TCL による Cisco IOS スクリプティング コンフィギュレーション ガイド

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

【注意】シスコ製品をご使用になる前に、安全上の注意（**www.cisco.com/jp/go/safety_warning/**）をご確認ください。本書は、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

このマニュアルに記載されている仕様および製品に関する情報は、予告なしに変更されることがあります。このマニュアルに記載されている表現、情報、および推奨事項は、すべて正確であると考えていますが、明示的であれ黙示的であれ、一切の保証の責任を負わないものとします。このマニュアルに記載されている製品の使用は、すべてユーザ側の責任になります。

対象製品のソフトウェア ライセンスおよび限定保証は、製品に添付された『Information Packet』に記載されています。添付されていない場合には、代理店にご連絡ください。

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system.All rights reserved.Copyright © 1981, Regents of the University of California.

ここに記載されている他のいかなる保証にもよらず、各社のすべてのマニュアルおよびソフトウェアは、障害も含めて「現状のまま」として提供されます。シスコおよびこれら各社は、商品性の保証、特定目的への準拠の保証、および権利を侵害しないことに関する保証、あるいは取引過程、使用、取引慣行によって発生する保証をはじめとする、明示されたまたは黙示された一切の保証の責任を負わないものとします。

いかなる場合においても、シスコおよびその供給者は、このマニュアルの使用または使用できないことによって発生する利益の損失やデータの損傷をはじめとする、間接的、派生的、偶発的、あるいは特殊な損害について、あらゆる可能性がシスコまたはその供給者に知らされていても、それらに対する責任を一切負わないものとします。

このマニュアルで使用している IP アドレスおよび電話番号は、実際のアドレスおよび電話番号を示すものではありません。マニュアル内の例、コマンド出力、ネットワーク トポロジ図、およびその他の図は、説明のみを目的として使用されています。説明の中に実際のアドレスおよび電話番号が使用されていたとしても、それは意図的なものではなく、偶然の一致によるものです。

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries.To view a list of Cisco trademarks, go to this URL:http://www.cisco.com/go/trademarks.Third-party trademarks mentioned are the property of their respective owners.The use of the word partner does not imply a partnership relationship between Cisco and any other company.(1110R)

© 2017 Cisco Systems, Inc. All rights reserved.

目 次

最初にお読みください 1

TCL による Cisco IOS XE スクリプティング 3

機能情報の確認 3

TCL による Cisco IOS XE スクリプティングの前提条件 4

TCL による Cisco IOS XE スクリプティングの制約事項 4

TCL による Cisco IOS XE スクリプティングについて 6

Cisco IOS XE ソフトウェアの TCL シェル 6

TCL プリコンパイラ 7

SNMP MIB オブジェクト アクセス 7

TCL シェルにおけるカスタム拡張 7

TCL シェルにおける SNMP MIB カスタム拡張 8

TCL による Cisco IOS XE スクリプティングの設定方法 12

TCL シェルのイネーブル化と CLI を使用したコマンドの入力 12

トラブルシューティングのヒント 15

TCL シェルを使用した SNMP MIB オブジェクトへのアクセス 15

トラブルシューティングのヒント 18

定義済み TCL スクリプトの実行 19

TCL による Cisco IOS XE スクリプティングの設定例 20

show interfaces コマンドを使用した TCL スクリプトの例 20

SMTP サポートのための TCL スクリプトの例 20

SNMP MIB アクセスのための TCL スクリプトの例 22

その他の参考資料 23

TCL による Cisco IOS XE スクリプティングの機能情報 24

用語集 25

第 1 章

最初にお読みください

Cisco IOS XE 16 に関する重要な情報

有効な 2 つのリリース、Cisco IOS XE Release 3.7.0E（Catalyst スイッチ用）および Cisco IOS XE Release 3.17S（アクセス/エッジルーティング用）が 1 つのバージョンとしての統合リリース Cisco IOS XE 16 に展開されて（マージされて）、スイッチング/ルーティング ポートフォリオに含まれる広範なアクセスおよびエッジ製品が盛り込まれた 1 つのリリースが実現されています。

（注） 技術構成ガイドにある機能情報の表には、その機能がいつ導入されたかが記載されています。その機能で他のプラットフォームがサポートされた時期については、記載されている場合もされていない場合もあります。特定の機能がご使用のプラットフォームでサポートされているかどうかを確認するには、製品のランディング ページに掲載されている技術構成ガイドを参照してください。ご使用の製品のランディング ページに技術構成ガイドが表示されていれば、その機能はプラットフォームでサポートされています。

第 2 章

TCL による Cisco IOS XE スクリプティング

TCL による Cisco IOS XE スクリプティング機能では、Cisco IOS XE のコマンドライン インターフェイス（CLI）からツール コマンド言語（TCL）バージョン 8.3.4 のコマンドを実行できます。

