



## セキュア シェルバージョン2 サポート

セキュア シェルバージョン2 サポート機能で、セキュア シェル (SSH) バージョン2 を設定できます (SSH バージョン1 サポートは、以前のシスコ ソフトウェア リリースに実装されていました)。SSH は、信頼性の高いトランスポート層の上部で実行され、強力な認証機能と暗号化機能を提供します。SSH では、信頼できる転送として定義されているのは TCP のみです。SSH で、ネットワーク上の他のコンピュータに安全にアクセスしたり、コマンドを安全に実行できます。SSH とともに提供されるセキュア コピー プロトコル (SCP) 機能で、ファイルを安全に転送できます。

- [セキュア シェルバージョン2 サポートに関する情報 \(1 ページ\)](#)
- [セキュア シェルバージョン2 サポートの設定方法 \(7 ページ\)](#)
- [セキュア シェルバージョン2 サポートの設定例 \(21 ページ\)](#)
- [セキュア シェルバージョン2 サポートの追加情報 \(24 ページ\)](#)
- [セキュア シェルバージョン2 サポートの機能履歴 \(25 ページ\)](#)

## セキュア シェルバージョン2 サポートに関する情報

### SSH バージョン2

セキュア シェルバージョン2 サポート機能で、SSH バージョン2 を設定できます。

SSH バージョン2 サーバの設定は、SSH バージョン1 の設定と同様です。 `ip ssh version` コマンドは、設定する SSH バージョンを定義します。このコマンドを設定しない場合、デフォルトで SSH は互換モードで実行されます。バージョン1 とバージョン2 両方の接続が利用できます。



(注) SSH バージョン1 は、標準として定義されていないプロトコルです。未定義のプロトコル (バージョン1) にデバイスがフォールバックしないようにするには、 `ip ssh version` コマンドを使用してバージョン2 を指定する必要があります。

**ip ssh rsa keypair-name** コマンドを使用すると、設定した Rivest、Shamir、および Adleman (RSA) キーを使用して SSH 接続を実行できます。すでに、SSH は生成済みの最初の RSA キーにリンクされています (つまり、最初の RSA キーペアが生成された時点で SSH はイネーブルになっています)。この動作は存在していますが、**ip ssh rsa keypair-name** コマンドを使用してこの動作を行わないようにすることができます。**ip ssh rsa keypair-name** コマンドをキーペアの名前を指定して設定すると、SSH は、キーペアが存在する場合に有効になるか、キーペアを後で作成する場合は後から有効になります。このコマンドを使用して SSH をイネーブルにする場合、Cisco ソフトウェアの SSH バージョン1では必要な、ホスト名とドメイン名を設定を設定する必要はありません。



(注) ログインバナーは SSH バージョン2でサポートされますが、セキュア シェルバージョン1ではサポートされません。

## セキュア シェルバージョン2のRSAキーに関する機能拡張

Cisco SSH バージョン2は、キーボードインタラクティブ認証方式およびパスワードベースの認証方式をサポートしています。RSA キーの SSH バージョン2 拡張機能は、クライアントとサーバ向けの RSA ベースの公開キー認証もサポートしています。

ユーザ認証：RSA ベースのユーザ認証では、各ユーザに関連付けられている秘密キー/公開キーのペアを認証に使用します。ユーザは秘密キー/公開キーのペアをクライアントで生成し、公開キーを Cisco SSH サーバで設定して、認証を完了します。

クレデンシャルの確立を試行する SSH ユーザは、秘密キーを使用して暗号化された署名を提示します。署名とユーザの公開キーは、認証のために SSH サーバに送信されます。SSH サーバでは、ユーザから提示された公開キーに対してハッシュを計算します。ハッシュは、サーバに一致するエントリがあるかどうかを判断するために使用されます。一致が見つかった場合、RSA ベースのメッセージ検証が公開キーを使用して実行されます。その結果、暗号化されたシグニチャに基づいて、ユーザのアクセスは認証されるか拒否されます。

サーバ認証：SSH セッションの確立中に、Cisco SSH クライアントは、キー交換フェーズ中に使用できるサーバ ホスト キーを使用して、SSH サーバを認証します。SSH サーバキーは、SSH サーバの識別に使用されます。これらのキーはSSHがイネーブルになるときに作成され、クライアント側で設定する必要があります。

