ユーザガイド

Reflection for Secure IT
Client and Server for UNIX

Attachmate Reflection®

Attachmate

ユーザガイド

Reflection for Secure IT

Client and Server for UNIX

Version 8.0 SP 1

2014年4月23日作成

Attachmate Corporation
705 5th Avenue South
Seattle, WA 98104
USA
+1.206.217.7100
http://www.attachmate.com

目次

第 1 章

インストール

この章の内容

Reflection for Secure IT, Client and Server for UNIX は、コンピュータ間の安全な接続に使用します。Reflection for Secure IT を使用して、安全なファイル転送、コンピュータの安全な遠隔管理、およびネットワーク全体にわたるアプリケーショントラフィックの保護されたトンネリングを実現できます。

使用可能なプラットフォームとシステム要件の詳細は、技術ノート 1944 『http://support.attachmate.com/techdocs/1944.html』 を参照してください。

クライアント機能

Reflection for Secure IT のクライアントおよびサーバいずれの製品でも、以下の Secure Shell クライアント機能がインストールされます。

- ssh（Secure Shell クライアント）
- ssh2_config（クライアント構成ファイル）
- sftp（安全なファイル転送）
- scp（安全なファイルコピー）
- ssh-keygen（鍵生成ユーティリティ）
- ssh-agent（鍵エージェント）
- ssh-add（エージェントへの ID の追加）
- ssh-askpass（X11 パスフレーズユーティリティ）
- ssh-certtool（証明書管理ユーティリティ）
- ssh-certview（証明書表示ユーティリティ）

既定では、クライアント実行ファイルは /usr/bin にインストールされます (Linux 上では ssh-askpass は /usr/libexec にインストールされます)。グローバルクライアント構成ファイルは /etc/ssh2/ にインストールされます。

サーバ機能

Reflection for Secure IT サーバは、上記のすべてのクライアント機能に加えて以下の Secure Shell サーバ機能を含みます。

- sshd（Secure Shell デーモン）
- sshd2_config（サーバ構成ファイル）
- ホストの公開鍵/秘密鍵ペア（以下の注意を参照）
- sftp-server（サーバ側ファイル転送対応サブシステム）

既定では、sshd サーバは /usr/sbin にインストールされます。sftp-server は /usr/bin にインストールされます(Linux 上では sftp-server は /usr/libexec にインストールされます)。サーバ構成ファイルは /etc/ssh2 にインストールされます。

注意: サーバのインストールパッケージは、既存のホスト鍵ペアがすでに存在しているかどうかを確認します。ホスト鍵が見つからない場合は、パッケージが新しいホスト鍵ペアを作成し、サーバはこの鍵ペアをホスト認証に使用します。ホスト鍵がすでに /etc/ssh2 に存在している場合、Reflection for Secure IT はこの鍵を使用します。OpenSSH ホスト鍵が /etc/ssh に存在する場合、Reflection for Secure IT は鍵を適正な形式に移行してから、この移行後の鍵を使用します。

必要なパッケージ

次の表に、対応プラットフォームごとに必要なサーバインストールファイルの名前を示します。

注意: ここに記載したサーバパッケージは、クライアント機能とサーバ機能 『7ページ』の両方をインストールします。インストールパッケージ名の「server」を「client」で置き換えると、クライアント機能のみがインストールされます。

プラットフォーム	サーバインストールパッケージ
IBM AIX 5.3、6.1、7.1 (POWER)	rsit-server-<*build*>-powerpc-aix5.3.bff.Z
HP-UX 11i v2、v3 (Itanium)	rsit-server-<*build*>-ia64-hpux-11.23.depot.Z
HP-UX 11i v2 (PA-RISC)	rsit-server-<*build*>-parisc-hpux-11.23.depot.Z
Red Hat Enterprise Linux 5、6 (Intel 86)	rsit-server-<*build*>-i386-rhel.rpm
Red Hat Enterprise Linux 5、6 (x86-64)	rsit-server-<*build*>-x86_64-rhel.rpm
SUSE Linux Enterprise Server 10、11 (Intel x86)	rsit-server-<*build*>-i586-sles.rpm
SUSE Linux Enterprise Server 10、11 (x86_ 64)	rsit-server-<*build*>-x86_64-sles.rpm
SUSE Linux Enterprise Server 10 (zSeries 64)	rsit-server-<*build*>-s390x-sles.rpm
Oracle Solaris 9 (SPARC V9)	rsit-server-<*build*>-sparc-solaris.pkg.Z
Oracle Solaris 10 (SPARC V9)	rsit-server-<*build*>-sparc-solaris10.pkg.Z
Oracle Solaris 10 (Intel x86)	rsit-server-<*build*>-i386-solaris10.pkg.Z
Oracle Solaris 10 (x86-64)	rsit-server-<*build*>-x64-solaris10.pkg.Z
Oracle Solaris 11 (SPARC)	rsit-server-<*build*>-sparc-solaris11.tar.gz

以前のバージョンまたは他の既存の Secure Shell プログラムの置き換え

すでに Secure Shell クライアントまたはサーバを実行中のシステムにインストールする場合は、Reflection for Secure IT をインストールする前に、以前のバージョンをアンインストールする必要があります。これは、旧バージョンの Reflection for Secure IT、および F-Secure SSH、OpenSSH、などその他すべての Secure Shell 実装が対象となります。

