12,14`
トポロジ モードのエンコーディング	演算子指定 IPv6 プレフィックスを分割するために使用される 8 ビットの符号なし整数 1: 望ましい深さ 2: 望ましい幅	1 2 奥行 幅

トランスポート アドレスとマスク	Ipv4 の場合、カンマ (,) で区切られた、ポート番号が後に続くドット付き表記で 4 個のオクテットの IP アドレス。
	IPv6 の場合、ドット付き表記または文字列。
	• カンマ (,) で区切られた、ポート番号が後に続くドット付き表記で有効な IPv6 アドレス。
	• カンマ (,) で区切られた、ポート番号が後に続く IPv6 アドレスの文字列表現。例: *x:x:x:x:x:x:x:x,* という形式の IPv6 アドレス。*x* はアドレスの 8 つの 16 ビット部分の 1 ～ 4 桁の 16 進数を表します。
		IPv4
		`10.112.125.111,5678`
		IPv6
		`2001.db8.0.0.8.800.200c.417a,5678`
		`2001:db8:0:0:8:800:200c:417a,5678`

エンコーディング	入力	例
符号なし整数 8	0 ～ 255	`14`
符号なし整数 16	0 ～ 65535	`1244`
符号なし 32 ビット整数	0 ～ 4294967295	`3455335`
符号なし整数 8 と符号なし整数 16	カンマで区切られた 1 個の符号なし整数 8 と 1 個の符号なし整数。	`3,12324`
符号なし整数 8 ペア	カンマで区切られた 2 個の符号なし整数 8。	`1,3`
符号なし整数 8 トリプレット	カンマで区切られた 3 個の符号なし整数 8。	`1,2,3`
検証	符号なし 8 ビット整数 値は検証を有効にするための値は、次のとおりです。 • 0 • 検証 検証を無効にするための値は、次のとおりです。 • 1 • 次のことを検証しないでください。 （注） Verify TLV のための定義の真偽が、DOCSIS 1.1 仕様 (オプション 26.11) に沿っていること。	`0 = verify` `1 = don’t verify`
ZTASCII	ASCII 文字列。エンコードされた文字列の末尾は NULL となります。	`This is an ASCII string`

BITS 値のシンタックス

BITS タイプを使用するとき、ラベル（「interval1 interval2 interval3」）または数字のビット位置（「0 1 2」）のいずれかを指定する必要があります。ラベルの値は 1 ベースであり、ビット値は、0 ベースであることに注意してください。

これは、ビット数字を使用するシンタックスを示します。

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

これは、ラベルを使用する顧客のオクテット文字列（FFFE000000000000）のシンタックスを示します。

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

OCTETSTRING シンタックス

OCTETSTRING は、末尾 NULL（たとえば、オクテット文字列）なしで 16 進数表記に変換される文字列か、単一引用符（「aa:bb:cc」など）内に含まれる 16 進数表記の文字列のいずれかを使用できます。

テンプレートの設定ファイルユーティリティの使用

PacketCable 1.0/1.5/2.0、DOCSIS 1.0/1.1/2.0/3.0/3.1、CableHome テンプレート、設定ファイルをテスト、検証、表示するため、設定ファイルユーティリティを使用します。これらのアクティビティは、個別の設定ファイルの導入を成功させるために重要です。テンプレートの詳細については、テンプレート ファイル：概要 （23 ページ） を参照してください。

設定ファイル ユーティリティは、RDU がインストールされている場合にのみ使用可能です。ユーティリティは *BPR_HOME/rdu/bin* ディレクトリにインストールされます。

エンコードされているテンプレート ファイルと復号化されているバイナリ ファイルの両方が、設定ファイル ユーティリティが呼び出されるディレクトリに存在する必要があります。

このセクションのすべての例では、RDU が動作しており、次の条件が適用されるものとします。

- プライム ケーブル プロビジョニング アプリケーションは、デフォルトのホーム ディレクトリ（*/opt/CSCObac*）にインストールされます。
- RDU ログイン名は、 **admin** です。
- RDU ログイン パスワードは **changeme** です。

（注） このセクションの例の一部では、結果の例に影響がない場合省略された情報が削除されています。これが発生した場合、例サマリのの前に、省略記号（...）で確認できます。

このセクションでは、次のトピックについて説明します。

設定ファイル ユーティリティの実行

後述の手順と例では、「run the configuration file utility」のフレーズは指定されたディレクトリから **runCfgUtil.sh** コマンドを入力することを意味します。設定ファイル ユーティリティを実行するには、*BPR_HOME/rdu/bin* ディレクトリから次のコマンドを実行します。

runCfgUtil.sh *options*

利用可能なオプションは下記の通りです。

- **-c** *shared*：DOCSIS テンプレート ファイルを解析する場合、CMTS 共有秘密鍵を指定します。デフォルトの共有秘密鍵を指定するには、**c cisco** と入力します。
- **-cablehome**：CableHome ポータル サービス設定ファイルとして入力ファイルを識別します。**-docsis** または **-pkt** オプションではこれを使用しないでください。
- **-d**：バイナリ入力ファイルをデコードします。**-e**オプションではこれを使用しないでください。
- **-docsis**：DOCSIS 設定ファイルとして入力ファイルを指定します。**-pkt** オプションではこのデフォルトを使用しないでください。
- **v***version*：使用されている DOCSIS バージョンを指定します。たとえば、DOCSIS 1.1 を使用している場合は、**-v 1.1** を入力します。バージョン番号を指定しない場合、コマンドは

DOCSIS のサポートされている最新バージョンを使用することがデフォルトになります。プライム ケーブル プロビジョニングがサポートするのは、1.0、1.1、2.0、3.0、3.1 です。

