



SSH バージョン 2

- [セキュアシェルバージョン 2 の機能履歴, on page 1](#)
- [セキュアシェルバージョン 2 とは, on page 2](#)
- [SSHv2 の拡張機能, on page 2](#)
- [RSA キー認証, on page 3](#)
- [SSH Ed25519 暗号化, on page 3](#)
- [セキュアシェルバージョン 2 の前提条件, on page 4](#)
- [セキュアシェルバージョン 2 の制約事項, on page 5](#)
- [設定時の注意事項, on page 5](#)
- [セキュアシェルバージョン 2 の設定方法, on page 6](#)
- [セキュアシェルバージョン 2 の設定例, on page 19](#)

セキュアシェルバージョン 2 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情報とプラットフォームサポート情報を示します。

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	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 26.2.1ea	SSH バージョン 2: SSH バージョン 2 の機能サポートが導入されました。	Cisco C9550 シリーズ スマートスイッチ

リリース	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 17.18.1	SSH バージョン 2: SSH バージョン 2 は、非信頼ネットワークで強力な暗号化と堅牢な認証による、セキュアなリモートアクセス、コマンド実行、およびファイル転送（SCP）を提供します。	Cisco C9350 シリーズ スマートスイッチ Cisco C9610 シリーズ スマートスイッチ

セキュアシェルバージョン 2 とは

セキュアシェルバージョン 2（SSH）は、非信頼ネットワークを介したセキュアなリモートアクセスとコマンド実行を可能にするネットワークプロトコルです。TCP など、信頼性の高いトランスポート層に強力な認証機能と暗号化機能を提供します。SSH には、セキュアなファイル転送用のセキュアコピープロトコル（SCP）も含まれます。

SSHv2 の拡張機能

- **Virtual Routing and Forwarding (VRF)-Aware SSH:** `ssh` コマンドは、SSH クライアント側の機能に VRF 認識を追加するように拡張されました。これにより、クライアント内の VRF インスタンス名は、正しいルーティングテーブルを検索して接続を確立するための IP アドレスを提供できます。



Note VRF-Aware SSH のサポートは、ソフトウェアリリースによって異なります。

- **Diffie-Hellman (DH) グループ交換のサポート:** Cisco SSH 実装では、従来 768 ビットのモジュラスが使用されていました。DH グループ 14（2048 ビット）およびグループ 16（4096 ビット）暗号化アプリケーションに対応するため、より大きなキーサイズの必要性が高まり、優先 DH グループを確立するクライアントとサーバー間のメッセージ交換が必要になっています。`ip ssh dh min size` コマンドは、SSH サーバー上の最小モジュラスサイズを設定します。
- **デバッグの機能拡張:** `debug ip ssh` コマンドは、デバッグを簡素化するために機能拡張されました。以前は、SSH 関連のすべてのデバッグメッセージを出力していました。現在では、キーワードを使用してコマンドを設定し、メッセージを特定の情報に制限できるようになりました。

RSA キー認証

ユーザー認証: RSA ベースのユーザー認証は、各ユーザーに関連付けられている秘密キー/公開キーのペアを使用します。ユーザーはこのキーのペアをクライアントで生成し、公開キーを Cisco SSH サーバーで設定します。Cisco Confidential 認証時に、SSH ユーザーは秘密キーを使用して暗号化された署名を提供します。この署名とユーザーの公開キーは、SSH サーバーに送信されます。サーバーは指定された公開キーに対してハッシュを計算し、一致するエントリが存在するかどうかを判断します。一致が見つかった場合、公開キーを使用した RSA ベースのメッセージ検証が実行され、暗号化された署名に基づいてユーザーが認証または拒否されます。

サーバー認証: Cisco SSH クライアントは、キー交換フェーズ中に交換されたサーバーのホストキーを使用して、セッションの確立時に SSH サーバーを認証します。SSH が有効なときに作成されるこれらのキーは、SSH サーバーを識別するために使用され、認証用にクライアントで設定する必要があります。クライアントが SSH セッションを確立しようとする、キー交換メッセージの一部として、サーバー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有効になっている場合、そのサーバーに対応するホストキーエントリがあるかどうかをクライアントが確認します。一致が見つかったら、クライアントはサーバーのホストキーを使用して署名の検証を試行します。サーバーの認証に成功すると、セッションの確立処理が続行されます。失敗すると、セッションは終了し、「Server Authentication Failed」というメッセージが表示されます。