- 機能情報の確認，3 ページ
- TCL による Cisco IOS XE スクリプティングの前提条件，4 ページ
- TCL による Cisco IOS XE スクリプティングの制約事項，4 ページ
- TCL による Cisco IOS XE スクリプティングについて，6 ページ
- TCL による Cisco IOS XE スクリプティングの設定方法，12 ページ
- TCL による Cisco IOS XE スクリプティングの設定例，20 ページ
- その他の参考資料，23 ページ
- TCL による Cisco IOS XE スクリプティングの機能情報，24 ページ
- 用語集，25 ページ

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、Bug Search Tool およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェアイメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

TCL による Cisco IOS XE スクリプティングの前提条件

- TCL プログラミングと Cisco IOS XE コマンドをよく理解していること。
- TCL コマンドは Cisco IOS XE CLI を使用して、TCL コンフィギュレーション モードから実行できます。グローバル コンフィギュレーション モードと同様に、TCL のコンフィギュレーション モードには、特権 EXEC モードからアクセスできます。**enable** コマンド パスワードを使用してアクセスを制限することで、特権 EXEC モードへのアクセスを管理する必要があります。

TCL による Cisco IOS XE スクリプティングの制約事項

- TCL スクリプトで Cisco IOS XE コンフィギュレーション コマンドを使用する場合、サブモード コマンドはコンフィギュレーション コマンドと同じ行に引用符で囲んだ変数として入力する必要があります。
- エラー メッセージが表示されますが、エラーによって TCL シェルが無限ループで動作する可能性があるため、TCL スクリプトが正常に動作していることを確認する必要があります。

注意 TCL サーバ ソケットを使用して Telnet ポートおよび FTP ポート（それぞれ 23 および 21）を待ち受けると、Cisco IOS XE ソフトウェアでのこれらのポートの通常の処理がプリエンプション処理されます。

- 次の表に、Cisco IOS XE ソフトウェアでは、標準的な TCL ドキュメントの記載どおりに動作しない TCL コマンドとライブラリ コールを示します。

表 *1*：*Cisco IOS XE* ソフトウェア上で通常とは異なる動作をする *TCL* コマンド オプション

コマンド	キーワード	引数	サポートの有無	注
after	**ms**	*script*	一部	CLI **tclsh** コマンドが使用されると、同じルータで Embedded Syslog Manager（ESM）がアクティブになるまで、イベントループが実装されません。 **after** を使用して入力されたコマンドは、**update** コマンドを使用するまで実行されません。スリープ モード（**after** コマンド）は **ms** キーワードでのみ動作します。
file	**-time**	*atime*	なし	ファイルへのアクセス時間を設定する、オプションの **-time** キーワードは、Cisco IOS XE ソフトウェアではサポートされていません。
file	**-time**	*mtime*	なし	ファイルの変更時間を設定する、オプションの **-time** キーワードは、Cisco IOS XE ソフトウェアではサポートされていません。

コマンド	キーワード	引数	サポートの有無	注
fileevent			一部	CLI **tclsh** コマンドが使用されると、同じルータで Embedded Syslog Manager（ESM）がアクティブになるまで、イベントループが実装されません。**fileevent** TCL コマンドを使用して入力されたコマンドは、**update** コマンドを使用するまで実行されません。
history	**!***n*		一部	**!***n* ショートカットは Cisco IOS XE ソフトウェアでは動作しません。**history** TCL コマンドは **redo***n* キーワードを指定して使用します。
負荷			なし	CLI **load** コマンドを使用すると、「dynamic loading not available on this system」というエラーメッセージが表示されます。

TCL による Cisco IOS XE スクリプティングについて

Cisco IOS XE ソフトウェアの TCL シェル

Cisco IOS XE TCL シェルは、ユーザが Cisco IOS XE CLI プロンプトから直接 TCL コマンドを実行できるようにするために設計されました。Cisco IOS XE ソフトウェアには、Embedded Syslog Manager（ESM）や音声自動応答装置（IVR）など、TCL インタープリタを実装の一部に使用する

サブシステムが搭載されています。これらのサブシステムには、TCL シェルでは使用できない独自の専用コマンドおよびキーワード オプションがあります。

Cisco IOS XE ソフトウェアで TCL スクリプトを構築し、実行する方法がいくつか開発されました。TCL シェルは有効化され、TCL コマンドを 1 行ずつ入力できます。入力された TCL コマンドは TCL インタープリタに送られます。コマンドが、有効な TCL であると認識された場合は実行され、その結果が tty に送信されます。コマンドが認識された TCL コマンドではなかった場合は、Cisco IOS XE CLI パーサーに送られます。コマンドが TCL コマンドまたは Cisco IOS XE コマンドではない場合、2 つのエラー メッセージが表示されます。定義済みの TCL スクリプトを Cisco IOS XE ソフトウェア以外で作成し、フラッシュ メモリまたはディスクメモリに転送して Cisco IOS XE ソフトウェア上で実行することもできます。また、TCL スクリプトを作成し、Cisco IOS XE ソフトウェアで実行する前にコードをプリコンパイルすることも可能です。