- **-e**: テンプレート入力ファイルにエンコードします。**-d** オプションではこのデフォルトを使用しないでください。
- **-g**：DOCSIS、PacketCable、または CableHome バイナリ ファイルのいずれかからテンプレート ファイルを生成します。
- **-h** *host:port*：ホストとポートを指定します。デフォルトのポート番号は 49187 です。
- **-i** *device-id*：テンプレートの解析中にマクロ変数を置換する場合、使用するデバイスを特定します。たとえば、デバイスの MAC アドレスが、1.6、00:00:00:00:00:01 の場合 **-i 1,6,00:00:00:00:00:01** と入力するか、デバイス DUID が 00:03:00:01:00:18:68:52:75:c0 の場合、**-i 00:03:00:01:00:18:68:52:75:c0** と入力します。このオプションを使用するとき、ユーザー名とパスワードを指定するために **-u** および **-p** オプションをそれぞれ使用する必要があります。**-m** オプションでこれを使用しないでください。
- **-l** *filename*：ローカル ファイル システム上にある入力ファイルを特定します。たとえば、入力ファイルは *any_file* と呼ばれ、**-l any_file** と入力します。**-r** オプションでこれを使用しないでください。
- **-loc** ロケール：na、euro、ietf など PacketCable ロケールを指定します（デフォルトでは na）。デフォルトは na です。MTA が欧州仕様 MTA の場合、ロケールは euro に設定する必要があります。
- **-m** *macro*：マクロ変数のキー値のペアを指定します。形式はキー=値です。複数のマクロ変数を必要とする場合は、キー値のペアの間にコンマを 2 個使用します。たとえば、key_1 = value_1,,key_2 = value_2。**-i** オプションでこれを使用しないでください。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。たとえば、パスワードが 123456 の場合 **-p 123456** を入力します。
- **-o** *filename*：バイナリ ファイルとして解析されたテンプレート ファイルを保存します。たとえば、*op_file* と呼ばれるファイルで出力を検出する場合は、**-o op_file** を入力します。
- **-pkt**：PacketCable MTA 設定ファイルとして入力ファイルを特定します。**-docsis** オプションではこれを使用しないでください。
- **-r** *filename*：RDU に追加されているリモート ファイルとして入力ファイルを特定します。たとえば、ファイルが *file25* と呼ばれる場合、**-r file25** と入力します。このオプションを使用するとき、ユーザー名とパスワードを指定するために **-u** および **-p** オプションをそれぞれ使用する必要があります。**-l** オプションでこれを使用しないでください。
- **-s**：解析されたテンプレートまたはバイナリ ファイルの内容を人が読める形式で表示します。
- **-t**：PacketCable エンコーディング タイプを指定します。**Secure** または **Basic**（デフォルトは Secure です）。

- **-u***username*：RDU に接続するときに使用するユーザー名を指定します。たとえば、ユーザー名が admin の場合は、 **-u admin** と入力します。
- **-E**：拡張 CMTS MIC（EMIC）計算を有効にして、EMIC 計算のデフォルトのオプションを特定します。デフォルトのオプションは次の通りです。
  - HMAC タイプ：MMH16
  - EMIC ダイジェスト タイプ：明示
  - **cisco** としての EMIC 共有秘密鍵。
- **-Ei**：EMIC 計算の暗黙として EMIC ダイジェスト タイプを特定します。
- **-Eh**：HMAC タイプを指定します。MD5 または MMH16（デフォルトは MMH16）。
- **-Es** *secret*：DOCSIS テンプレート ファイルを解析するときに、EMIC 共有秘密鍵を指定します。

（注） 設定ファイル ユーティリティでは、テンプレート ファイルにオプション 19（TFTP サーバ タイムスタンプ）およびオプション 20（TFTP プロビジョニング サーバ モデムアドレス）は含まれません。しかし、プライム ケーブル プロビジョニング TFTP 混合には含まれます。また、オプション 6（CM MIC）および 7（CMTS MIC）は両方ともエンコードされたテンプレート ファイルに自動的に挿入されます。したがって、これらの完全性チェック（MIC）を指定する必要はありません。

プライム ケーブル プロビジョニングにテンプレートを追加する

プライム ケーブル プロビジョニング テンプレートをテストするために設定ファイルユーティリティを使用するには：

ステップ 1 テンプレート ファイル：概要 （23 ページ） で説明されているようにテンプレートを開発します。テンプレートに他のテンプレートが含まれている場合は、すべての参照されているテンプレートが同じディレクトリに存在することを確認します。

ステップ 2 ローカル ファイル システムで設定ファイル ユーティリティを実行します。テンプレートの構文を確認するか、設定ファイルユーティリティ プロセスが CRS として出力を返すようにテンプレートを処理できます。

テンプレートにマクロ変数が含まれている場合は、指定された順序でこれらの操作を行います。

a) コマンド ライン代入をテストします。
b) RDU に追加されたデバイスでテストします。

ステップ 3 テンプレート（使用されている含まれたテンプレート）を RDU に追加します。

ステップ 4 設定ファイルユーティリティを実行して、ファイルを解析します。『Testing Template Processing for an External Template File』を参照してください。

テンプレートにマクロ変数が含まれている場合は、指定された順序でこれらの操作を行います。

a) コマンドライン代入をテストします。
b) RDU に追加されたデバイスでテストします。

ステップ 5 すべてのテストが成功した後、テンプレートを使用するサービス クラスを設定します。

バイナリ ファイルをテンプレート ファイルに変換する

runCfgUtil.sh コマンドを使用して、テンプレート ファイルにバイナリ設定メモリ ファイルを変換します。プライム ケーブル プロビジョニング 動的設定の生成は、作成されたテンプレートに基づくものです。既存のテスト済みバイナリ ファイルをテンプレート ファイルに自動的に変換すると、プロセスを高速化し、エラーが発生する可能性を軽減します。

構文の説明

runCfgUtil.sh -g -l*binary_file* **-o** *template_file*

- **-g**：入力バイナリ ファイルから生成する必要があるテンプレートファイルを指定します。
- **-l** *binary_file*：パス名を含むローカル入力ファイルを指定します。すべてのケースで、入力バイナリ ファイル名には *.cm* ファイル拡張子があります。例：*bronze.cm*。
- **-o** *template_file*：パス名を含む、出力テンプレート ファイルを指定します。すべてのケースで、出力テンプレートファイル名には *.tmpl*ファイル拡張子があります。例：、*test.tmpl*。