サーバ認証の場合、Cisco SSH クライアントが各サーバにホスト キーを割り当てる必要があります。クライアントがサーバとの間で SSH セッションを確立しようとする時、クライアントはキー交換メッセージの一部として、サーバの署名を受信します。厳密なホストキーのチェック フラグがクライアント側でイネーブルの場合、そのサーバに対応するホスト キー エントリがあるかどうかクライアントで確認されます。一致が見つかったら、クライアントはサーバホストキーを使用して署名の検証を試行します。サーバの認証に成功すると、セッションの確立処理は続行します。失敗すると、処理は終了し、「Server Authentication Failed」というメッセージが表示されます。



(注) 公開キーをサーバで格納する際、メモリを使用します。したがって、SSHサーバで設定できる公開キーの数は、1 ユーザに最大 2 つの公開キーを作成した場合 10 ユーザ分に限られます。



(注) シスコサーバは RSA ベースのユーザ認証をサポートしていますが、シスコクライアントは認証方式として公開キーを提案できません。RSA ベースの認証に対するオープンな SSH クライアントからの要求を Cisco サーバが受信した場合、サーバは認証要求を受け入れます。



(注) サーバ認証の場合、サーバの RSA 公開キーを手動で設定し、Cisco SSH クライアント側で **ip ssh stricthostkeycheck** コマンドを設定します。

## SNMP トラップ生成

ご使用のリリースに応じて、簡易ネットワーク管理プロトコル (SNMP) トラップは、トラップが有効で SNMP デバッグがオンになっている場合、SSH セッションが終了した際に自動的に生成されます。SNMP トラップの有効化に関する情報については、『SNMP Configuration Guide』の「Configuring SNMP Support」モジュールを参照してください。



(注) **snmp-server host** コマンドを設定する場合、IP アドレスは、SSH (telnet) クライアントがあり、SSH サーバへの IP 接続が可能な PC のアドレスにする必要があります。

また、**debug snmp packet** コマンドを使用して SNMP デバッグを有効にし、トラップを表示する必要があります。トラップ情報には、送信バイト数や SSH セッションで使用されたプロトコルなどの情報が含まれます。

次の例では、設定済みの SNMP トラップを示します。トラップ通知は、SSH セッションが終了すると自動的に生成されます。この例の a、b、c、d は SSH クライアントの IP アドレスです。

```
snmp-server  
snmp-server host a.b.c.d public tty
```

次に、**debug snmp packet** コマンドの出力例を示します。出力には、SSH セッションの SNMP トラップ情報が含まれます。

```
Switch# debug snmp packet  
  
SNMP packet debugging is on  
Device1# ssh -l lab 10.0.0.2  
Password:
```

```
Switch# exit

[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Switch#
```

## SSH キーボードインタラクティブ認証

SSH キーボードインタラクティブ認証機能は、SSH での汎用メッセージ認証とも呼ばれ、異なる種類の認証メカニズムを実装するために使用できる方式です。基本的に、現在サポートされている、ユーザの入力のみが必要な認証方式はすべて、この機能で実行することができます。この機能は自動的にイネーブルになります。

次の方式がサポートされています。

- Password
- サーバが送信するチャレンジに応答する番号またはストリングを印刷する SecurID およびハードウェア トークン
- プラグイン可能な認証モジュール (PAM)
- S/KEY (およびその他の使い捨てキー)