同じルータ上の複数のユーザが、干渉なく同時に TCL コンフィギュレーション モードに入ることができます。これは、それぞれの TCL シェル セッションが個別のインタープリタおよび TCL サーバ セッションを起動するためです。各 TCL プロセスによって処理される tty インターフェイス番号はサーバ プロセス名で表され、**show process** CLI コマンドを使用して表示できます。

TCL シェルを使用して、Cisco IOS XE CLI EXEC コマンドを TCL スクリプトで実行できます。TCL シェルを使用して CLI コマンドを実行すると、初心者ユーザに手順を説明するメニューの作成、繰り返し作業の自動化、**show** コマンドのカスタム出力の作成が可能です。

TCL プリコンパイラ

Cisco IOS XE TCL の実装により、TclPro プリコンパイラでプリコンパイルされたスクリプトのロードがサポートされます。プリコンパイルされたスクリプトは難読化されているため、セキュリティおよび一貫性の対策を許容します。

SNMP MIB オブジェクト アクセス

簡易ネットワーク管理プロトコル（SNMP）MIB オブジェクトに簡単にアクセスできるように設計された、UNIX ライクな SNMP コマンドのセットが作成されました。TCL シェルは手動でまたは TCL スクリプトを使用してイネーブルにすることができ、新しいコマンドを入力して、MIB オブジェクトに対して指定された get 操作および set 操作を実行できるようになります。ユーザの操作性を向上させるために、新しいコマンドには UNIX の SNMP アクセスに使用されるコマンドと似た名前が付けられています。SNMP コマンドにアクセスする方法については、TCL シェルを使用した SNMP MIB オブジェクトへのアクセス，（15 ページ）を参照してください。

TCL シェルにおけるカスタム拡張

Cisco IOS XE での TCL シェルの実装には、いくつかのカスタム コマンド拡張が含まれます。これらの拡張は TCL コンフィギュレーション モードのみで動作します。次の表に、これらのコマンド拡張を示します。

表 *2*：*Cisco IOS XE* のカスタム *TCL* コマンド拡張

コマンド	説明
ios_config	Cisco IOS XE CLI コンフィギュレーション コマンドを実行します。
log_user	TCL コンフィギュレーション モードで TCL コマンド出力を切り替えます。
typeahead	ルータの標準入力（stdin）バッファ ファイルにテキストを書き込みます。
tclquit	TCL シェルを終了します。**exit** と同じです。

TCL シェルにおける SNMP MIB カスタム拡張

Cisco IOS XE での TCL シェルの実装には、SNMP MIB オブジェクトへのアクセスのためのいくつかのカスタム コマンド拡張が含まれます。これらの拡張は TCL コンフィギュレーション モードのみで動作します。次の表に、これらのコマンド拡張を示します。

表 *3*：*SNMP MIB* アクセスのための *Cisco IOS XE* カスタム *TCL* コマンド拡張

コマンド	説明
snmp_getbulk	MIB テーブルの大規模なセクションを取得します。このコマンドは、SNMP **getbulk** コマンドと類似しています。構文のフォーマットは次のとおりです。 **snmp_getbulk***community-string non-repeaters max-repetitions oid* [*oid2 oid3...*] • オブジェクトを取得する SNMP コミュニティを指定するには、*community-string* 引数を使用します。 • get-next 操作で取得するオブジェクトの数を指定するには、*non-repeaters* 引数を使用します。 • 残りのオブジェクトを取得する間に試行する get-next 操作の最大回数を指定するには、*max-repetitions* 引数を使用します。 • 取得するオブジェクト ID（複数可）を指定するには、*oid* 引数を使用します。

コマンド	説明
snmp_getid	ルータ上の SNMP エンティティから、次の変数を取得します。 • sysDescr.0 • sysObjectID.0 • sysUpTime.0 • sysContact.0 • sysName.0 • sysLocation.0 このコマンドは、SNMP **getid** コマンドと類似しています。構文のフォーマットは次のとおりです。 **snmp_getid***community-string*
snmp_getnext	ルータ上の SNMP エンティティから、個別の変数を 1 セット取得します。このコマンドは、SNMP **getnext** コマンドと類似しています。構文のフォーマットは次のとおりです。 **snmp_getnext***community-stringoid* [*oid2oid3...*]
snmp_getone	ルータ上の SNMP エンティティから、個別の変数を 1 セット取得します。このコマンドは、SNMP **getone** コマンドと類似しています。構文のフォーマットは次のとおりです。 **snmp_getone***community-stringoid* [*oid2oid3...*]