Secure Shell を現在実行中のシステムにインストールするには

1 root としてログインします。

2 (サーバのみ) **sshd** サービスを停止します。

3 既存の Secure Shell 製品をアンインストールします。

4 (AIX のみ) 以前のバージョンをアンインストールするために **installp** を実行したディレクトリ内で `.toc` 隠しファイルの有無を確認します。このファイルが存在する場合は、同ファイルを削除するか、名前を変更します。

5 Reflection for Secure IT クライアントまたはサーバをインストールします。

6 公開鍵認証を使用している場合は、使用しているファイルおよびディレクトリが正しい権限を使用して構成されていることを確認してください。Reflection for Secure IT のこのリリースでは、バージョン 7.2 と比べて格段に向上したセキュリティレベルが必要とされます。ファイルとディレクトリの保護が十分でない場合、公開鍵の認証は失敗します。詳細については、「ファイルおよびディレクトリの権限 『135ページ』」を参照してください。

   注意: **StrictModes** の設定は、公開鍵認証で使用されるファイルおよびディレクトリに必要な保護レベルに影響を与えます。十分な安全レベルを確保するため、既定ではこの設定が有効になっています。この設定を無効にした場合でも、一部のファイルとディレクトリの権限は有効になります。

7 (オプション) 既定以外のクライアント/サーバ構成ファイルを設定していた場合は、構成ファイルのディレクトリ内に同ファイルのバックアップコピーが存在します(詳細については、以下の注意を参照)。既定以外の設定を新規構成ファイルに結合するには、これらのバックアップファイルを使用します。

注意:

- サーバのインストールパッケージは、既存のホスト鍵ペアがすでに存在しているかどうかを確認します。ホスト鍵が見つからない場合は、パッケージが新しいホスト鍵ペアを作成し、サーバはこの鍵ペアをホスト認証に使用します。ホスト鍵がすでに `/etc/ssh2` に存在している場合、Reflection for Secure IT はこの鍵を使用します。OpenSSH ホスト鍵が `/etc/ssh` に存在する場合、Reflection for Secure IT は鍵を適正な形式に移行してから、この移行後の鍵を使用します。
- 構成ファイルのバックアップがどのようにして作成されるかは、対応するオペレーティングシステムによって異なります。
  - AIX を除くすべてのプラットフォーム上では、既定のクライアント/サーバ構成ファイルに対して変更を加えた場合、アンインストール時にインストーラによってファイルのバックアップが実行されます(このバックアップに追加されるファイル拡張子はネイティブインストーラによって異なります)。
  - AIX 上では、アンインストール時にバックアップファイルは作成されません。代わりに、Reflection for Secure IT のインストール時に既定以外の構成ファイルが存在している場合にバックアップファイルが作成されます。
- 以前のバージョンの Reflection for Secure IT で作成された鍵ペアは、現行バージョンと互換性があるため、変換は必要ありません。
- **StrictModes** の既定値は、現在、クライアントとサーバの両方で「yes」に設定されています。
- `/etc/pam.d/ssh` が存在する場合、このファイルがバックアップされ、新規のファイルがその場所に配置されます。
- サブ構成ファイルが存在する場合、そのファイルはそのまま維持されます。

Linux でのインストールおよびアンインストール

Reflection for Secure IT を Linux にインストールするには

1 root としてログインします。

2 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

3 **rpm** を使用してパッケージをインストールします。

```
rpm -ivh package_name.rpm
```

たとえば、以下のようになります。

```
rpm -ivh rsit-server-8.0.0.999-x86_64-rhel.rpm
```

アンインストールするには

1 root としてログインします。

2 以下のいずれかのコマンドを入力します。

対象	使用コマンド
サーバ	`rpm -e --nodeps rsit-server`
クライアント	`rpm -e --nodeps rsit-client`

Sun Solaris でのインストールおよびアンインストール

Reflection for Secure IT を Solaris にインストールするには

1 root としてログインします。

2 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

3 **uncompress** を使用してパッケージを圧縮解除します。

```
uncompress package_name.pkg.Z
```

たとえば、以下のようになります。

```
uncompress rsit-client-7.2.0.999-sparc-solaris10.pkg.Z
```

4 **pkgadd** を使用してパッケージをインストールします。

```
pkgadd -d package_name.pkg
```

例えば、次のように入力します。

```
pkgadd -d rsit-client-7.2.0.999-sparc-solaris10.pkg
```

注意: Solaris 10 を実行中のシステムでは、ゾーンを使用して、単一の Solaris インスタンスを孤立した複数のアプリケーション環境へパーティショニングできます。ゾーン環境での Reflection for Secure IT のインストールについては、技術ノート 2254『http://support.attachmate.com/techdocs/2254.html』(英語) を参照してください。

アンインストールするには

1 root としてログインします。

2 **pkgrm** コマンドを使用してパッケージを削除します。

対象	使用コマンド
サーバ	`pkgrm RSITsshs`
クライアント	`pkgrm RSITsshc`

Oracle Solaris 11 でのインストールおよびアンインストール

Solaris 11では、Reflection for Secure IT のインストールに Image Packaging System (IPS) が使用されます。

Reflection for Secure IT を Solaris 11 にインストールするには

1 root としてログインします。

2 既存の SSH 製品がある場合はこれをアンインストールします。Reflection for Secure IT をアンインストールするには、以下のアンインストール手順を参照してください。Oracle SSH をアンインストールするには、**pkg uninstall** を使用してすべての SSH コンポーネントを削除します。次に例を示します。

```
pkg uninstall service/network/ssh
pkg uninstall gnu-tar //Required in global zone only
pkg uninstall network/ssh
pkg uninstall ssh-key
```