### 例：クライアント側のデバッグの有効化

次の例では、クライアント側のデバッグがオンになっており、プロンプトの最大数が 6 (SSH キーボードインタラクティブ認証方式のために 3 つ、パスワード認証方式のために 3 つ) になっています。

```
Password:
Password:
Password:
Password:
Password:
Password: cisco123
Last login: Tue Dec 6 13:15:21 2005 from 10.76.248.213
user1@courier:~> exit
logout
[Connection to 10.76.248.200 closed by foreign host]
Device1# debug ip ssh client

SSH Client debugging is on

Device1# ssh -l lab 10.1.1.3
```

```

Password:
*Nov 17 12:50:53.199: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version exchange successful
*Nov 17 12:50:53.203: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.335: SSH CLIENT0: key exchange successful and encryption on
*Nov 17 12:50:53.335: SSH2 CLIENT 0: using method keyboard-interactive
Password:
Password:
Password:
*Nov 17 12:51:01.887: SSH2 CLIENT 0: using method password authentication
Password:
Password: lab
Device2>

*Nov 17 12:51:11.407: SSH2 CLIENT 0: SSH2_MSG_USERAUTH_SUCCESS message received
*Nov 17 12:51:11.407: SSH CLIENT0: user authenticated
*Nov 17 12:51:11.407: SSH2 CLIENT 0: pty-req request sent
*Nov 17 12:51:11.411: SSH2 CLIENT 0: shell request sent
*Nov 17 12:51:11.411: SSH CLIENT0: session open

```

## 例：ブランクパスワードの変更による ChPass の有効化

次の例では、ChPass 機能が有効になっており、SSH キーボードインタラクティブ認証方式を使用してブランクパスワードが変更されています。TACACS+ アクセスコントロールサーバ (ACS) は、バックエンド AAA サーバとして使用されています。

```

Device1# ssh -l cisco 10.1.1.3

Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

```

## 例：ChPass の有効化および初回ログインでのパスワード変更

次の例では、ChPass 機能が有効になっており、TACACS+ ACS はバックエンドサーバとして使用されています。パスワードは、SSH キーボードインタラクティブ認証方式を使用して最初のログインで変更されています。

```

Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

```

## 例：ChPassの有効化および3回ログインした後のパスワードの失効

```
Password:cisco1
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco12
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Device2>
```

## 例：ChPassの有効化および3回ログインした後のパスワードの失効

次の例では、ChPass機能が有効になっており、TACACS+ ACSはバックエンドAAAサーバとして使用されています。パスワードは、SSHキーボードインタラクティブ認証方式を使用して3回ログインした後に期限切れになります。

```
Device# ssh -l cisco. 10.1.1.3

Password: cisco

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2>
```

# セキュア シェルバージョン2 サポートの設定方法

## ホスト名およびドメイン名を使用した SSH バージョン2 のデバイス設定

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例：  Device> enable	特権 EXEC モードを有効にします。  • パスワードを入力します（要求された場合）。
ステップ 2	<b>configure terminal</b> 例：  Device# configure terminal	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。
ステップ 3	<b>hostname name</b> 例：  Device(config)# hostname cisco7200	デバイスのホスト名を設定します。
ステップ 4	<b>ip domain-name name</b> 例：  cisco7200(config)# ip domain-name example.com	デバイスのドメイン名を設定します。
ステップ 5	<b>crypto key generate rsa</b> 例：  cisco7200(config)# crypto key generate rsa	ローカルおよびリモート認証用に SSH サーバをイネーブルにします。
ステップ 6	<b>ip ssh [time-out seconds   authentication-retries integer]</b> 例：  cisco7200(config)# ip ssh time-out 120	(任意) デバイス上で SSH 制御変数を設定します。
ステップ 7	<b>ip ssh version [1   2]</b> 例：	(任意) デバイスで実行する SSH のバージョンを指定します。

	コマンドまたはアクション	目的
	<code>cisco7200(config)# ip ssh version 1</code>	
ステップ 8	<b>exit</b> 例 : <code>cisco7200(config)# exit</code>	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 モードを開始します。 <ul style="list-style-type: none"> <li>デフォルト ホストに戻るには、<b>no hostname</b> コマンドを使用します。</li> </ul>

## RSA キー ペアを使用した SSH バージョン2のデバイス設定

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例 : <code>Device&gt; enable</code>	特権 EXEC モードを有効にします。 <ul style="list-style-type: none"> <li>パスワードを入力します (要求された場合)。</li> </ul>
ステップ 2	<b>configure terminal</b> 例 : <code>Device# configure terminal</code>	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。
ステップ 3	<b>ip ssh rsa keypair-name keypair-name</b> 例 : <code>Device(config)# ip ssh rsa keypair-name sshkeys</code>	SSH に使用する RSA キー ペアを指定します。 (注) シスコ デバイスには複数の RSA キー ペアを設定できません。
ステップ 4	<b>crypto key generate rsa usage-keys label key-label modulus modulus-size</b> 例 : <code>Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768</code>	デバイスでローカルおよびリモート認証を行う SSH サーバを有効にします。 <ul style="list-style-type: none"> <li>SSH バージョン2 では、絶対サイズは 768 ビット以上である必要があります。</li> </ul> (注) RSA キー ペアを削除するには、 <b>crypto key zeroize rsa</b> コマンドを使用します。RSA キー ペアを削除すると、SSH サーバは自動的に無効になります。