コマンド	説明
snmp_setany	

コマンド	説明

	指定された変数の現在の値を取得し、変数でset要求を実行します。このコマンドは、SNMP **setany** コマンドと類似しています。構文のフォーマットは次のとおりです。
	snmp_setany*community-stringoidtypeval* [*oid2type2val2...*]
	• 取得するオブジェクト タイプを指定するには、*type* 引数を使用します。*type* は次のいずれかです。
	  • **-i**：整数。管理対象オブジェクトのコンテキスト内の番号が付けられたタイプを指定する場合は、32 ビットの数字が使用されます。たとえば、ルータ インターフェイスの動作ステータスを 1 に設定した場合はアップ、2 に設定した場合はダウンを示します。
	  • **-u**：Unsigned32。0 ～ 2 32 - 1 の範囲の 10 進数を表す場合に使用される 32 ビットの数字。
	  • **-c**：Counter32。最小値 0、最大値 2 32 - 1 の 32 ビット数。最大値に到達すると、カウンタが 0 にリセットされ、そこから再び開始されます。
	  • **-g**：Gauge。最小値 0、最大値 2 32 - 1 の 32 ビット数。数値は意図的に増減できます。たとえば、gauge オブジェクト タイプを使用して、ルータ上のインターフェイスの速度を測定できます。
	  • **-o**：オクテット文字列。16 進数表記のオクテット文字列は、物理アドレスを表す場合に使用されます。
	  • **-d**：表示文字列。テキスト表記のオクテット文字列は、テキスト文字列を表す場合に使用されます。
	  • **-ipv4**：IP バージョン 4 アドレス。
	  • **-oid**：オブジェクト ID。

コマンド	説明
	• 取得するオブジェクト ID（複数可）の値を指定するには、*val* 引数を使用します。

TCL による Cisco IOS XE スクリプティングの設定方法

TCL シェルのイネーブル化と CLI を使用したコマンドの入力

インタラクティブ TCL シェルをイネーブルにし、Cisco IOS XE CLI プロンプトを使用して TCL コマンドを 1 行ずつ入力するには、次の作業を実行します。オプションの手順には、符号化ファイルのデフォルトの位置、および初期化スクリプトの指定が含まれます。

手順の概要

1. **enable**
2. **configureterminal**
3. **scriptingtclencdir***location-url*
4. **scriptingtclinit***init-url*
5. **scriptingtcllow-memory***bytes*
6. **exit**
7. **tclsh**
8. 必要な TCL コマンド言語構文を入力します。
9. **ios_config“***cmd***”“***cmd-option***”**
10. **exec“***exec-cmd***”**
11. **exit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。

	コマンドまたはアクション	目的
ステップ 2	**configureterminal** 例： `Router# configure terminal`	（任意）グローバルコンフィギュレーションモードを開始します。 • 符号化ファイルと初期化スクリプトのいずれか、または両方を使用している場合は、TCL シェルのイネーブル化と CLI を使用したコマンドの入力からTCL シェルのイネーブル化と CLI を使用したコマンドの入力 までを実行します。
ステップ 3	**scriptingtclencdir***location-url* 例： `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	（任意）TCL **encoding** コマンドで使用される外部符号化ファイルのデフォルトの場所を指定します。
ステップ 4	**scriptingtclinit***init-url* 例： `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	（任意）TCL シェルがイネーブルの場合に実行する初期化スクリプトを指定します。
ステップ 5	**scriptingtcllow-memory***bytes* 例： `Router(config)# scripting tcl low-memory 33117513`	（任意）TCL ベースのアプリケーションに対する空きメモリの下限を指定します。メモリのしきい値は 0～4294967295 バイトに設定できます。 （注） メモリの最低空き容量がしきい値を下回ると、TCL は実行しているスクリプトを中断します。これによって、TCL インタープリタが割り当てる RAM が多すぎて、ルータがクラッシュすることを防止できます。
ステップ 6	**exit** 例： `Router(config)# exit`	（任意）グローバルコンフィギュレーションモードを終了し、特権 EXEC モードに戻ります。
ステップ 7	**tclsh** 例： `Router# tclsh`	インタラクティブ TCL シェルをイネーブルにし、TCL コンフィギュレーションモードを開始します。

	コマンドまたはアクション	目的
ステップ 8	必要な TCL コマンド言語構文を入力します。 例： `Router(tcl)# proc get_bri {}`	TCL コンフィギュレーションモードで入力されたコマンドが、まずインタラクティブ TCL インタープリタに送信されます。コマンドが有効な TCL コマンドではない場合は CLI パーサーに送信されます。
ステップ 9	**ios_config**“*cmd*”“*cmd-option*” 例： `Router(tcl)# ios_config “interface Ethernet 2/0” “no keepalive”`	（任意）TCL スクリプトを使用して CLI コマンドおよびオプションで TCL コマンド **ios_config** を指定し、ルータ設定を変更します。すべての引数とサブモード コマンドを CLI コンフィギュレーション コマンドと同じ行に入力する必要があります。 • この例では、引用符で囲まれた最初の引数がイーサネットインターフェイスを設定し、インターフェイス コンフィギュレーションモードを開始します。引用符で囲まれた 2 つ目の引数はキープアライブ オプションを設定します。これらの 2 つの CLI 文が TCL コマンドの別々の行に入力されると、この設定は機能しません。
ステップ 10	**exec**“*exec-cmd*” 例： `Router(tcl)# exec “show interfaces”`	（任意）CLI コマンドで TCL コマンド **exec** を指定して、TCL スクリプトから Cisco IOS XE CLI EXEC モード コマンドを実行します。 • この例では、ルータのインターフェイス情報が表示されます。
ステップ 11	**exit** 例： `Router(tcl)# exit`	TCL コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