3 IPS リポジトリを作成し、発行元を「attachmate」に設定します。この例で示したリポジトリ名は推奨される名前であり、必ずしもこの名前を使用する必要はありません。ただし、発行元の名前は「attachmate」に設定する必要があります。次に例を示します。

```
pkgrepo create attachmate-repository
pkgrepo -s attachmate-repository set publisher/prefix=attachmate
```

4 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

5 パッケージを解凍します。

サーバ	`tar xvf rsit-server-<n.n.nn>-sparc-solaris11.tar.gz`
クライアント	`tar xvf rsit-client-<n.n.nn>-sparc-solaris11.tar.gz`

6 パッケージを公開します。パッケージのディレクトリ名とマニフェスト名はいずれも、サーバインストールとクライアントインストールで異なります。

サーバ	`pkgsend -s attachmate-repository publish -d pkgs sshdmanifest`
クライアント	`pkgsend -s attachmate-repository publish -d pkgc sshmanifest`

7 作成したリポジトリ (この例では「attachmate-repository」) の発行元を、構成済みの発行元に追加します。

```
pkg set-publisher -p attachmate-repository
```

8 パッケージをインストールします。

サーバ `pkg install RSITsshs`

クライアント `pkg install RSITsshc`

インストール場所の変更 (Solaris 11)

Reflection for Secure IT は `relocate.mog` という名前の **pkgmogrify** 入力ファイルをインストールします。このファイルは次の 2 つの構成オプションを使用した既定外の場所へのインストールに対応しています。

- **sysconfdir** を使用すると、構成ファイルと鍵 (デフォルトでは `/etc/ssh2` にインストール) を別の場所にインストールするように指定できます。
- **prefix** を使用すると、バイナリとマニュアルページ（デフォルトでは `/usr` にインストール）を別の場所にインストールするように指定できます。

既定外の場所にインストールするには

1 配布パッケージをダウンロードして解凍します。

2 以下に示す構文を使用して、解凍したファイルを含むディレクトリから **pkgmogrify** コマンドを実行します。

- **sysconfdir** と **prefix** のサンプルパスを、使用するインストールディレクトリで置き換えます。
- クライアントをインストールしている場合、**sshdmanifest** を **sshmanifest** で置き換え、**pkgs** を **pkgc** で置き換えます。

```
# pkgmogrify -D prefix=/opt/usr -D sysconfdir=/opt/etc/ssh2
-P pkgs/etc/rsit.conf sshdmanifest relocate.mog | pkgfmt >
sshdmanifest.relocate
```

注意: このコマンドによって、パッケージの公開時に使用される新しいマニフェストファイル (この例では `sshdmanifest.relocate`) が作成されます。(元の `sshdmanifest` ではなく) この新しいマニフェストを使用する必要があります。**pkgmogrify** コマンドを実行した後に既定を使用してインストールする場合、または別の再配置パスを使用する場合、まず解凍したファイルセットを削除してから再度 `tar xvf <pkg_name>.tar.gz` を実行して、すべてのファイルを元のバージョンに戻します。

3 前の手順で作成した新しいマニフェストを使用してパッケージを公開します (クライアントインストールの場合は「pkgs」を「pkgc」で置き換えます)。次に例を示します。

```
pkgsend -s attachmate-repository publish -d pkgs sshdmanifest.relocate
```

4 リポジトリを指定して、パッケージをインストールします(クライアントインストールの場合は「RSITsshs」を「RSITsshc」で置き換えます)。

```
pkg set-publisher -p attachmate-repository
pkg install RSITsshs
```

注意:

- 既定外の場所へインストールした後にバイナリおよびマニュアルページにアクセスできるようにするには、システムの PATH 変数および MANPATH 変数を変更します。
- インストールされた項目のうち、起動および停止スクリプト、暗号モジュール、および PKI クライアントディレクトリは移動されません。

グローバルゾーン以外へのインストール

グローバルゾーンで **pkg install** コマンドを実行する場合、パッケージはグローバルゾーンにのみインストールされ、その他のゾーンにはインストールされません。Reflection for Secure IT をグローバルゾーン以外にインストールするには、次の 2 つの方法があります。

- グローバルゾーンでリポジトリを作成し、グローバルゾーン以外でこのリポジトリを使用してインストールを実行します。グローバルゾーンに対する発行元の構成変更は、システムリポジトリを介してグローバルゾーン以外からも直ちに確認できます。
- グローバルゾーン以外でリポジトリを作成します。

注意: いずれの方法を使用する場合も、上記で説明したとおり、先に Oracle SSH パッケージを削除します。

システムリポジトリを使用してグローバルゾーン以外にインストールするには

1 グローバルゾーンでリポジトリを作成します。

2 次に示すように、グローバルゾーン以外から **pkg publisher** を使用して、発行元「attachmate」が使用可能であることを確認します。

```
# pkg publisher
PUBLISHER                    TYPE     STATUS P LOCATION
solaris        (syspub)      origin   online T <system-repository>
attachmate     (syspub)      origin   online F <system-repository>
```

3 ゾーン管理者として **pkg install** コマンドを実行します(クライアントインストールの場合は「RSITsshs」を「RSITsshc」で置き換えます)。

```
# pkg install RSITsshs
```