バイナリ ファイルをテンプレート ファイルに変換するには：

ステップ 1 ディレクトリを */opt/CSCObac/rdu/samples/docsis* に変更します。

ステップ 2 使用するテンプレート ファイルを選択します。この例では、*unprov.cm* と呼ばれる既存のバイナリ ファイルを使用します。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin # **runCfgUtil.sh -g-l unprov.cm -o test.tmpl - docsis**

-docsis：DOCSIS 設定ファイルになるように入力ファイルを指定します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

#################################################################
## Template File Generator
## Generated on Fri Oct 12 16:12:51 EST 2007
#################################################################

#################################################################
## Each generated option will be represented by the following:
## The first line will represent a description of the
## generated option
## The second line will represent the generated option
```

```
## The third line will represent the custom version
## of the generated option
################################################################

# (3) Network Access Control
Option 3  01
# Option 3  hex 01

# (4.1) Class ID
Option 4.1 1
# Option 4.1 hex 01

# (4.2) Maximum Downstream Rate
Option 4.2 128000
# Option 4.2 hex 0001F400

# (4.3) Maximum Upstream Rate
Option 4.3 64000
# Option 4.3 hex 0000FA00

# (4.4) Upstream Channel Priority
Option 4.4 1
# Option 4.4 hex 01

# (4.5) Guaranteed Minimum Upstream Channel Data Rate
Option 4.5 0
# Option 4.5 hex 00000000

# (4.6) Maximum Upstream Channel Transmit Burst
Option 4.6 1600
# Option 4.6 hex 0640

# (4.7) Class-of-Service Privacy Enable
Option 4.7 00
# Option 4.7 hex 00

# (11) SNMP MIB Object
Option 11
.iso.org.dod.internet.experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.
docsDevNmAccessStatus.1,INTEGER,createAndGo
# Option 11  hex 3082000F060A2B060103530102010701020104

...

# (18) Maximum Number of CPEs
Option 18  1
# Option 18  hex 01
```

ローカル テンプレート ファイルのテンプレート処理のテスト

ローカルファイルシステムに保存されているテンプレートファイルの処理をテストするには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -pkt -l *file*

- **-pkt**：PacketCable MTA ファイルとして入力ファイルを特定します。
- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。

• *file*：解析されない入力テンプレート ファイルを識別します。

ローカル ファイル システム上にあるテンプレート ファイルを解析するには：

ステップ 1 */opt/CSCObac/rdu/samples/packet_cable* にディレクトリを変更します。

ステップ 2 使用するテンプレート ファイルを選択します。この例では、*unprov_packet_cable.tmpl* と呼ばれる既存のテンプレート ファイルを使用します。これが PacketCable MTA テンプレートであるため、**-pkt**オプションが使用されます。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin/**runCfgUtil.sh -pkt -l unprov_packet_cable.tmpl**

unprov_packet_cable.tmpl：解析する入力テンプレート ファイルを特定します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off                File Bytes          Option              Description         Value

0                  FE0101              254                 Telephony Config    1
                                                           File Start/End

3                  0B153013060E        11                  SNMP MIB Object     .iso.org.dod.internet
                    2B06010401A                                                .private.enterprises
                   30B0202010101                                               .cableLabs.clabProject
                    0700020102                                                 .clabProjPacketCable
                                                                               .pktcMtaMib.pktcMtaMibObjects

                                                                               .pktcMtaDevBase.
                                                                               pktcMtaDevEnabled.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.
```

Testing Template Processing for an External Template File

外部テンプレート ファイルの処理をテストするには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -docsis -r *file* **-u** *username* **-p** *password*

• **-r**：RDU に追加されているファイルとして入力ファイルを識別します。

- *file*：解析されない入力テンプレート ファイルを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*：RDU に接続するときに使用するパスワードを指定します。
- **-docsis**：DOCSIS テンプレートとしてファイルを識別します。

RDU に追加されているテンプレート ファイルを解析するには。

ステップ 1 使用するテンプレート ファイルを選択します。この例では、 *unprov.tmpl* と呼ばれる既存のテンプレート ファイルを使用します。**-docsis** オプションは、DOCSIS テンプレートが使用されているため使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -docsis -r unprov.groovy -u admin -p changeme
```

- **unprov.tmpl**：入力ファイルを特定します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

（注） 次に示す結果は、図のみであり簡略化のためトリミングされています。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.1

Off             File Bytes      Option          Description     Value

0               030101          3               Network Access  On
                                                Control

3               041F            4               Class of Service

5               010101          4.1             Class ID        1

8               02040000FA00    4.2             Maximum         128000 bits/sec
                                                Downstream Rate

14              03040000FA00    4.3             Maximum Upstream 64000 bits/sec
                                                Rate

20              040101          4.4             Upstream Channel 1
                                                Priority

...

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

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

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.
```

ローカル テンプレート ファイルおよび 共有 秘密鍵 の追加のテンプレート処理のテスト

テンプレート ファイルの処理をテストし、指定する 共有秘密鍵を追加するには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -docsis -l *file* **-c** *shared*

- **-e**：エンコード オプションを識別します。
- **-docsis**：DOCSIS テンプレート ファイルとして入力ファイルを識別します。
- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。
- *file*：解析されない入力テンプレート ファイルを識別します。
- **-c**： DOCSIS テンプレート ファイルを解析する場合、CMTS 共有秘密鍵を指定します。
- *shared*—新しい共有秘密鍵を指定します。デフォルトの共有秘密鍵は、**cisco** です。

ローカルに保存されているテンプレートファイルを解析するには、ユーザー指定の共有秘密鍵のを設定します。

ステップ 1 */Opt/CSCObac/rdu/templates*にディレクトリを変更します。

ステップ 2 解析するテンプレート ファイルを選択します。この例では、 *unprov.tmpl* と呼ばれる既存のテンプレート ファイルを使用します。**-docsis** オプションは、DOCSIS テンプレートであるため使用されます。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