例

次の出力例は、ルータのイーサネット インターフェイス 0 についての情報を示しています。**showinterfaces** コマンドは TCL コンフィギュレーション モードで実行されています。

```
Router# tclsh
Router(tcl)# exec “show interfaces”
Ethernet 0 is up, line protocol is up
```

```
  Hardware is MCI Ethernet, address is 0000.0c00.750c (bia 0000.0c00.750c)
  Internet address is 10.108.28.8, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Last clearing of "show interface" counters 0:00:00
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 2000 bits/sec, 4 packets/sec
     1127576 packets input, 447251251 bytes, 0 no buffer
     Received 354125 broadcasts, 0 runts, 0 giants, 57186* throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5332142 packets output, 496316039 bytes, 0 underruns
     0 output errors, 432 collisions, 0 interface resets, 0 restarts
.
.
.
```

トラブルシューティングのヒント

TCL スクリプトで TCL **puts** コマンドを使用して、コマンドの実行をトレースします。

TCL シェルを使用した SNMP MIB オブジェクトへのアクセス

インタラクティブ TCL シェルをイネーブルにし、TCL コマンドを入力して MIB オブジェクトに対する操作を実行するには、次の作業を実行します。

はじめる前に

SNMP コミュニティ設定は、ルータの実行コンフィギュレーションに含める必要があります。

手順の概要

1. **enable**
2. **configureterminal**
3. **scriptingtclencdir***location-url*
4. **scriptingtclinit***init-url*
5. **exit**
6. **tclsh**
7. 必要な TCL コマンド言語構文を入力します。
8. **snmp_getbulk***community-stringnon-repeatersmax-repetitionsoid*[*oid2 oid3...*]
9. **snmp_getid***community-string*
10. **snmp_getnext***community-stringoid*[*oid2 oid3...*]
11. **snmp_getone***community-stringoid*[*oid2 oid3...*]
12. **snmp_setany***community-stringoidtypeval*[*oid2 type2 val2...*]
13. **exit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configureterminal** 例： `Router# configure terminal`	（任意）グローバル コンフィギュレーション モードを開始します。 • 符号化ファイルと初期化スクリプトのいずれか、または両方を使用している場合は、TCL シェルを使用した SNMP MIB オブジェクトへのアクセス から TCL シェルを使用した SNMP MIB オブジェクトへのアクセス までを実行します。
ステップ 3	**scriptingtclencdir***location-url* 例： `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	（任意）TCL **encoding** コマンドで使用される外部符号化ファイルのデフォルトの場所を指定します。
ステップ 4	**scriptingtclinit***init-url* 例： `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	（任意）TCL シェルがイネーブルの場合に実行する初期化スクリプトを指定します。
ステップ 5	**exit** 例： `Router(config)# exit`	（任意）グローバル コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 6	**tclsh** 例： `Router# tclsh`	インタラクティブ TCL シェルをイネーブルにし、TCL コンフィギュレーション モードを開始します。
ステップ 7	必要な TCL コマンド言語構文を入力します。 例： `Router(tcl)# proc get_bri {}`	TCL コンフィギュレーション モードで入力されたコマンドが、まずインタラクティブ TCL インタープリタに送信されます。コマ

	コマンドまたはアクション	目的
		ンドが有効な TCL コマンドではない場合は CLI パーサーに送信されます。
ステップ 8	**snmp_getbulk***community-stringnon-repeatersmax-repetitionsoid*[*oid2 oid3...*] 例： `Router(tcl)# snmp_getbulk public 1 3 1.3.6.1.2.1.1.1 1.3.6.1.2.1.10.18.8.1.1`	（任意）MIB テーブルの大規模なセクションを取得します。 • オブジェクトを取得する SNMP コミュニティを指定するには、*community-string* 引数を使用します。 • get-next 操作で取得するオブジェクトの数を指定するには、*non-repeaters* 引数を使用します。 • 残りのオブジェクトを取得する間に試行する get-next 操作の最大回数を指定するには、*max-repetitions* 引数を使用します。 • 取得するオブジェクト ID（複数可）を指定するには、*oid* 引数を使用します。
ステップ 9	**snmp_getid***community-string* 例： `Router(tcl)# snmp_getid private`	（任意）ルータ上の SNMP エントリから、引数 sysDesrc.0、sysObjectID.0、sysUpTime.0、sysContact.0、sysName.0、および sysLocation.0 を取得します。 • オブジェクトを取得する SNMP コミュニティを指定するには、*community-string* 引数を使用します。
ステップ 10	**snmp_getnext***community-stringoid*[*oid2 oid3...*] 例： `Router(tcl)# snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0`	（任意）MIB テーブルから個別の変数のセットを取得します。 • オブジェクトを取得する SNMP コミュニティを指定するには、*community-string* 引数を使用します。 • 取得するオブジェクト ID（複数可）を指定するには、*oid* 引数を使用します。