グローバルゾーン以外でリポジトリを使用してインストールするには

1 グローバルゾーンでリポジトリを作成した場合、発行元「attachmate」をグローバルゾーンで無効にします。

```
pkg set-publisher --disable attachmate
```

2 グローバルゾーン以外で root としてログインします。

3 次に示すように、**pkg publisher** を使用して、発行元「attachmate」が使用可能ではないことを確認します。

```
# pkg publisher
PUBLISHER                    TYPE     STATUS P LOCATION
solaris        (syspub)      origin   online T <system-repository>
```

4 グローバルゾーン以外でリポジトリを作成します。この例はサーバインストールを構成するものです。

```
# tar xvf rsit-server-8.0.1.99--sparc-solaris11.tar.gz
# pkgrepo create zone-repository
# pkgrepo -s zone-repository set publisher/prefix=attachmate
# pkgsend -s zone-repository publish -d pkgs sshdmanifest
# pkg set-publisher -p zone-repository
# pkg publisher
PUBLISHER                    TYPE     STATUS P LOCATION
solaris        (syspub)      origin   online T <system-repository>
attachmate                   origin   online F file:///export/zone-repository/
```

5 パッケージをインストールします。(クライアントインストールの場合は「RSITsshs」を「RSITsshc」で置き換えます)。

```
# pkg install RSITsshs
```

パッケージの更新

(パッケージリストを指定せずに) **pkg update** コマンドを使用すると、グローバルゾーンに対するすべてのゾーンの同期を維持できます。

```
#pkg update
```

パッケージリストを指定した場合、現在のゾーンでのみ更新が実行されます。次のコマンドをグローバルゾーンで実行すると、グローバルゾーンでのみ更新が実行されます。グローバルゾーン以外で実行すると、グローバルゾーン以外でのみ更新が実行されます。

```
#pkg update RSITsshs
```

Solaris 11 でのアンインストール

Reflection for Secure IT をアンインストールするには

1 root としてログインします。

2 **pkg uninstall** コマンドを使用してパッケージを削除します。

サーバ	`pkg uninstall RSITsshs`
クライアント	`pkg uninstall RSITsshc`

HP-UX でのインストールおよびアンインストール

Reflection for Secure IT を HP-UX にインストールするには

1 root としてログインします。

2 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

3 **uncompress** を使用してパッケージを圧縮解除します。

```
uncompress package_name.depot.Z
```

たとえば、以下のようになります。

```
uncompress rsit-client-7.2.0.999-ia64-hpux-11.23.depot.Z
```

4 **swinstall** を使用して、圧縮解除されたパッケージをインストールします。

```
swinstall -s full_path_and_package_name.depot RSIT
```

たとえば、以下のようになります。

```
swinstall -s /rsit/rsit-server-7.2.0.999-ia64-hpux-11.23.depot RSIT
```

アンインストールするには

1 root としてログインします。

2 (サーバのみ) サーバスクリプトを使用して **sshd** サービスを停止します。

```
/sbin/init.d/sshd2 stop
```

3 **swremove** を使用してパッケージをアンインストールします。

```
swremove RSIT
```

注意: HP-UX では、非標準の場所へのインストールには対応していません。

IBM AIX でのインストールおよびアンインストール

Reflection for Secure IT を IBM AIX にインストールするには

1 root としてログインします。

2 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

3 **uncompress** コマンドを使用してパッケージを圧縮解除します。

```
uncompress package_name.bff.Z
```

たとえば、以下のようになります。

```
uncompress rsit-server-7.2.0.999-powerpc-aix5.bff.Z
```

4 **installp** コマンドを使用してパッケージをインストールします。

```
installp -d. RSIT.ssh
```

アンインストールするには

1 root としてログインします。

2 (サーバのみ) サーバスクリプトを使用して **sshd** サービスを停止します。

```
/etc/rc.d/init.d/sshd stop
```

3 **installp** コマンドを使用してパッケージをアンインストールします。

```
installp -u RSIT.ssh
```

4 ステップ 3 で **installp** を実行したディレクトリで隠しファイル `.toc` を削除します。

注意: IBM AIX では、非標準の場所へのインストールには対応していません。

既存の構成ファイルからの設定の移行

Reflection for Secure IT とともにインストールされる移行スクリプトは、以下の製品のいずれかを使用して構成された設定の移行に使用することができます。

F-Secure UNIX クライアントおよびサーバ
Reflection for Secure IT 6.x UNIX クライアントおよびサーバ
Reflection for Secure IT 7.x UNIX クライアントおよびサーバ

移行スクリプトは以下の場所にインストールされます。

```
/etc/ssh2/migrate.sh
```

このスクリプトは構成ファイルを調べて、設定の変更が必要かどうかを判断します。 変更が必要な場合は、これらの変更を適用することを確認するよう求められます。 移行を確認すると、必要な更新が行われた新規の構成ファイルが、元のファイルのバックアップとともに作成されます。 すべての操作の詳細がスクリプトの出力およびログファイルに記録されます。 ログファイルには、どの設定が移行され、どの設定が移行できなかったかが示されます。 ログファイルは、変換されたファイルと同じディレクトリに作成され、変換後のファイル名をベースとする名前 (`sshd2_config_migration.log` など) が付けられます。