/opt/CSCObac/rdu/bin/**runCfgUtil.sh -e -docsis -l unprov.tmpl -c shared**

- **unprov.tmpl**: ローカル ファイル システムの入力ファイルを特定します。
- **shared**：新しい共有秘密鍵を特定します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.1
```

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 C9152B44DB95 5420843EF6FE	6	CM MIC Configuration Setting	8506547FC9152B44 DB955420843EF6FE
270	0710644B675B 70B7BD3E09AC 210F794A1E8F	7	CMTS MIC Configuration Setting	644B675B70B7BD3E 09AC210F794A1E8F
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.
```

ローカル テンプレート ファイルおよび EMIC 共有 秘密鍵 の追加のテンプレート処理のテスト

テンプレート ファイルの処理をテストし、指定する EMIC 共有秘密鍵を追加するには、**runCfgUtil.sh** コマンドを使用します。

例 1：

この例では、EMIC を有効にする **runCfgUtil**.sh コマンドを使用する方法について説明します。

- HMAC タイプとしての MMH16。
- EMIC ダイジェストの明示的なオプションです。

- EMIC 計算の EMIC 共有秘密鍵としての cisco。

構文の説明

runCfgUtil.sh -E -docsis -l *filename*

- **-E**：EMIC 計算を有効にします。
- **-docsis**：DOCSIS テンプレート ファイルとして入力ファイルを識別します。
- **-l***filename*：パス名を含む、入力テンプレート ファイルを指定します。すべてのケースで、入力テンプレート ファイルには .tmpl ファイル拡張子があります。例：test.tmpl。

デフォルト設定で EMIC 計算を実行するには。

ステップ 1 使用するテンプレート ファイルを選択します。この例では、 *unprov.tmpl* と呼ばれる既存のテンプレート ファイルを使用します。**-docsis** オプションは、DOCSIS テンプレートが使用されているため使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin/runCfgUtil.sh -E -l test.tmpl
```

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.1
```

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	0204001F4000	4.2	Maximum Downstream Rate	2048000 bits/sec
14	03040007D000	4.3	Maximum Upstream Rate	512000 bits/sec
20	040106	4.4	Upstream Channel Priority	6
23	050400000000	4.5	Guaranteed Minimum Upstream Channel Data Rate	0 bits/sec

29	06020640	4.6	Maximum Upstream Channel Transmit Burst	1600 bytes
33	070100	4.7	Class-of-Service Privacy Enable	Disabled
249	120103	18	Maximum Number of CPEs	3
252	2B1C	43	DOCSIS Extension Field	
254	0803FFFFFF	43.8	Vendor ID	FF-FF-FF
259	0615	43.6	Extended CMTS MIC Configuration Setting	
261	010102	43.6.1	Extended CMTS MIC HMAC type	2
264	020678007BEC 1C80	43.6.2	Extended CMTS MIC Bitmap	78007BEC1C80
272	03081605487D 3D7A9403	43.6.3	Explicit Extended CMTS MIC Digest Subtype	1605487D3D7A9403
282	06108BFC8016 39BE8D7F396E E49D402832FC	6	CM MIC Configuration Setting	8BFC801639BE8D7F 396EE49D402832FC
300	071053F94674 11C355EF01D0 104995AB9797	7	CMTS MIC Configuration Setting	53F9467411C355EF 01D0104995AB9797
318	FF	255	End-of-Data Marker	
319	00	0	PAD	

例 2：

この例では、EMIC を有効にして、デフォルトの設定を変更する **runCfgUtil.sh** コマンドを使用する方法について説明します。

構文の説明

runCfgUtil.sh -E -Ei -Eh *MD5* **-Es** *secret* **-l** *filename*

- **-E**：EMIC 計算を有効にします。
- **-Ei**：EMIC 計算の暗黙として EMIC ダイジェスト タイプを指定します。
- **-l***filename*：パス名を含む、入力テンプレート ファイルを指定します。すべてのケースで、入力テンプレート ファイルには .tmpl ファイル拡張子があります。例：test.tmpl。
- **-Eh**：HMAC タイプを指定します。MD5 または MMH16（デフォルトは MMH16）。
- **-Es** *secret*：DOCSIS テンプレート ファイルを解析するときに、EMIC 共有秘密鍵を指定します。

デフォルト設定として含まれていないオプションを使用して、EMIC の計算を実行します。

ステップ 1 使用するテンプレート ファイルを選択します。この例では、 *unprov.tmpl* と呼ばれる既存のテンプレート ファイルを使用します。**-docsis** オプションは、DOCSIS テンプレートが使用されているため使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin/runCfgUtil.sh -E -Ei -Eh MD5 -Es secret -l test.tmpl
```

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.1
```

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	0204001F4000	4.2	Maximum Downstream Rate	2048000 bits/sec
14	03040007D000	4.3	Maximum Upstream Rate	512000 bits/sec
20	040106	4.4	Upstream Channel Priority	6
23	050400000000	4.5	Guaranteed Minimum Upstream ChannelData Rate	0 bits/sec

29	06020640	4.6	Maximum Upstream Channel Transmit Burst	1600 bytes
33	070100	4.7	Class-of-Service Privacy Enable	Disabled
249	120103	18	Maximum Number of CPEs	3
252	2B12	43	DOCSIS Extension Field	
254	0803FFFFFF	43.8	Vendor ID	FF-FF-FF
259	060B	43.6	Extended CMTS MIC Configuration Setting	
261	010101	43.6.1	Extended CMTS MIC HMAC type	1
264	020678007BEC 1C80	43.6.2	Extended CMTS MIC Bitmap	78007BEC1C80
272	06107E6CF875 32C551791CCF 34770C94FA03	6	CM MIC Configuration Setting	7E6CF87532C55179 1CCF34770C94FA03
290	0710C9F8007A 40E4913779D3 C1C53599141B	7	CMTS MIC Configuration Setting	C9F8007A40E49137 79D3C1C53599141B
308	FF	255	End-of-Data Marker	
309	00	0	PAD	
310	00	0	PAD	
311	00	0	PAD	