	コマンドまたはアクション	目的
ステップ 11	**snmp_getone***community-stringoid*[*oid2 oid3...*] 例： `Router(tcl)# snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0`	（任意）MIB テーブルから個別の変数のセットを取得します。 • オブジェクトを取得する SNMP コミュニティを指定するには、*community-string* 引数を使用します。 • 取得するオブジェクト ID（複数可）を指定するには、*oid* 引数を使用します。
ステップ 12	**snmp_setany***community-stringoidtypeval*[*oid2 type2 val2...*] 例： `Router(tcl)# snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCL-SNMP_TEST`	（任意）指定された変数の現在の値を MIB テーブルから取得し、変数で set 要求を実行します。 • オブジェクトの値を取得および設定する SNMP コミュニティを指定するには、*community-string* 引数を使用します。 • 取得および設定するオブジェクト ID（複数可）を指定するには、*oid* 引数を使用します。 • 取得および設定するオブジェクトタイプを指定するには、*type* 引数を使用します。 • 取得および設定するオブジェクトの値を指定するには、*val* 引数を使用します。
ステップ 13	**exit** 例： `Router(tcl)# exit`	TCL コンフィギュレーションモードを終了し、特権 EXEC モードに戻ります。

トラブルシューティングのヒント

TCL スクリプトで TCL **puts** コマンドを使用して、コマンドの実行をトレースします。

定義済み TCL スクリプトの実行

Cisco IOS XE ソフトウェアで定義済み TCL スクリプトを実行するには、次の任意の作業を実行します。

はじめる前に

次の作業を実行する前に、Cisco IOS XE ソフトウェアで実行する TCL スクリプトを生成する必要があります。TCL スクリプトは、Cisco IOS XE ファイル システム（IFS）でサポートされる TFTP、FTP、および rcp などのファイル システムを使用して、内部フラッシュ メモリに転送できます。また、TCL スクリプトはリモートの場所から転送することもできます。

手順の概要

1. **enable**
2. **tclsh**
3. ファイル名とパスを指定して TCL source コマンドを入力します。
4. **exit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**tclsh** 例： `Router# tclsh`	インタラクティブ TCL シェルをイネーブルにし、TCL コンフィギュレーション モードを開始します。
ステップ 3	ファイル名とパスを指定して TCL source コマンドを入力します。 例： `Router(tcl)# source slot0:test.tcl`	TCL コンフィギュレーション モードで入力されたコマンドが、まずインタラクティブ TCL インタープリタに送信されます。コマンドが有効な TCL コマンドではない場合は CLI パーサーに送信されます。
ステップ 4	**exit** 例： `Router(tcl)# exit`	TCL コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

TCL による Cisco IOS XE スクリプティングの設定例

show interfaces コマンドを使用した TCL スクリプトの例

TCL 正規表現エンジンを使用して、スクリプトで **show** コマンドから特定の情報をフィルタリングし、カスタム フォーマットで表示できます。次に、**show interfaces** コマンドの出力をフィルタリングし、ルータ上の BRI インターフェイスのカンマ区切りリストを作成する例を示します。

```
tclsh
proc get_bri {} {
    set check ""
    set int_out [exec "show interfaces"]
    foreach int [regexp -all -line -inline "(^BRI\[0-9]/\[0-9])" $int_out] {
        if {![string equal $check $int]} {
            if {[info exists bri_out]} {
                append bri_out "," $int
             } else {
                 set bri_out $int
             }
             set check $int
        }
    }
    return $bri_out
}
```

SMTP サポートのための TCL スクリプトの例

次の TCL スクリプトは、ルータからの E メール メッセージの送信に役立ちます。

```
##
## Place required comments here!!!
##
package provide sendmail 2.0
# Sendmail procedure for Support
namespace eval ::sendmail {
    namespace export initialize configure sendmessage sendfile
    array set ::sendmail::sendmail {
        smtphost    mailhub
        from        ""
        friendly    ""
    }
    proc configure {} {}
    proc initialize {smtphost from friendly} {
        variable sendmail
        if {[string length $smtphost]} then {
            set sendmail(smtphost) $smtphost
        }
        if {[string length $from]} then {
            set sendmail(from) $from
        }
        if {[string length $friendly]} then {
            set sendmail(friendly) $friendly
        }
    }
    proc sendmessage {toList subject body {tcl_trace 0}} {
```

```
        variable sendmail
        set smtphost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)
        if {$trace} then {
            puts stdout "Connecting to $smtphost:25"
        }
        set sockid [socket $smtphost 25]
## DEBUG
set status [catch {
        puts $sockid "HELO $smtphost"
        flush $sockid
        set result [gets $sockid]
        if {$trace} then {
            puts stdout "HELO $smtphost\n\t$result"
        }
        puts $sockid "MAIL From:<$from>"
        flush $sockid
        set result [gets $sockid]
        if {$trace} then {
            puts stdout "MAIL From:<$from>\n\t$result"
        }
        foreach to $toList {
            puts $sockid "RCPT To:<$to>"
            flush $sockid
        }
        set result [gets $sockid]
        if {$trace} then {
            puts stdout "RCPT To:<$to>\n\t$result"
        }
        puts  $sockid "DATA "
        flush $sockid
        set result [gets  $sockid]
        if {$trace} then {
            puts stdout "DATA \n\t$result"
        }
        puts  $sockid "From: $friendly <$from>"
        foreach to $toList {
            puts $sockid "To:<$to>"
        }
        puts  $sockid "Subject: $subject"
        puts  $sockid "\n"
        foreach line [split $body "\n"] {
            puts  $sockid " $line"
        }
        puts  $sockid "."
        puts  $sockid "QUIT"
        flush $sockid
        set result [gets  $sockid]
        if {$trace} then {
            puts stdout "QUIT\n\t$result"
        }
} result]
        catch {close $sockid }
        if {$status} then {
       return -code error $result
        }
        return
    }
    proc sendfile {toList filename subject {tcl_trace 0}} {
        set fd [open $filename r]
        sendmessage $toList $subject [read $fd] $trace
        return
    }
}
```