グローバル構成ファイルを移行するには

注意: 引数を指定しないで移行スクリプトを実行すると、`/etc/ssh2` ディレクトリ内にあるファイルが移行されます。 `/etc/ssh2/sshd2_config` と `/etc/ssh2/ssh2_config` に既定以外の設定が含まれている場合は、これらのファイルを移行するかどうか確認するよう求められます。 これらの設定に既定値が含まれている場合 (これは、以前のバージョンをアンインストールして、現行バージョンをインストールした後の予期される状態)、スクリプトは最新のバックアップファイル (`*.rpmsave`、`*.save`、`*.backup` など) を探して、それらのバックアップファイル内の設定を移行するかどうかを確認するよう求めます。

1 root としてログインします。

2 以前のバージョンをアンインストールします。

3 新規のバージョンをインストールします。

4 引数を指定しないで移行スクリプトを実行します。

```
/etc/ssh2/migrate.sh
```

5 プロンプトに対して応答します。

6 移行した設定および移行ログを確認し、必要な場合は、移行したバックアップファイルを `sshd2_config` および `ssh2_config` に結合します。

ユーザ構成ファイルを移行するには

1 root としてログインします。

2 移行スクリプトを実行して、移行するファイルを指定します。例えば、以下のようになります。

```
/etc/ssh2/migrate.sh client ~/.ssh2/ssh2_config
```

PKI 設定を移行するには

Reflection PKI Services Managerが、Reflection for Secure IT 6.x または F-Secure 構成ファイルが存在するコンピュータ上にインストールされている場合、以下の手順を使用して証明書設定を移行できます。

1 root としてログインします。

2 **pkid** と **-m** オプションを使用して、以前のバージョンの構成ファイルから設定を移行します。例えば、以下のようになります。

`/etc/ssh2/` にある `sshd2_config` および `ssh2_config` ファイル内の PKI 設定を、PKI Services Manager 構成フォルダ内の `pki_config` および `pki_map` ファイルに移行するには

```
/usr/local/sbin/pkid -m /etc/ssh2/
```

`sshd2_config.backup` 内の PKI 設定を移行して、指定された出力ディレクトリに新規の PKI Services Manager 構成ファイルを作成するには

```
/usr/local/sbin/pkid -b /output/path/ -m /etc/ssh2/sshd2_config.backup
```

3 PKI Services Manager データディレクトリ内の `logs` ディレクトリに作成される移行ログを確認します。(既定では、「情報」レベルとしてログに記録されます。 このレベルは、**-d** を使用して高めることができます。)

注意: 移行先のフォルダ内の `pki_config` ファイルにすでにトラストアンカが構成済みである場合、移行は発生しません。 これは、すでに構成済みの変更が移行によって上書きされないようにするためです。

Reflection PKI Services Manager のインストール

Reflection PKI Services Manager は、X.509 証明書検証サービスを提供するサービスです。 ユーザまたはサーバ証明書認証に対応するには、このアプリケーションをダウンロードしてインストールする必要があります。 これは、無料です。

Reflection PKI Services Manager をインストールするには

1 root としてログインします。

2 インストールパッケージをコンピュータにコピーして、このファイルが含まれるディレクトリに移動します。

3 gzip を使用してパッケージを解凍します。

`gzip -d` *`package_name.`*`tar.gz`

例:

```
gzip -d pkid_1.2.0.999-i386-solaris.gz
```

4 tar を使用してファイルを展開します。

```
tar -xf package_name.tar
```

この結果、パッケージ名に基づいてディレクトリが作成されます。例:

```
pkid_1.2.0.999--i386-solaris/
```

5 このディレクトリに変更します。例:

```
cd pkid_1.2.0.999-i386-solaris
```

6 インストールスクリプトを実行します。

```
./install.sh
```

7 インストール場所の指定を求められます。既定の場所から変更しない (推奨) 場合は、これらのプロンプトに対して [Enter] キーを押します。

注意:

- UNIX では、インストールスクリプトによって自動的にサービスが起動されます。
- Reflection PKI Services Manager が証明書を検証できるようにするには、既定の構成を編集してファイルをマッピングする必要があります。

アンインストールするには

1 root としてログインします。

2 アンインストールスクリプトを実行します。このスクリプトは、PKI Services Manager データフォルダ中の bin ディレクトリにインストールされます。
既定のパスは次のとおりです。

```
/opt/attachmate/pkid/bin/uninstall.sh
```

注意: 注意: アンインストールスクリプトによって、既存の構成ディレクトリ (既定では (`/opt/attachmate/pkid/config/`) の名前が、現在の日付と時刻に基づいた名前を使用して変更されます。例えば、config.20110101143755 のようになります。local-store ディレクトリと、このディレクトリに追加した証明書は、変更されずに残ります。

第 2 章

基本的な操作

この章の内容

サーバの起動および停止

sshd サービスはインストール後に自動的に起動します。

sshd サービスの起動、停止、再起動に使用できるスクリプトがインストールされています。スクリプトの名前とインストール場所は、使用しているオペレーティングシステムによって異なります。このスクリプトを使用してサーバを起動した時は、以下の **sshd** コマンドが呼び出されます。

```
sshd -oPidFile=sshd_PidFile_keyword_value
```

注意: **inetd** を使用して **sshd** を起動してはなりません。 この構成には対応していません。 この構成を FIPS モードで試行した場合は、各ユーザ接続に極めて長い時間がかかることになります。この理由は、**sshd** では、接続ごとに必須のセルフテストを実行する必要があるからです。