Note

- サーバーに公開キーを保存すると、メモリを消費します。したがって、SSH サーバーで設定できる公開キーの数は、1 ユーザーに最大 2 つの公開キーを作成した場合 10 ユーザーに制限されます。
- シスコサーバーは、RSA ベースのユーザー認証をサポートしています。ただし、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 OpenSSH クライアントからの要求を Cisco サーバーが受信した場合、サーバーは認証要求を受け入れます。
- 適切なサーバー認証の場合、サーバーの 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。

SSH Ed25519 暗号化

Ed25519 は、次の機能を提供する公開キー署名システムです。

- 単一署名の高速検証
- 高速バッチ検証
- 高速署名

- 高速キー生成
- 高いセキュリティレベル
- フールプルーフのセッションキー
- 小さな署名（64 バイト）
- 小さなキー（32 バイト）

SSH およびスイッチアクセス

シェル（SSH）は、デバイスへのセキュアなリモート接続を提供し、Telnet よりも強力な暗号化と認証を提供します。このソフトウェアリリースは、SSH バージョン 2（SSHv2）をサポートします。SSH 機能は IPv4 および IPv6 で一貫しており、IPv6 トランスポート上でリモート IPv6 ノードとのセキュアで暗号化された接続を可能にします。

SNMP トラップ生成の Simple Network Management Protocol（SNMP）トラップは、ソフトウェアリリースに応じて、SSH セッションの終了時に自動生成されます。これらのトラップを有効にするには、次を実行します。

- SNMP トラップが有効になっていることを確認します。
- `debug snmp packet` コマンドを使用して SNMP デバッグを有効にし、トラップを表示します。
- `snmp-server host` コマンドを設定する場合は、SSH（または Telnet）クライアントをホストしている PC の IP アドレスを指定し、SSH サーバーへの IP 接続を確認します。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。

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、これによってユーザー入力のみを必要とするさまざまな認証メカニズムを実装できます。この機能は自動的に有効になります。サポートされている認証方式は次のとおりです。

- パスワード
- SecurID およびハードウェアトークン（サーバーのチャレンジに番号または文字列で応答する）
- 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）

セキュアシェルバージョン 2 の前提条件

- ご使用のデバイスに必要なイメージがロードされていることを確認します。SSH サーバーには、ソフトウェアリリースに応じた k9（Triple Data Encryption Standard（3DES））ソフトウェアイメージが必要です。

- SSH バージョン 2 をサポートする SSH リモートデバイスを使用してシスコデバイスに接続します。
- デバイスに認証、許可、アカウントティング（AAA）を設定します。SCP は AAA に依存して正しく機能するため、AAA を設定すると SSH サーバーで SCP が有効になります。

**Note**

SSH バージョン 2 サーバーおよびクライアントは、特定のシスコソフトウェアリリースでサポートされています。SSH クライアントは、SSH バージョン 1 と SSH バージョン 2 の両方のプロトコルをサポートし、ソフトウェアリリースに応じて k9 イメージで使用できます。

セキュアシェルバージョン 2 の制約事項

- SSH のサーバーとクライアントには、Triple Data Encryption Standard（3DES）ソフトウェアイメージが必要です。
- サポートされるアプリケーションは、実行シェル、リモートコマンドの実行、Secure Copy Protocol（SCP）のみです。
- Rivest、Shamir、および Adleman（RSA）キー生成は SSH サーバーに必要です。SSH クライアントでは RSA キーの生成は必要ありません。
- RSA キーペアのサイズは、768 ビット以上である必要があります。
- サポートされていない機能は、次のとおりです。
 - ポート フォワーディング。
 - Compression

設定時の注意事項

SSH バージョン 2 を設定するには、**ip ssh version** コマンドを使用します。このコマンドを設定しない場合、デフォルトで SSH は互換モードで動作します。SSH バージョン 1 と SSH バージョン 2 両方の接続が利用できます。