	コマンドまたはアクション	目的
ステップ 5	<b>ip ssh [time-out <i>seconds</i>   authentication-retries <i>integer</i>]</b> 例 : Device(config)# ip ssh time-out 12	デバイス上で SSH 制御変数を設定します。
ステップ 6	<b>ip ssh version 2</b> 例 : Device(config)# ip ssh version 2	デバイスで実行する SSH のバージョンを指定します。
ステップ 7	<b>exit</b> 例 : Device(config)# exit	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 モードを開始します。

## RSA ベースのユーザ認証を実行するための Cisco SSH サーバの設定

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例 : Device> enable	特権 EXEC モードを有効にします。 <ul style="list-style-type: none"><li>パスワードを入力します (要求された場合)。</li></ul>
ステップ 2	<b>configure terminal</b> 例 : Device# configure terminal	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。
ステップ 3	<b>hostname <i>name</i></b> 例 : Device(config)# hostname host1	ホスト名を指定します。
ステップ 4	<b>ip domain-name <i>name</i></b> 例 : host1(config)# ip domain-name name1	Cisco ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。
ステップ 5	<b>crypto key generate rsa</b> 例 :	RSA キー ペアを生成します。

	コマンドまたはアクション	目的
	<code>host1(config)# crypto key generate rsa</code>	
ステップ 6	<b>ip ssh pubkey-chain</b> 例 : <code>host1(config)# ip ssh pubkey-chain</code>	SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA キーを設定し、公開キーコンフィギュレーションモードを開始します。 <ul style="list-style-type: none"> <li>サーバに保存されている RSA 公開キーが、クライアントに保存されている公開キーと秘密キーのペアを使用して検証されると、ユーザ認証は成功です。</li> </ul>
ステップ 7	<b>username <i>username</i></b> 例 : <code>host1(conf-ssh-pubkey)# username user1</code>	SSH ユーザ名を設定し、公開キーユーザコンフィギュレーションモードを開始します。
ステップ 8	<b>key-string</b> 例 : <code>host1(conf-ssh-pubkey-user)# key-string</code>	リモートピアの RSA 公開キーを指定し、公開キーデータコンフィギュレーションモードを開始します。 (注) オープン SSH クライアントから (言い換えると <code>.ssh/id_rsa.pub</code> ファイルから) 公開キー値を取得できます。
ステップ 9	<b>key-hash <i>key-type key-name</i></b> 例 : <code>host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1</code>	(任意) SSH キータイプとバージョンを指定します。 <ul style="list-style-type: none"> <li>秘密キー/公開キーペアの設定では、キータイプを <code>ssh-rsa</code> にする必要があります。</li> <li><b>key-string</b> コマンドが設定されている場合に限りこの手順は任意です。</li> <li><b>key-string</b> コマンドと <b>key-hash</b> コマンドのいずれかを設定する必要があります。</li> </ul>

	コマンドまたはアクション	目的
		(注) 公開キー ストリングのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別のシスコデバイスからハッシュ値をコピーすることもできます。初めて公開キーデータを入力する場合、 <b>key-string</b> コマンドを使用して公開キーデータを入力することを推奨します。
ステップ 10	<b>end</b> 例：  host1(conf-ssh-pubkey-data)# end	公開キーデータ 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。  • デフォルト ホストに戻るには、 <b>no hostname</b> コマンドを使用します。

## RSA ベースのサーバ認証を実行するための Cisco IOS SSH サーバの設定

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例：  Device> enable	特権 EXEC モードを有効にします。  • パスワードを入力します（要求された場合）。
ステップ 2	<b>configure terminal</b> 例：  Device# configure terminal	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。
ステップ 3	<b>hostname name</b> 例：  Device(config)# hostname host1	ホスト名を指定します。
ステップ 4	<b>ip domain-name name</b> 例：  host1(config)# ip domain-name name1	Cisco ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。