---

sshd サービスを直接起動するには

1 root としてログインします。

2 以下のように完全なパス情報を含めます。

```
/usr/sbin/sshd options
```

サーバスクリプトを Linux で実行するには

注意: 以下のコマンドはすべての Linux プラットフォーム上で機能します。ただし、実際のスクリプトファイルは、異なる場所へインストールされる場合があります。

---

1 root としてログインします。

2 以下のコマンドを使用して、**sshd** サービスを起動、停止、および再起動します。

```
/etc/init.d/sshd start
/etc/init.d/sshd stop
/etc/init.d/sshd restart
```

サーバスクリプトまたはサービスを Oracle Solaris で実行するには

1 root としてログインします。

2 以下のコマンドを使用して、**sshd** サービスを起動、停止、および再起動します。

- Solaris 8 および 9 では、以下のコマンドを使用して、**sshd** サービスを起動、停止、および再起動します。

  ```
  /etc/init.d/sshd2 start
  /etc/init.d/sshd2 stop
  /etc/init.d/sshd2 restart
  ```

- Solaris 10 では、以下のサービスオプションを使用して、サービスを起動、停止、および再起動し、サービスの状態をチェックします。

  ```
  svcadm enable network/ssh
  svcadm disable network/ssh
  svcadm restart network/ssh
  svcs -l network/ssh
  ```

- Solaris 11 では、以下のサービスオプションを使用して、サービスを起動、停止、および再起動し、サービスの状態をチェックします。

  ```
  svcadm enable network/RSITsshs
  svcadm disable network/RSITsshs
  svcadm restart network/RSITsshs
  svcs -l network/RSITsshs
  ```

サーバスクリプトを HP-UX で実行するには

1 root としてログインします。

2 以下のコマンドを使用して、sshd サービスを起動、停止、および再起動します。

```
/sbin/init.d/sshd2 start
/sbin/init.d/sshd2 stop
/sbin/init.d/sshd2 restart
```

サーバスクリプトを IBM AIX で実行するには

1 root としてログインします。

2 以下のコマンドを使用して、sshd サービスを起動、停止、および再起動します。

```
/etc/rc.d/init.d/sshd start
/etc/rc.d/init.d/sshd stop
/etc/rc.d/init.d/sshd restart
```

SSH 接続の確立

通常、既定状態のままホストに接続し、パスワードを使用してサーバにログインすることができます。リモートサーバにターミナル接続するには **ssh** を使用します。構文は次のとおりです。

ssh [*options*] [*user@*]*hostname*[*#port*] [*remote_command* [*arguments*] ...]

ユーザを指定しないと、クライアントは現在のログイン名を使用して接続します。ポートを指定しないと、クライアントは既定のポート (クライアント構成ファイル内で変更されていない限り 22) を使用します。

コマンドを指定しないと、**ssh** は、リモートホスト上に新規のセッションを作成します。コマンドを指定すると、同コマンドがホスト上で実行されてから、**ssh** が終了します。ユーザを指定しないと、現在のユーザ名が使用されます。

既定値を使用して、リモートサーバへの端末セッションを開くには、以下の手順に従います。

1 **ssh** を使用して、サーバに接続します。たとえば、以下のようになります。

```
ssh joe@myhost
```

2 初回ホスト接続時、ホスト認証の確認を求めるメッセージが表示されます。たとえば、以下のようになります。

```
Host key not found in hostkeys database.
Key fingerprint:
xesem-cyvic-puhef-penyl-dugid-kxpif-tizyh-behen-gymum-fozyb-cuxex
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)? [Enter=no]
```

当該ホストのシステム管理者に問い合わせて、ホスト鍵の有効性を確認できます (この情報を取得するために管理者が使用できる手順については、「ホスト公開鍵の指紋の表示 『40ページ 』」を参照してください)。

3 このメッセージに対する応答として yes と入力して、このホストへの接続を受け入れます。これにより、(~/.ssh2/hostkeys 内の) 既知のホスト鍵リストへホスト鍵が追加されます。鍵を保持しているホストは、信頼されているホストとなり、不明なホストに関するメッセージは以降の接続から表示されなくなります。

4 パスワードを入力して、接続を完了します。

注意: 初期接続を簡略化するとともに、不明の鍵をユーザが受け入れることができる危険性をなくすために、管理者は、ユーザ固有またはグローバルの既知のホストリストに手作業でホスト鍵を追加することができます。詳細については、「クライアントの既知のホストリストへの鍵の追加 『39ページ 』」を参照してください。

---

sftp を使用したファイル転送

sftp を使用すると、ローカルコンピュータとリモートホスト間で安全なファイル転送を行うことができます。また、ディレクトリの作成やファイルパーミッションの変更など、ほかのファイル管理コマンドを実行することもできます。**sftp** は、対話型で使用することも、操作を自動化するためにバッチファイルと組み合わせて使用することもできます。コマンドラインオプションの詳細については、「sftp コマンドラインオプション 『152ページ 』」を参照してください。**sftp** コマンドリファレンスについては、「対応している sftp コマンド 『155ページ 』」を参照してください。