コマンド行のマクロ変数を指定する

マクロ変数を指定するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -l *file* **-m** *"macros"*

- **-e**：エンコード オプションを識別します。
- **-l**：入力ファイルがローカル ファイル システム上にあることを指定します。
- *file*：解析されない入力テンプレート ファイルを識別します。
- **-m**：テンプレートを解析するとき、代わりに使用するマクロ変数を指定します。
- *"macros"*：希望するマクロを識別します。複数のマクロ変数が必要な場合は、各マクロ間に 2 つのコンマ区切り文字を挿入します。

コマンド行でマクロ変数の値を指定します。

ステップ 1 */opt/CSCObac/rdu/templates* にディレクトリを変更します。

ステップ 2 使用するテンプレート ファイルを選択します。

ステップ 3 テンプレートでマクロ変数を識別します。この例では、マクロ変数はマクロ 1（オプション 3）およびマクロ 11（オプション 4.2）です。

ステップ 4 マクロ変数の値を識別します。マクロ 1 の値は 1 に、マクロ 11 の値は 64000 に設定されています。

ステップ 5 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -l macro.tmpl -m "macro1=1,,macro11=64000"
```

- **macro.tmpl**：入力ファイルを識別します。
- **macro1=1,macro11=64000**：マクロ変数のキー値のペアを識別します。複数のマクロ変数が必要なため、2 つのコンマ区切り記号はキー値のペアの間で挿入されます。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off               File Bytes         Option             Description        Value

0                 030101             3                  Network Access     On
                                                        Control

3                 041F               4                  Class of Service

5                 010101             4.1                Class ID           1

8                 02040000FA00       4.2                Maximum Downstream 64000 bits/sec
                                                         Rate

14                03040000FA00       4.3                Maximum Upstream   64000 bits/sec
                                                        Rate
```

```
20                  040101              4.4                 Upstream Channel   1
                                                            Priority


...

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.
```

マクロ変数のデバイスの指定

マクロ変数のデバイスを指定するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**：エンコード オプションを識別します。
- **-r**：RDU に追加されているファイルとして入力ファイルを識別します。
- *file*：解析されない入力テンプレート ファイルを識別します。
- **-i**：マクロ変数を解析するときに使用するデバイスを指定します。
- *MAC*：そのデバイスの MAC アドレスを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。

マクロ変数の置換に使用するデバイスを指定するには：

ステップ 1 使用するテンプレート ファイルを選択します。この例では、既存のテンプレート ファイル *macro.tmpl* を使用します。

ステップ 2 テンプレートでマクロ変数を識別します。この例では、マクロ変数はマクロ 1（オプション 3）およびマクロ 11（オプション 4.2）です。

ステップ 3 使用するデバイスを識別します。この例では、デバイスが RDU に存在し、プロパティとしてマクロ変数が設定されていることを前提としています。マクロ 1 の値は 1 に、マクロ 11 の値は 64000 に設定されています。

ステップ 4 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin/runCfgUtil.sh -e -r macro.tmpl -i "1,6,00:01:02:03:04:05" -u admin -p changeme
```

- **macro.tmpl**：入力ファイルを識別します。
- **1,6,00:01:02:03:04:05**：デバイスの MAC アドレスを識別します。ここで使用される MAC アドレスは、例を示すことのみを目的としています。

- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off               File Bytes        Option             Description       Value

0                 030101            3                  Network Access    On
                                                       Control

3                 041F              4                  Class of Service

5                 010101            4.1                Class ID          1

8                 02040000FA00      4.2                Maximum Downstream 64000 bits/sec
                                                        Rate

14                03040000FA00      4.3                Maximum Upstream  64000 bits/sec
                                                       Rate

20                040101            4.4                Upstream Channel  1
                                                       Priority

...

0 error(s), 0 warning(s) detected. Parsing of macro.tmpl was successful.
The file macro.tmpl was parsed successfully in 159 ms.
The parser initialization time was 42 ms.
The parser parse time was 117 ms.
```

バイナリ ファイルへの出力を指定する

runCfgUtil.sh コマンドを使用して、バイナリ ファイルとして解析されたテンプレートの出力を指定します。

構文の説明

runCfgUtil.sh -l *input_file* **-o** *output_file*

- **-l**：入力ファイルをローカル ファイル システム上に置くように指定します。
- *input_file*: 解析されない入力テンプレート ファイルを特定します。
- **-o**：解析されたテンプレート ファイルを、バイナリ ファイルとして保存するように指定します。

- *output_file*：解析されたテンプレート ファイルのバイナリの内容が保存されているファイルの名前を特定します。

テンプレートからバイナリ ファイルへの解析の出力を指定します。

ステップ 1 */opt/CSCObac/rdu/templates* にディレクトリを変更します。

ステップ 2 使用するテンプレート ファイルを選択します。

ステップ 3 出力ファイル名を特定します。この例では、*unprov.cm* を使用しています。

ステップ 4 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -l unprov.tmpl -o unprov.cm
```

- **unprov.tmpl**：バイナリ ファイルに解析されている既存のテンプレート ファイルを特定します。
- **unprov.cm**：使用される出力ファイル名を特定します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0


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.
```

ローカル バイナリ ファイルの表示

ローカル システムに保存されているバイナリ ファイルを表示するには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -d -l *file*

- **-d**：コマンドが表示のためにバイナリ入力ファイルを解釈するように指定します。
- **-l**：入力ファイルがローカル ファイル システムに存在することを示します。
- *file*：表示する既存のバイナリ入力ファイルを特定します。