	コマンドまたはアクション	目的
ステップ 5	<b>crypto key generate rsa</b> 例：  host1(config)# crypto key generate rsa	RSA キー ペアを生成します。
ステップ 6	<b>ip ssh pubkey-chain</b> 例：  host1(config)# ip ssh pubkey-chain	SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA キーを設定し、公開キー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	<b>server server-name</b> 例：  host1(conf-ssh-pubkey)# server server1	デバイスでの公開キー認証について SSH サーバを有効にし、公開キーサーバコンフィギュレーションモードを開始します。
ステップ 8	<b>key-string</b> 例：  host1(conf-ssh-pubkey-server)# key-string	リモートピアの RSA 公開キーを指定し、公開キーデータコンフィギュレーションモードを開始します。  (注) オープン SSH クライアントから（言い換えると .ssh/id_rsa.pub ファイルから）公開キー値を取得できます。
ステップ 9	<b>exit</b> 例：  host1(conf-ssh-pubkey-data)# exit	公開キーデータコンフィギュレーションモードを終了し、公開キーサーバコンフィギュレーションモードを開始します。
ステップ 10	<b>key-hash key-type key-name</b> 例：  host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1	(任意) SSH キータイプとバージョンを指定します。  <ul style="list-style-type: none"> <li>• 秘密キー/公開キーペアの設定では、キータイプを ssh-rsa にする必要があります。</li> <li>• <b>key-string</b> コマンドが設定されている場合に限りこの手順は任意です。</li> <li>• <b>key-string</b> コマンドと <b>key-hash</b> コマンドのいずれかを設定する必要があります。</li> </ul>

	コマンドまたはアクション	目的
		(注) 公開キー ストリングのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別のシスコデバイスからハッシュ値をコピーすることもできます。初めて公開キーデータを入力する場合、 <b>key-string</b> コマンドを使用して公開キーデータを入力することを推奨します。
ステップ 11	<b>end</b> 例：  host1(conf-ssh-pubkey-server)# end	公開キーサーバ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 12	<b>configure terminal</b> 例：  host1# configure terminal	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。
ステップ 13	<b>ip ssh stricthostkeycheck</b> 例：  host1(config)# ip ssh stricthostkeycheck	サーバ認証が実行されることを確認します。  <ul style="list-style-type: none"> <li>• 障害が発生すると、接続は終了します。</li> <li>• デフォルトホストに戻るには、<b>no hostname</b> コマンドを使用します。</li> </ul>

## リモート デバイスとの暗号化セッションの開始



- (注) 接続するデバイスは、シスコ ソフトウェアでサポートされる暗号化アルゴリズムを備えたセキュアシェル (SSH) サーバをサポートしている必要があります。また、デバイスを有効にする必要はありません。SSH はディセーブルモードで実行できます。

## 手順

	コマンドまたはアクション	目的
ステップ 1	<pre>ssh [-v {1   2}   -c {aes128-ctr   aes192-ctr   aes256-ctr   aes128-cbc   3des   aes192-cbc   aes256-cbc}   -l user-id   -l user-id:vrf-name number ip-address ip-address   -l user-id:rotary number ip-address   -m {hmac-md5-128   hmac-md5-96   hmac-sha1-160   hmac-sha1-96}   -o numberofpasswordprompts n   -p port-num] {ip-addr   hostname} [command   -vrf]</pre> <p>例 :</p> <pre>Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24</pre>	リモート ネットワーク デバイスとの暗号化されたセッションを開始します。

## SSH サーバでの Secure Copy Protocol のイネーブル化



- (注) 次のタスクでは、SCP のサーバ側機能を設定します。このタスクは、デバイスでリモートのワークステーションからファイルを安全にコピーできる一般的な設定を示しています。

## 手順

	コマンドまたはアクション	目的
ステップ 1	<p><b>enable</b></p> <p>例 :</p> <pre>Device&gt; enable</pre>	特権 EXEC モードを有効にします。 <ul style="list-style-type: none"> <li>パスワードを入力します (要求された場合)。</li> </ul>
ステップ 2	<p><b>configure terminal</b></p> <p>例 :</p> <pre>Device# configure terminal</pre>	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。
ステップ 3	<p><b>aaa new-model</b></p> <p>例 :</p> <pre>Device(config)# aaa new-model</pre>	AAA アクセス コントロール モデルをイネーブルにします。