対話型 sftp セッションを開くには、以下の手順に従います。

1 リモートホストへ接続します。たとえば、以下のようになります。

```
sftp joe@myhost.com
```

注意: Secure Shell サーバ上で使用している名前が現在のユーザ名と同じである場合は、ユーザ名を省略できます。

接続の確立が成功すると、以下のプロンプトが表示されます。

```
sftp>
```

2 以下のいずれかの操作を実行します。

操作	使用コマンド
対応しているコマンドのリストの表示	**help** 例えば、以下のように入力します。 `sftp> help`
対応しているコマンドに関する詳細の表示	**help** *command* 例えば、以下のように入力します。 `sftp> help put`
ファイルの転送および管理	使用可能なコマンド 『155ページ』 例えば、ファイル demo をローカルの作業ディレクトリからリモートの作業ディレクトリに転送するには、以下のように入力します。 `sftp> put demo`
セッションの終了	**quit** 例えば、以下のように入力します。 `sftp> quit`

注意: 初めてホストに接続すると、ホストの信頼性を確認するよう求めるメッセージが表示される場合があります。詳細については、「クライアント接続の確立 『22ページ』」を参照してください。

scp を使用したファイル転送

scp を使用すると、ローカルコンピュータとリモートホスト間または 2 つのリモートホスト間で安全なファイルコピーを行うことができます。 基本構文は以下のとおりです。

scp [[*user@*]*host*[*#port*]:]*source* [[*user@*]*host*[*#port*]:]*destination*

転送元ファイル名と転送先ファイル名の両方には、ホストとユーザの指定を含めることができます。これは、ファイルが当該ホストとの間でコピーされることを示すためです。

ローカルファイルを既定のリモートディレクトリにコピーするには

- 開始するには、以下のような例を使用します。

```
scp file_src joe@myhost.com:
```

リモートファイルをローカルの作業ディレクトリにコピーするには

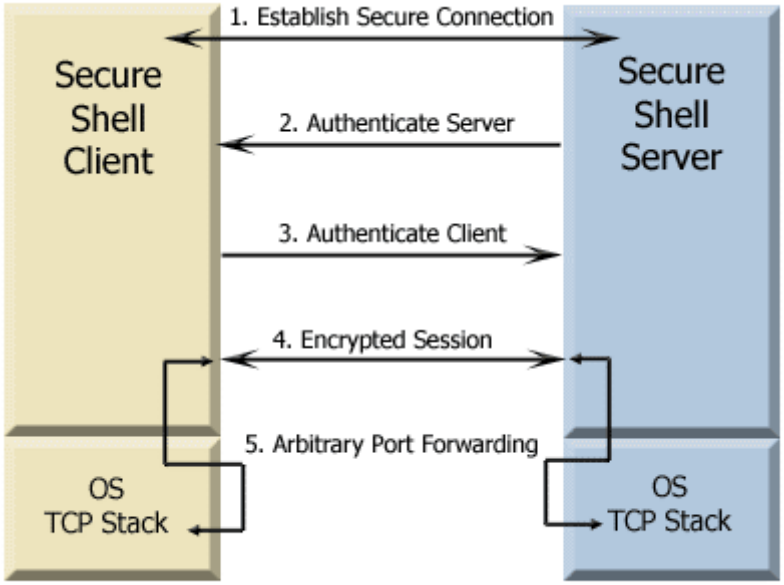
- 開始するには、以下のような例を使用します。

```
scp joe@myhost:/demo*.htm
```

詳細は、安全なファイルコピー 『68ページ』を参照してください。 コマンドラインオプションの詳細については、「**scp** コマンドラインオプション 『148ページ』」を参照してください。

Secure Shell の理解

以下の図は、Secure Shell トンネルの作成および Secure Shell トンネルを使用した安全なデータ送信に関わる基本的な手順の概略を示しています。



1. **安全な接続を確立します。**

   クライアントとサーバは、セッションの暗号化に使用する共有鍵と暗号、およびデータの完全性保証の確認に使用するハッシュを作成するために交渉します。詳細は、データ保護 『33ページ 』を参照してください。

2. **サーバを認証します。**

   サーバ認証によって、クライアントはサーバの ID を確認できます。 サーバからクライアントへの認証は、この認証プロセス中に 1 回だけ可能です。この認証に失敗した場合は、接続できません。詳細は、サーバ認証 『37ページ 』を参照してください。

3. **クライアントを認証します。**

   クライアント認証によって、サーバはクライアントユーザの ID を確認できます。既定で、クライアントは認証を複数回試行できます。サーバとクライアントは、1 つまたは複数の認証方式に合意するように交渉します。詳細は、クライアント認証 『49ページ 』を参照してください。

4. **暗号化されたセッションを介してデータを送信します。**

   暗号化されたセッションが確立されると、Secure Shell サーバとクライアント間で交換されるすべてのデータが暗号化されます。この段階で、ユーザはサーバへの安全なリモートアクセスが可能になり、保護されたチャネルを通じて安全にコマンドを実行し、ファイルを転送することができます。詳細は、安全なファイル転送 『65ページ 』を参照してください。

5. **ポート転送を使用して、その他のクライアントとサーバ間で安全な通信を行います。**

   トンネリングとしても知られるポート転送は、アクティブなセッションにおける Secure Shell チャネルを通じて通信をリダイレクトするための方法を提供します。ポート転送が構成されると、指定のポートへ送信されるすべてのデータは、保護されたチャネルを通じてリダイレクトされます。詳細は、ポート転送 『75ページ 』を参照してください。

第 3 章

構成ファイル

この章の内容

クライアント構成ファイル

Reflection for Secure IT 構成ファイルは、**ssh** を使用して作成された接続を制御します。また、クライアント構成ファイル内の設定によって、**scp** および **sftp** 接続も制御します。 既定のグローバル構成ファイルは以下のとおりです。

```
/etc/ssh2/ssh2_config
```