SNMP MIB アクセスのための TCL スクリプトの例

TCL シェルおよび TCL コマンドを使用して MIB 上でアクションを実行できます。次に、SNMP へのアクセスを許可するコミュニティアクセス文字列の設定例を示します。パブリックアクセスは読み取り専用ですが、プライベート アクセスは読み取りと書き込みが可能です。次に、**snmp_getbulk** TCL コマンド拡張を使用して、テーブルの大量のセクションを一度に取得する例を示します。

snmp_getbulk コマンドを発行する場合は、*non-repeaters* と *max-repetitions* の 2 つの引数を設定する必要があります。*non-repeaters* 引数は、シンプルな **snmp_getnext** 操作で最初の N 個のオブジェクトが取得されることを指定します。*max-repetitions* 引数は、残りのオブジェクトを取得するために最大 M 回の **snmp_getnext** 操作が試行されることを指定します。

次の例では、sysUpTime（1.3.6.1.2.1.1.2.0）、ifDescr（1.3.6.1.2.1.2.2.1.2）、および ifType（1.3.6.1.2.1.2.2.1.3）の 3 つのバインディングが使用されます。要求される変数バインドの合計数は、式 N + (M * R) によって指定されます。N は non-repeaters の数（この例では 1）、M は max-repetitions の数（この例では 5）、R は要求されるオブジェクト数（この場合は 2 つで、ifDescr と ifType）です。式 1 + (5 * 2) を使用すると、11 になります。これは、この **snmp_getbulk** 要求コマンドで取得できる変数バインドの合計数です。

個別の変数の例における結果には、sysUpTime.0 で取得された値が 1336090 として含まれています（単位はミリ秒）。ifDescr.1（最初のインターフェイスの説明）で取得された値は FastEthernet0/0、ifType.1（最初のインターフェイス タイプ）で取得された値は ethernetCsmacd に対応する 6 です。

```
snmp-server community public RO
snmp-server community private RW
tclsh
  snmp_getbulk public 1 5 1.3.6.1.2.1.1.2.0  1.3.6.1.2.1.2.2.1.2  1.3.6.1.2.1.2.2.1.3
  {<obj oid='sysUpTime.0' val='1336090'/>}
  {<obj oid='ifDescr.1' val='FastEthernet0/0'/>}
  {<obj oid='ifType.1' val='6'/>}
  {<obj oid='ifDescr.2' val='FastEthernet1/0'/>}
  {<obj oid='ifType.2' val='6'/>}
  {<obj oid='ifDescr.3' val='Ethernet2/0'/>}
  {<obj oid='ifType.3' val='6'/>}
  {<obj oid='ifDescr.4' val='Ethernet2/1'/>}
  {<obj oid='ifType.4' val='6'/>}
  {<obj oid='ifDescr.5' val='Ethernet2/2'/>}
  {<obj oid='ifType.5' val='6'/>}
```

次に、**snmp_getid** TCL コマンド拡張を使用して、ルータ上の SNMP エンティティから sysDescr.0、sysObjectID.0、sysUpTime.0、sysContact.0、sysName.0、および sysLocation.0 変数（この例では、system.1.0、system.2.0、system.3.0、system.4.0、system.5.0、および system.6.0 と表記）を取得する例を示します。

```
tclsh
 snmp_getid public
 {<obj oid='system.1.0' val='Cisco Internetwork Operating System Software
 Cisco IOS XE(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
 [geotpi2itd1 124]
 Copyright (c) 1986-2003 by Cisco Systems, Inc.
 Compiled Wed 21-May-03 16:16 by engineer'/>}
 {<obj oid='system.2.0' val='products.223'/>}
 {<obj oid='sysUpTime.0' val='6664317'/>}
 {<obj oid='system.4.0' val='1-800-553-2447 - phone the TAC'/>}
 {<obj oid='system.5.0' val='c7200.myCompany.com'/>}
 {<obj oid='system.6.0' val='Bldg 24, San Jose, CA'/>}
```