ローカル ファイル システム上にあるバイナリ ファイルを表示するには：

ステップ 1 */opt/CSCObac/rdu/samples/packet_cable* にディレクトリを変更します。

ステップ 2 表示するバイナリ ファイルを選択します。

ステップ 3 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin # runCfgUtil.sh-d-l unprov_packet_cable.bin
```

unprov_packet_cable.bin：表示する既存のバイナリ入力ファイルを特定します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0
Warning: Expecting config file of type docsis, but input file is of type pktc1.0.  Decoding as
pktc1.0

Off                 File Bytes          Option              Description         Value

0                   FE0101              254                 Telephony Config    1
                                                            File Start/End

      3             0B153013060E        11                  SNMP MIB Object     .iso.org.dod.internet.private.enterpri
                    2B06010401A3                                                ses.cableLabs.clabProject.clabProjPack
                    0B0202010101                                                etCable.pktcMaMib.pktcMaMibObjects.p
                    0700020102                                                  ktcMaDevBase.pktcMaDevEnable1(0),INTE
                                                                                GER,fals e(2)

...
```

（注） デフォルト入力ファイルがDOCSISであり、この例ではバイナリPacketCableファイルを使用しているため、この例では警告が表示されます。**-pkt**オプションを使用してPacketCableファイルとして入力ファイルを指定する場合、警告は表示されません。次に例を示します。

```
/opt/CSCObac/rdu/bin/# runCfgUtil.sh -d -pkt -l unprov_packet_cable.bin
```

外部バイナリ ファイルの表示

外部のバイナリ ファイルを表示するには、 **runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -d -r *file* **-u** *username* **-p** *password*

- **-d**：表示のため、コマンドがバイナリ入力ファイルを復号化するように指定します。
- **-r**：RDUに追加されているファイルとして入力ファイルを識別します。
- *file*：RDUで既存のバイナリ ファイルを特定します。
- **-u** *username*：RDUに接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDUに接続するときに使用するパスワードを指定します。

RDUに追加されているバイナリ ファイルを表示するには：

ステップ 1 表示するバイナリ ファイルを選択します。この例では、既存のバイナリ ファイル *unprov.cm* を使用し、RDU が localhost:49187 であると見なされます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -d -r unprov.cm -u admin -p changeme
```

- **unprov.cm**：RDU で既存のバイナリ ファイルを特定します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off               File Bytes        Option            Description       Value

0                 030100            3                 Network Access    Off
                                                      Control

3                 041F              4                 Class of Service

5                 010101            4.1               Class ID          1

8                 02040001F400      4.2               Maximum Downstream 128000 bits/sec
                                                       Rate

14                03040000FA00      4.3               Maximum Upstream  64000 bits/sec
                                                      Rate

20                040101            4.4               Upstream Channel  1
                                                      Priority

...

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

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

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.
```

PacketCable Basic フローのアクティブ化

runCfgUtil.sh コマンドを使用して、PacketCable Basic フローの整合性ハッシュを Basic フロー スタティック 設定ファイルに生成および挿入することをサポートします。

構文の説明

runCfgUtil.sh -t {**basic** | **secure**} **-pkt -r** *filename* **-u** *username* **-p** *password*

- **basic**：PacketCable Basic フロー整合性ハッシュを MTA スタティック設定ファイルに計算し挿入します。
- **secure**：PacketCable Basic フロー整合性ハッシュを MTA スタティック設定ファイルに挿入することを停止します。これがデフォルト設定です。
- **-r**：RDU に追加されているファイルとして入力ファイルを識別します。
- *filename*：入力ファイルを特定します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。
- **-pkt**：PacketCable MTA 設定ファイルとして入力ファイルを特定します。

PacketCable Basic フローの整合性ハッシュを Basic フロー スタティック 設定ファイルに生成および挿入することをサポートするには：

ステップ 1 PacketCable Basic フロー整合性ハッシュを挿入する Basic Flow スタティック設定ファイルを選択します。この例では、 *example_mta_config.tmpl* を使用します。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -t basic -pkt -r example_mta_config.tmpl -u admin -p changeme
```

- **example_mta_config.tmpl**：Basic Flow スタティック設定ファイルを特定します。
- **admin**：デフォルトのユーザー名を識別します。
- **changeme**：デフォルトのパスワードを識別します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off               File Bytes        Option            Description       Value

0                 FE0101            254               Telephony Config  1
                                                      File Start/End

3                 0B153013060E      11                SNMP MIB Object   .iso.org.dod.internet.
                  2B06010401A3                                          private.enterprises.ca
                  0B0202010101                                          bleLabs.clabProject.cl
                  0700020101                                            abProjPacketCable.pktc
                                                                        MtaMib.pktcMtaMibObjec
                                                                        ts.pktcMtaDevBase.pktc
                                                                        MtaDevEnabled.0,INTEGE
                                                                        R,true(1)

26                0B2530230610      11                SNMP MIB Object   .iso.org.dod.internet.
                  2B06010401A3                                          private.enterprises.ca
                  0B0202020102                                          bleLabs.clabProject.cl
                  01010109040F                                          abProjPacketCable.pktc
                  434D532E4950                                          SigMib.pktcSigMibObjec
                  464F4E49582E                                          ts.pktcNcsEndPntConfig
                  434F4D                                                Objects.pktcNcsEndPntC
                                                                        onfigTable.pktcNcsEndP
                                                                        ntConfigEntry.pktcNcsE
                                                                        ndPntConfigCallAgentId
                                                                        .9,STRING,CMS.IPFONIX.
                                                                        COM

...

371               FE01FF            254               Telephony Config  255
                                                      File Start/End

0 error(s), 0 warning(s) detected. Parsing of example_mta_config.tmpl was successful.
The file example_mta_config.tmpl was parsed successfully in 100 ms.
The parser initialization time was 44 ms.
The parser parse time was 56 ms.
```

.Tmpl 拡張子付きのファイルが、テンプレート処理中に Basic ハッシュ計算と挿入が透過的にが発生する、動的設定テンプレートになると仮定します。結果として、SecureおよびBasicモードのプロビジョニングのため同じテンプレートを使用できます。