**Note**

SSH バージョン 1 は、標準化されたプロトコルではありません。デバイスがこの未定義のプロトコルに戻ら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 を指定します。

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。このコマンドを使用すると、以前のホスト名とドメイン

ン名の設定なしで SSH を有効にできます。SSH は、指定されたキーペアが存在するか、または後で生成される場合に有効になります。



Note 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、SSH 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

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 configure terminal

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3 hostname *name*

Example:

```
Device(config)# hostname catalyst9k
```

デバイスのホスト名を設定します。

Step 4 ip domain name *name*

Example:

```
catalyst9k(config)# ip domain name example.com
```

デバイスのドメイン名を設定します。

Step 5 crypto key generate rsa

Example:

```
catalyst9k(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。

Step 6 `ip ssh [time-out seconds | authentication-retries integer ]`

Example:

```
catalyst9k(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。

Step 7 `ip ssh version [1 | 2 ]`

Example:

```
catalyst9k(config)# exit
```

（任意）デバイスで実行する SSH のバージョンを指定します。

Step 8 `exit`

Example:

```
catalyst9k(config)# ip ssh version 1
```

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。

- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

RSA キー ペアを使用した SSH バージョン 2 のデバイス設定

Procedure

Step 1 `enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3 `ip ssh rsa keypair-name keypair-name`

Example:

```
Device(config)# ip ssh rsa keypair-name sshkeys
```

SSH に使用する RSA キー ペアを指定します。

Note

シスコ デバイスには複数の RSA キー ペアを設定できます。

Step 4 `crypto key generate rsa usage-keys label key-label modulus modulus-size`

Example:

```
Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768
```

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。

- 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。

Note

RSA キー ペアを削除するには、`crypto key zeroize rsa` 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。

Step 5 `ip ssh [time-out seconds | authentication-retries integer ]`

Example:

```
Device(config)# ip ssh time-out 12
```

デバイス上で SSH 制御変数を設定します。

Step 6 `ip ssh version 2`

Example:

```
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。

Step 7 `exit`

Example:

```
catalyst9k(config)# ip ssh version 1
```

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。

ユーザー認証を実行するための Cisco SSH サーバーの設定

Procedure

Step 1 `enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3 `hostname name`

Example:

```
Device(config)# hostname host1
```

ホスト名を指定します。

Step 4

`ip domain name name`

Example:

```
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。

Step 5

`crypto key generate rsa`

Example:

```
host1(config)# crypto key generate rsa
```

RSA キー ペアを生成します。

Step 6

`ip ssh pubkey-chain`

Example:

```
host1(config)# ip ssh pubkey-chain
```

SSH サーバー上のユーザーおよびサーバー認証用に公開キーを設定し、公開キー コンフィギュレーション モードを開始します。

- サーバーに保存されている公開キーが、クライアントに保存されている公開キーまたは秘密キーのペアを使用して検証されると、ユーザー認証は成功です。

Step 7

`username username`

Example:

```
host1(conf-ssh-pubkey)# username user1
```

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。

Step 8

`key-string`

Example:

```
host1(conf-ssh-pubkey-user)# key-string
```

リモートピアの公開キーを指定し、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。

Note

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。

Step 9

`exit`

Example:

```
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。

Step 10

`key-hash key-type key-name`

Example:

```
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1
or
host1(conf-ssh-pubkey-data)# key-hash ssh-ed25519 DFEAF10390E560AEA745CCBA53E044ED
```

(任意) SSH キー タイプとバージョンを指定します。

- 秘密キー/公開キーペアの設定では、キータイプを `ssh-rsa` または `ssh-ed25519` に指定できます。
- この手順は、`key-string` コマンドが設定されている場合にのみ任意です。
- `key-string` コマンドまたは `key-hash` コマンドのいずれかを設定する必要があります。

Note

公開キースtring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、`key-string` 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。

Step 11 `end`

Example:

```
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。

Step 12 `configure terminal`

Example:

```
host1# configure terminal
```

グローバル設定モードを開始します。

Step 13 `ip ssh stricthostkeycheck`

Example:

```
host1(config)# ip ssh stricthostkeycheck
```

サーバ認証が実行されることを確認します。

- 障害が発生すると、接続は終了します。
- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

Step 14 `end`

Example:

```
host1(config)# end
```

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。

サーバー認証を実行するための Cisco IOS SSH クライアントの設定

Procedure

Step 1

`enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

`configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3

`hostname name`

Example:

```
Device(config)# hostname host1
```

ホスト名を指定します。

Step 4

`ip domain name name`

Example:

```
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。

Step 5

`crypto key generate rsa`

Example:

```
host1(config)# crypto key generate rsa
```

RSA キー ペアを生成します。

Step 6

`ip ssh pubkey-chain`

Example:

```
host1(config)# ip ssh pubkey-chain
```

SSH サーバー上のユーザーおよびサーバー認証用に公開キーを設定し、公開キー コンフィギュレーション モードを開始します。

Step 7

`server server-name`

Example:

```
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。

Step 8 key-string**Example:**

```
host1(conf-ssh-pubkey-server)# key-string
```

リモートピアの公開キーを指定し、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。

Note

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。

Step 9 exit**Example:**

```
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。

Step 10 key-hash key-type key-name**Example:**

```
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1
or
host1(conf-ssh-pubkey-data)# key-hash ssh-ed25519 DFEAF10390E560AEA745CCBA53E044ED
```

（任意）SSH キー タイプとバージョンを指定します。

- 秘密キー/公開キーペアの設定では、キータイプを ssh-rsa または ssh-ed25519 に指定できます。
- この手順は、key-string コマンドが設定されている場合にのみ任意です。
- key-string コマンドまたは key-hash コマンドのいずれかを設定する必要があります。

Note

公開キースtring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、key-string 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。

Step 11 end**Example:**

```
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。

Step 12 configure terminal**Example:**

```
host1# configure terminal
```

グローバル設定モードを開始します。

Step 13 ip ssh stricthostkeycheck**Example:**

```
host1(config)# ip ssh stricthostkeycheck
```

サーバ認証が実行されることを確認します。

- 障害が発生すると、接続は終了します。
- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

Step 14 `end`**Example:**

```
host1(config)# end
```

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。

リモート デバイスとの暗号化セッションの開始

**Note**

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 SSH サーバーをサポートしている必要があります。デバイスで SSH サービスを明示的に有効にする必要はありません。SSH は、アクティブに実行されていない場合でも設定できます。

Procedure

Step 1 `enable`**Example:**

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `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]`**Example:**

```
Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24
```

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。

セキュア シェル接続のステータスの確認

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

show ssh

Example:

```
Device# show ssh
```

SSH サーバ接続のステータスを表示します。

Step 3

exit

Example:

```
Device# exit
```

特権 EXEC モードを終了し、ユーザー EXEC モードに戻ります。

例

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 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 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 1 接続（バージョン 2 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.
```

セキュアシェルバージョン 2 のステータスの確認

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

show ip ssh

Example:

```
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。

Step 3

exit

Example:

```
Device# exit
```

特権 EXEC モードを終了し、ユーザー 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

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

debug ip ssh

Example:

```
Device# debug ip ssh
```

SSH のデバッグを有効にします。

Step 3

debug snmp packet

Example:

```
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。

Example

What to do next

-

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 の設定例

例: セキュア シェルバージョン 2 の設定

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

例: セキュア シェルバージョン 1 および 2 の設定

```
Device> enable
Device# configure terminal
Device(config)# no ip ssh version
Device(config)# end
```

例: リモート デバイスでの暗号化セッションの開始

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

例: SNMP トラップの設定

次の例では、SNMP トラップの設定方法を示します。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 10.1.1.1 は SSH クライアントの IP アドレスです。

```
Device> enable
Device# configure terminal
Device(config)# snmp-server trap link switchover
Device(config)# snmp-server host 10.1.1.1 public tty
Device(config)# end
```

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]
```

例: SSH のデバッグの強化

```

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
Cisco Confidential
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
```


翻訳について

このドキュメントは、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。