	コマンドまたはアクション	目的
ステップ 4	<b>aaa authentication login default local</b> 例 :  Device(config)# aaa authentication login default local	認証時にローカルのユーザ名データベースを使用するように、ログイン時の AAA 認証を設定します。
ステップ 5	<b>aaa authorization exec defaultlocal</b> 例 :  Device(config)# aaa authorization exec default local	ユーザアクセスを制限するパラメータをネットワークに設定します。許可を実行し、ユーザ ID で EXEC シェルの実行を許可するかどうかを定義します。その後、システムで許可にローカルデータベースを使用する必要があることを指定します。
ステップ 6	<b>username name privilege privilege-level password password</b> 例 :  Device(config)# username samplename privilege 15 password password1	ユーザ名ベースの認証システムを確立し、ユーザ名、権限レベル、および非暗号化パスワードを指定します。  (注) <i>privilege-level</i> 引数の最小値は 15 です。権限レベルが 15 未満の場合、接続が切断されます。
ステップ 7	<b>ip ssh time-out seconds</b> 例 :  Device(config)# ip ssh time-out 120	デバイスが SSH クライアントの応答を待つ時間間隔を、秒単位で設定します。
ステップ 8	<b>ip ssh authentication-retries 整数</b> 例 :  Device(config)# ip ssh authentication-retries 3	インターフェイスのリセット後、認証を試行する回数を設定します。
ステップ 9	<b>ip scp server enable</b> 例 :  Device(config)# ip scp server enable	デバイスで、リモートワークステーションから安全にファイルをコピーできるようにします。
ステップ 10	<b>exit</b> 例 :  Device(config)# exit	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 11	<b>debug ip scp</b> 例 :	(任意) SCP 認証の問題に関する診断情報を提供します。

	コマンドまたはアクション	目的
	Device# debug ip scp	

## セキュア シェル接続のステータスの確認

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例： Device> enable	特権 EXEC モードを有効にします。  • パスワードを入力します（要求された場合）。
ステップ 2	<b>show ssh</b> 例： Device# show ssh	SSH サーバ接続のステータスを表示します。
ステップ 3	<b>exit</b> 例： Device# exit	特権 EXEC モードを終了し、ユーザ EXEC モードに戻ります。

次の **show ssh** コマンドの出力例には、バージョン 1 およびバージョン 2 接続の複数の SSH バージョン 1 およびバージョン 2 接続のステータスが表示されています。

```
-----
Device# show ssh

Connection      Version Encryption      State                Username
0                1.5      3DES              Session started     lab
Connection Version Mode Encryption      Hmac                State
Username
1                2.0      IN    aes128-cbc    hmac-md5    Session started     lab
1                2.0      OUT   aes128-cbc    hmac-md5    Session started     lab
-----
```

次の **show ssh** コマンドの出力例には、バージョン 2 接続（バージョン 1 接続なし）の複数の SSH バージョン 2 およびバージョン 1 接続のステータスが表示されています。

```
-----
Device# show ssh

Connection Version Mode Encryption      Hmac                State
Username
1                2.0      IN    aes128-cbc    hmac-md5    Session started     lab
1                2.0      OUT   aes128-cbc    hmac-md5    Session started     lab
-----
```



```
%No SSHv1 server connections running.
```

次の **show ssh** コマンドの出力例には、バージョン2接続（バージョン1接続なし）の複数の SSH バージョン1 およびバージョン2接続のステータスが表示されています。

```
-----
Device# show ssh

Connection      Version Encryption      State                Username
0               1.5          3DES              Session started     lab
%No SSHv2 server connections running.
-----
```

## セキュアシェルステータスの確認

### 手順

	コマンドまたはアクション	目的
ステップ1	<b>enable</b> 例： Device> enable	特権 EXEC モードを有効にします。  • パスワードを入力します（要求された場合）。
ステップ2	<b>show ip ssh</b> 例： Device# show ip ssh	SSH のバージョンおよび設定データを表示します。
ステップ3	<b>exit</b> 例： Device# exit	特権 EXEC モードを終了し、ユーザ EXEC モードに戻ります。