ただし、ハッシュを挿入する前に、Secure スタティック バイナリ設定ファイルを Basic スタティック設定ファイルに変換する場合、この手順に従います。

a) 次を使用して、Secure スタティック ファイルをテンプレートに変換します。

   # **runCfgUtil -l** *input_static_filename* **-pkt -g -o** *output_template_filename*

b) 次を使用して、Secure スタティック テンプレートを Basic スタティック設定ファイルに変換します。

   # **runCfgUtil -t basic -l** *input_template_name* **-pkt -o** *output_Basic_static_filename*

このコマンドは、Basic 整合性ハッシュを Basic スタティック設定ファイルに対して計算し、挿入します。

マルチベンダー サポート用 TLV 43s の生成

マルチ ベンダー サポートを提供する TLV 43s を生成するには、**runCfgUtil.sh** コマンドを使用します。

構文の説明

runCfgUtil.sh -docsis -r *filename* **-u** *username* **-u** *password*

- **-docsis**—DOCSIS テンプレート ファイルとして入力ファイルを識別します。
- *filename*: 解析されない入力テンプレート ファイルを識別します。
- **r**：RDU に追加されているファイルとして入力ファイルを識別します。
- **-u** *username*：RDU に接続するときに使用するユーザー名を指定します。
- **-p** *password*— RDU に接続するときに使用するパスワードを指定します。

RDU に追加されているテンプレート ファイルを使用して TLV 43s を生成するには：

ステップ 1 使用するテンプレート ファイルを選択します。この例では、 *test.tmpl*と呼ばれる既存のテンプレート ファイルを使用します。**-docsis** オプションは、DOCSIS テンプレートが使用されているため使用されます。

ステップ 2 このコマンドを使用して設定ファイル ユーティリティを実行します。

```
/opt/CSCObac/rdu/bin # runCfgUtil.sh - docsis r test.tmpl -u admin -p] changeme
```

- **test.tmpl**DOCSIS 設定ファイルを識別します。
- **admin**—デフォルトのユーザー名を識別します。
- **changeme**—デフォルトのパスワードを識別します。

ユーティリティを実行後、これらと同じような結果が次に表示される必要があります。

```
Cisco Prime Cable Provisioning Configuration Utility
Version: 5.0

Off                File Bytes          Option              Description         Value

0                  2B14                43                  DOCSIS Extension
                                                           Field

2                  0803FFFFFF          43.8                Vendor ID           FF-FF-FF

7                  050D                43.5                L2VPN Encoding
```

9	010502345600 03	43.5.1	VPNID Subtype	0234560003
16	0204	43.5.2	NSI Encapsulation Subtype	
18	02020019	43.5.2.2	IEEE 802.1Q Format Subtype	25
22	2B0B	43	DOCSIS Extension Field	
24	080300000C	43.8	Vendor ID	00-00-0C (CISCO SYSTEMS, INC.)
29	010418017A30	43.1	Static Downstream Frequency	402750000
35	2B1D	43	DOCSIS Extension Field	
37	080300000C	43.8	Vendor ID	00-00-0C (CISCO SYSTEMS, INC.)
42	0316626F6F74 5F6D6F6E6974 6F725F696D61 67652E62696E	43.3	Update Boot Monitor Image	boot_monitor_image.bin
66	061071E79068 3DE8B9950536 8936F4C5312F	6	CM MIC Configuration Setting	71E790683DE8B995 05368936F4C5312F
84	0710DB0EED14 B5B3428D2B15 0DA582B41A54	7	CMTS MIC Configuration Setting	DB0EED14B5B3428D 2B150DA582B41A54
102	FF	255	End-of-Data Marker	
103	00	0	PAD	

```
0 error(s), 0 warning(s) detected. Parsing of test.tmpl was successful.
The file test.tmpl was parsed successfully in 250 ms.
The parser initialization time was 109 ms.
The parser parse time was 141 ms.
```

ダイナミック DOCSIS テンプレートで MIB を使用する

プライム ケーブル プロビジョニングに同梱されている MIB の完全なリストについては、『SNMP VarBind』を参照してください。

アプリケーション プログラミング インターフェイス（API）通話を使用するか、*rdu.properties* を変更することによって、MIB を追加できます。詳細については、PacketCable の設定を参照してください。

SNMP TLVs をテンプレートに追加できます。

- MIB を使用できない場合。『MIB を使用しない SNMP TLV の追加』を参照してください。
- ベンダー固有の MIB の場合。『ベンダー固有の MIB を持つ SNMP TLV の追加』を参照してください。

MIB 管理機能拡張

次のリストは、MIB 管理用にプライム ケーブル プロビジョニング 5.2.1 からサポートされています。

1. MIB 拡張機能は、RDU サーバで使用されている MIB ファイルの詳細についての管理オプションを提供し、RDU サーバが RDU データベースで使用可能な MIB ファイルをロードします。以前のリリースでは、MIB ファイルはファイル システムからロードされました（BPR_HOME/rdu/mib ディレクトリ）からロードされました。
2. [MIB List] フィールドの [Configuration] > [Defaults] > [System Defaults] メニュー オプションの Administrative Web UI で、テキスト ボックスに手動でファイル名を入力する代わりに、新しいボタン（MIB ファイルの選択）がデータベースの表示リストから MIB ファイルを設定するために追加されます。
3. MIB ファイル タイプ用の [Configuration] > [Files] ページの [Administrative Web UI] で、新しいクイック ビュー ボタンが MIB ファイル詳細を表示するために追加されます（ファイル名、モジュール名、親モジュール、モジュールによる参照）
4. 新しいオプション - mib が、MIB の依存関係を検出し一覧にするために runCfgUtil ツールに追加されます。デフォルトでは、BPR_HOME/rdu/mib ディレクトリで利用可能なすべてのファイルを検証します。ファイル名/ディレクトリ名が引数として指定される場合、そのディレクトリのファイルの依存関係を検証し、一覧にします。
5. 次の表では、MIB 管理の API に追加される検証について説明します。