次に、**snmp_getnext** TCL コマンド拡張を使用して、ルータ上の SNMP エンティティから個別の変数のセットを取得する例を示します。

```
snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
 {<obj oid='system.2.0' val='products.223'/>}
 {<obj oid='sysUpTime.0' val='6683320'/>}
```

次に、**snmp_getone** TCL コマンド拡張を使用して、ルータ上の SNMP エンティティから個別の変数のセットを取得する例を示します。

```
snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
 {<obj oid='system.1.0' val='Cisco Internetwork Operating System Software
 Cisco IOS XE(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
 [geotpi2itd1 124]
 Copyright (c) 1986-2003 by Cisco Systems, Inc.
 Compiled Wed 21-May-03 16:16 by engineer'/>}
 {<obj oid='system.2.0' val='products.223'/>}
```

次に、**snmp_setany** TCL コマンド拡張を使用して、ルータの設定を変更する例を示します。この例では、ルータのホスト名が TCLSNMP-HOST に変更されます。

```
tclsh
 snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCLSNMP-HOST
 {<obj oid='system.5.0' val='TCLSNMP-HOST'/>}
```

その他の参考資料

ここでは、署名付き TCL スクリプト機能の関連資料について説明します。

関連資料

関連項目	マニュアル タイトル
Cisco PKI の概要：PKI の理解と計画 PKI の実装と管理	『*Security Configuration Guide, Release 12.4*』
PKI コマンド：コマンド構文、コマンド モード、コマンド履歴、デフォルト、使用に関する注意事項、および例	『*Cisco IOS Security Command Reference, Release 12.4*』

標準

規格	タイトル
なし	--

MIB

MIB	MIB のリンク
なし	選択したプラットフォーム、Cisco ソフトウェア リリース、およびフィーチャ セットの MIB を検索してダウンロードする場合は、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

RFC

RFC	タイトル
なし	--

シスコのテクニカル サポート

説明	リンク
シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンラインリソースを提供しています。 お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。 シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。	http://www.cisco.com/en/US/support/index.html

TCL による Cisco IOS XE スクリプティングの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェアイメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *4*：*TCL* による *Cisco IOS XE* スクリプティングの機能情報

機能名	リリース	機能情報
TCL による Cisco IOS XE スクリプティング	Cisco IOS XE	TCL による Cisco IOS XE スクリプティング機能では、Cisco IOS XE のコマンドライン インターフェイス（CLI）からツール コマンド言語（TCL）バージョン 8.3.4 のコマンドを実行できます。 この機能は、Cisco IOS XE Release 2.1 で、Cisco ASR 1000 シリーズ アグリゲーション サービス ルータに導入されました。 次のコマンドが導入または変更されました。**scriptingtclencdir**、**scriptingtclinit**、**scriptingtcllow-memory**、**tclquit**、**tclsh**
TCL SNMP MIB アクセス	Cisco IOS XE	TCL SNMP MIB アクセス機能では、簡易ネットワーク管理プロトコル（SNMP）MIB オブジェクトに簡単にアクセスするための UNIX ライクな SNMP コマンドのセットが導入されました。

用語集

CA：認証局。証明書要求の管理と、関係する IP セキュリティ ネットワーク デバイスへの証明書の発行を担当しているサービス。このサービスは、参加デバイスを一元的に管理します。またこれらのサービスによって受信者は、明示的に信頼してアイデンティティを確認し、デジタル証明書を作成できます。

証明書：ユーザ名またはデバイス名を公開キーにバインドする電子ドキュメント。証明書は、一般的にデジタル署名を確認するために使用されます。

CRL：証明書失効リスト。失効した証明書のリストが含まれる電子ドキュメントです。CRL は、証明書を発行した CA によって作成され、デジタル署名されます。CRL には、証明書の発行日と失効日が含まれています。現行の CRL が失効すると、新しい CRL が発行されます。

IPsec：IP セキュリティ。

ピア証明書：ピアが提示する証明書で、ピアの公開キーが含まれており、トラストポイント CA によって署名されています。

PKI：公開キー インフラストラクチャ。セキュアに設定された通信に使用されているネットワーク コンポーネントの暗号キーと ID 情報を管理するシステムです。

RA：登録局。CA のプロキシとして機能するサーバで、CA がオフラインのときでも CA の機能を継続できます。RA は CA サーバ上に設定するのが通常ですが、別アプリケーションとして、稼働のための別デバイスを必要とする場合もあります。

RSA キー：公開キー暗号化システムで、Ron Rivest（ロナルド・リベスト）、Adi Shamir（アディ・シャミア）、Leonard Adleman（レオナルド・エーデルマン）の 3 人によって開発されました。デバイスの証明書を取得するには、RSA キー ペア（公開キーと秘密キー）が必要です。

SHA1：Secure Hash Algorithm 1。

SSH：セキュア シェル。

SSL：Secure Socket Layer。