### 例

次の **show ip ssh** コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン1 およびバージョン2接続の認証の再試行回数が表示されています。

```
-----
Device# show ip ssh

SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
-----
```

次の **show ip ssh** コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン2接続（バージョン1接続なし）の認証の再試行回数が表示されています。

```
-----
Device# show ip ssh

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
-----
```

次の **show ip ssh** コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン1接続（バージョン2接続なし）の認証の再試行回数が表示されています。

```
-----
Device# show ip ssh

3d06h: %SYS-5-CONFIG_I: Configured from console by console
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
-----
```

## セキュア シェルバージョン2のモニタリングと維持

### 手順

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例： Device> enable	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	<b>debug ip ssh</b> 例： Device# debug ip ssh	SSH のデバッグを有効にします。
ステップ 3	<b>debug snmp packet</b> 例： Device# debug snmp packet	デバイスによって送受信されたすべての SNMP パケットのデバッグを有効にします。

### 例

次の **debug ip ssh** コマンドの出力例は、接続が SSH バージョン2接続であることを示します。

```
Device# debug ip ssh

00:33:55: SSH1: starting SSH control process
00:33:55: SSH1: sent protocol version id SSH-1.99-Cisco-1.25
00:33:55: SSH1: protocol version id is - SSH-2.0-OpenSSH_2.5.2p2
00:33:55: SSH2 1: send: len 280 (includes padlen 4)
00:33:55: SSH2 1: SSH2_MSG_KEXINIT sent
00:33:55: SSH2 1: ssh_receive: 536 bytes received
00:33:55: SSH2 1: input: packet len 632
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: ssh_receive: 96 bytes received
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: input: padlen 11
00:33:55: SSH2 1: received packet type 20
00:33:55: SSH2 1: SSH2_MSG_KEXINIT received
00:33:55: SSH2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
00:33:55: SSH2 1: newkeys: mode 1
00:33:55: SSH2 1: SSH2_MSG_NEWKEYS sent
00:33:55: SSH2 1: waiting for SSH2_MSG_NEWKEYS
00:33:55: SSH2 1: ssh_receive: 16 bytes received
00:33:55: SSH2 1: input: packet len 16
00:33:55: SSH2 1: partial packet 8, need 8, maclen 0
00:33:55: SSH2 1: input: padlen 10
00:33:55: SSH2 1: newkeys: mode 0
00:33:55: SSH2 1: received packet type 2100:33:55: SSH2 1: SSH2_MSG_NEWKEYS received
00:33:56: SSH2 1: ssh_receive: 48 bytes received
00:33:56: SSH2 1: input: packet len 32
00:33:56: SSH2 1: partial packet 16, need 16, maclen 16
00:33:56: SSH2 1: MAC #3 ok
00:33:56: SSH2 1: input: padlen 10
00:33:56: SSH2 1: received packet type 5
00:33:56: SSH2 1: send: len 32 (includes padlen 10)
00:33:56: SSH2 1: done calc MAC out #3
00:33:56: SSH2 1: ssh_receive: 64 bytes received
00:33:56: SSH2 1: input: packet len 48
00:33:56: SSH2 1: partial packet 16, need 32, maclen 16
00:33:56: SSH2 1: MAC #4 ok
00:33:56: SSH2 1: input: padlen 9
00:33:56: SSH2 1: received packet type 50
00:33:56: SSH2 1: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
```

```
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: pty-req request
00:34:04: SSH2 1: setting TTY - requested: height 24, width 80; set: height 24,
width 80
00:34:04: SSH2 1: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, maclen 16
00:34:04: SSH2 1: MAC #9 ok
00:34:04: SSH2 1: input: padlen 11
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: x11-req request
00:34:04: SSH2 1: ssh_receive: 48 bytes received
00:34:04: SSH2 1: input: packet len 32
00:34:04: SSH2 1: partial packet 16, need 16, maclen 16
00:34:04: SSH2 1: MAC #10 ok
00:34:04: SSH2 1: input: padlen 12
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: shell request
00:34:04: SSH2 1: shell message received
00:34:04: SSH2 1: starting shell for vty
00:34:04: SSH2 1: send: len 48 (includes padlen 18)
00:34:04: SSH2 1: done calc MAC out #7
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #11 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #8
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #12 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #9
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
```

```
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
```