表 *4: MIB* 管理 *API* の検証

API	タスク	サンプル エラー メッセージ

addFile	無効な MIB ファイル < arris_cm.mib > が追加されています。	MIB ファイル < arris_cm.mib > の追加に失敗しました。有効な MIB ファイルではありません:。< arris_cm.mib >。
	親 MIB < ARRIS MIB > を追加することがなく、MIB ファイル < arris_cm.mib > を追加します。	MIB ファイル < arris_cm.mib > の追加に失敗しました。データベースで依存の MIB ファイル < ARRIS MIB > を検索することはできません。
	データベース内に存在する名前の新しい MIB ファイル < SNMPv2 SMI > を追加します。	MIB ファイル < SNMPv2 SMI > の追加に失敗しました。ファイルがすでにデータベース内に存在します。
	すでにデータベースに MIB ファイル < URI TC MIB > を追加します。別の名前 < Renamed_URI TC MIB > と同じ MIB ファイルを追加します。	すでにファイル名 < uri tc mib > を持つデータベース内に存在する MIB ファイル < Renamed_URI-TC-MIB。MIB > モジュール < URI TC MIB > の追加に失敗しました。
DeleteFile	一部の他の MIB の親である MIB ファイル < ARRIS MIB > を削除します。	MIB ファイル < Arris mib > の削除に失敗しました。< Arris_cm.mib、arris_mta_pp.mib > MIB ファイルは、MIB ファイル < arris mib > に依存します。
	ロードされる MIB ファイルを削除します。（システムのデフォルト MIB リストで設定されています）。	MIB ファイル < Arris mib > の削除に失敗しました。MIB ファイルは現在使用されています。管理者のガイダンスの下で更新されるシステム デフォルト設定が必要です。
ReplaceFile	無効なコンテンツ（raw テキスト ファイル）に MIB ファイル < ARRIS MIB > を置換します。	MIB ファイル < arris mib > の置換操作に失敗しました。有効な MIB ファイルではありません。< ARRIS MIB >。
	MIB ファイル < ARRIS MIB > を別の有効な MIB ファイル < arris_cm.mib > に置き換えます。	MIB ファイル < arris-mib.MIB > の置換操作に失敗しました。MIB モジュール < ARRIS-CM-DEVICE-MIB > は、既存の MIB モジュール < ARRIS MIB > と競合しています。
	新しい親 MIB < ARRIS-ROUTER-DEVICE-MIB > を追加することなく、MIB ファイル < arris_mta_pp.mib > を置換します。	< Arris_mta_pp.mib > MIB を置換できません。データベースの依存 MIB モジュール < ARRIS-ROUTER-DEVICE-MIB > を検索することはできません。
	子によって使用されているいくつかの OID を削除することで、MIB ファイル < ARRIS MIB > を置換します。	MIB ファイル < arris-mib.MIB > の置換操作に失敗しました。MIB ファイル < arris-mib.MIB > は子 MIB と互換性がありません。

（注） RunCfgUtil スクリプトは、RDU データベースではなく引き続きファイル システム（BPR_HOME/rdu/mibs）から MIB を使用します。

MIB 移行

プライム ケーブル プロビジョニング 6.1.2 移行では、migrateDB スクリプトは MIB ディレクトリからすべてのカスタム ファイルを移行しません。rdu.properties に記載されている MIB ファイル名のみ移行します。MibList プロパティが rdu.properties で定義されていない場合、システムのデフォルト設定から MIB ファイル名を使用し、データベースにこれらのファイルを移行します。

ユーザー定義 MIB の管理

プライム ケーブル プロビジョニング データベースの移行手順では、セクションの下に記載されている推奨シーケンス内のコンポーネントを移行する必要があります。

（注） 手順 1 ~ 4 を参照してください。 Cisco プライム ケーブル プロビジョニング 6.1.1 クイック スタート ガイド

手順 1：RDU データベースをバックアップします。

手順 2：RDU データベースのバックアップを復元します。

手順 3：データベースの整合性を確認します。

手順 4：プロパティ ファイルをバックアップします。

手順 5：顧客特定の MIB ファイルとともに RDU データベースを移行します。下記で説明されている -mibdir オプションとともに、Cisco プライム ケーブル プロビジョニング 6.1.1 クイック スタート ガイド に記載されている手順に従います。

- 現在インストールされている（5.1 または 5.2）でカスタム MIB ファイルを追加した場合は、最初に $BPR_HOME/rdu/mib/ をバックアップし、それらをプライム ケーブル プロビジョニング 6.1.2 サーバに手動でコピーします。

（注） バックアップされた MIB ファイルを、デフォルトの $BPR_HOME/rdu/mibs 以外にプライム ケーブル プロビジョニング 6.1.2 サーバ内の場所にコピーして、デフォルトの 6.1.2 MIB ファイルが上書きされないようにする必要があります。

- バックアップされたデータベースと、存在する場合はバックアップされたカスタム MIB ディレクトリで、migrateDb.sh ツールを実行します。MigrateDb.sh スクリプトは、$BPR_HOME/migrationディレクトリに存在します。MigrateDb.sh スクリプトは、以下に示すように新しい MIB 移行オプションをサポートします。

- mibdir：カスタム MIB ディレクトリ パスのオプション パラメータです。-mibdir オプションは、ファイル システムに存在し、ロードされるカスタム MIB ファイルが含まれているディレクトリのパスの後にする必要があります。

例：

```
# $BPR_HOME/migration/migrateDb.sh -dbdir /var/backup/rdu-backup-20120829-031028 -mibdir
/tmp/mibs
```

- dbdir：移行するデータベースのバックアップの場所を指定します。この場合は、/var/backup です。

- mibdir：移行するカスタム mib のバックアップの場所を指定します。この場合は、/tmp/mibs です。

（注） データベースの移行とともに MIB 移行が行われ、別々に実行することはできません。