## セキュア シェルバージョン2 サポートの設定例

### 例：セキュア シェルバージョン2 の設定

```
Device# configure terminal
Device(config)# ip ssh version 2
```

### 例：リモート デバイスでの暗号化セッションの開始

```
Device# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -l shaship 10.76.82.24
```

### 例：サーバ側 **SCP** の設定

次の例では、SCP のサーバ側機能の設定方法を示します。この例では、デバイスでの AAA 認証および許可も設定しています。この例では、ローカルに定義されたユーザ名とパスワードを使用します。

```
Device# configure terminal
Device(config)# aaa new-model
Device(config)# aaa authentication login default local
Device(config)# aaa authorization exec default local
Device(config)# username samplename privilege 15 password password1
Device(config)# ip ssh time-out 120
Device(config)# ip ssh authentication-retries 3
Device(config)# ip scp server enable
```

## 例 : SNMP トラップの設定

次の例では、設定済みの SNMP トラップを示します。トラップ通知は、SSH セッションが終了すると自動的に生成されます。この例の a、b、c、d は SSH クライアントの IP アドレスです。

```
snmp-server
snmp-server host a.b.c.d public tty
```

次に、**debug snmp packet** コマンドの出力例を示します。出力には、SSH セッションの SNMP トラップ情報が含まれます。

```
Device1# debug snmp packet
```

```
SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:
```

```
Device2# exit
```

```
[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Device1#
```

## 例 : SSH キーボードインタラクティブ認証

### 例 : SNMP のデバッグ

次に、**debug snmp packet** コマンドの出力例を示します。出力には、SSH セッションの SNMP トラップ情報が含まれます。

```
Device1# debug snmp packet
```

```
SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:
```

```
Device2# exit
```

```
[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
```

```

ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Device1#

```

## 例：SSH のデバッグの強化

次に、**debug ip ssh detail** コマンドの出力例を示します。出力には、SSH プロトコルとチャネル要求に関するデバッグ情報が含まれます。

```

Device# debug ip ssh detail

00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2:kex: client->server enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally

```

次に、**debug ip ssh packet** コマンドの出力例を示します。出力には、SSH パケットに関するデバッグ情報が含まれます。

```

Device# debug ip ssh packet

00:05:43: SSH2 0: send:packet of length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0

```

```

00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok

```

## セキュア シェルバージョン2サポートの追加情報

### 関連資料

関連項目	マニュアルタイトル
この章で使用するコマンドの完全な構文および使用方法の詳細。	<i>Consolidated Platform Command Reference, Cisco IOS Release 15.2(7)Ex (Catalyst Micro Switches)</i>

### 標準

標準	タイトル
IETF Secure Shell Version 2 Draft 規格	<a href="#">Internet Engineering Task Force の Web サイト</a>

### シスコのテクニカルサポート

説明	リンク
右の URL にアクセスして、シスコのテクニカルサポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	<a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a>



## セキュアシェルバージョン2 サポートの機能履歴

次の表に、このモジュールで説明する機能のリリースおよび関連情報を示します。

これらの機能は、特に明記されていない限り、導入されたリリース以降のすべてのリリースで使用できます。

リリース	機能	機能情報
Cisco IOS Release 15.2(7)E3k	セキュア シェルバージョン2 サポート	セキュア シェルバージョン2 サポート機能を使用して、セキュア シェル (SSH) バージョン2 を設定できます (SSH バージョン1 のサポートは、以前の Cisco IOS ソフトウェアリリースで実装されていました)。SSH は、信頼性の高いトランスポート層の上部で実行され、強力な認証機能と暗号化機能を提供します。SSH バージョン2 は、AES カウンタベース暗号化モードもサポートします。

Cisco Feature Navigator を使用すると、プラットフォームおよびソフトウェアイメージのサポート情報を検索できます。Cisco Feature Navigator には、<http://www.cisco.com/go/cfn> [英語] からアクセスします。