このファイルは、Reflection for Secure IT のインストール時にインストールされます。インストールされた同ファイルには、クライアント設定の既定値を示すコメントアウトされた行が含まれます。 このファイルの複製が `/etc/ssh2/ssh2_config.example` にインストールされます。

さらに、以下の場所を使用して、個々のユーザ用に構成ファイルを作成することができます。

```
~/.ssh2/ssh2_config
```

ssh クライアントは、以下の順番で設定を累加的に処理します。 1 つの設定が複数の場所に構成されている場合は、同じ設定について、最後に処理された値が、それ以前のすべての値を置き換えます。

1. システム全体にわたる構成ファイル:`/etc/ssh2/ssh2_config`
2. ユーザ固有の構成ファイル: `~/.ssh2/ssh2_config`
3. **ssh** コマンドラインで **-F** スイッチを使用して指定されるオプションのユーザ構成ファイル
4. **ssh**、**scp**、および **sftp** で使用されるコマンドラインオプション

クライアント構成ファイルのキーワードの詳細については、「クライアント構成のキーワード 『107ページ』」を参照してください。

構成ファイルの形式

構成ファイルは、キーワードと、その後に続く値から構成されます。また、オプションのホストスタンザを使用すると、個々のホストまたはホストのグループに固有の設定を構成することができます。1 つの設定がファイル内の複数の場所に構成されている場合は、一覧のより下の方で構成された値が、前出の値を置き換えます。

番号記号 (#) で開始する行はすべてコメントです。空の行は無視されます。

正規表現

正規表現は、POSIX 拡張構文を使用して評価されます。正規表現のルールの詳細については、以下を参照してください。

http://www.opengroup.org/onlinepubs/7990989775/xbd/re.html

キーワードの構文

キーワードはすべて値を必要とします。キーワードと値は、スペースで区切るか、またはオプションのスペースと1 つの「=」だけで区切ることができます。値にスペースが含まれている場合はその値を引用符 (一重または二重) で囲みます。たとえば、以下のようになります。

```
key value
```

```
key=value
```

```
key="value with spaces"
```

```
key=value1, value2
```

キーワードは、大文字と小文字が区別されません。

ホストスタンザ

クライアント構成ファイルはホストスタンザに対応しています。 ホストスタンザは、各種設定をそれぞれのホストに適用するために使用します。ホストスタンザを作成するには、個々のホストまたはホストのグループを識別する正規表現を使用します。当該正規表現を改行の先頭に配置し、その後にコロン (:) を続けます。*この行に空白を含めることはできません。接続が開始されると、クライアントは、ホストスタンザ表現と、当該接続用に指定されたホスト名との照合を行います。ホストスタンザ表現が、指定されたホストに一致している場合、当該スタンザ内の値が接続に適用されます。クライアントは、ファイルの最後尾に達するまで、一致するホストスタンザの検索を続行し、該当する設定をすべて適用します。同じキーワードについては最後の値が前出のすべての値を置き換えるため、グローバル設定をホスト固有の設定よりも前に配置する必要があります。スタンザの外部にある設定はすべての接続に適用されますが、スタンザ内に配置された後続の設定によって置き換えられる可能性があります。*

グローバル設定は、「.*:」のラベルの付いたスタンザを作成することによって構成できます。このスタンザ内の設定は、コマンドラインに指定したすべてのホストに適用されます。

注意: このスタンザ内で構成したグローバル設定は、ホストが指定されていない接続には適用されません。ホストを指定せずに接続の作成を正常に完了するには、Host キーワードがホストスタンザの外部に表示されている構成ファイルを使用する必要があります。

以下の例では、既定のユーザ名を「joe」に設定し、samplehost への接続で同ユーザ名を「guy」に変更します。

```
.*:
   user=joe
samplehost:
   user=guy
```

コマンドラインオプション

クライアントとサーバの接続は、構成ファイルを使用する以外に (使用する代わりに) コマンドラインオプションを使用して構成することができます。コマンドラインオプションは、構成ファイルの設定を置き換えます。構成ファイル内で構成可能なオプションはすべて、コマンドラインで **-o** オプションを使用して設定することもできます。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

以下のコマンドラインは、ID ファイルを指定する 2 つの等価な方法を示しています。

```
ssh -i testfile myname@myserver
ssh -o IdentificationFile=testfile myname@myserver
```

サーバ構成ファイル

Reflection for Secure IT サーバ構成ファイルには、**sshd** サーバ用の構成設定が含まれます。既定のグローバル構成ファイルは `/etc/ssh2/sshd2_config` です。代替ファイルを指定するには、**sshd** コマンドラインで **-f** オプションを使用します。また、特定のクライアントホストまたはユーザ用にオプションのサブ構成ファイルを作成および使用することもできます。

構成ファイルのサンプルが `/etc/ssh2/sshd2_config` にインストールされます。このファイルには、使用可能なすべての設定およびその既定値を示すコメント行が含まれます。同ファイルの複製は `/etc/ssh2/sshd2_config.example` にインストールされます。

サーバ構成ファイルの基本形式は、クライアント構成ファイルと同じです。詳細については、「構成ファイルの形式 『27ページ 』」を参照してください。

メインのサーバ構成ファイルに加えた変更は、直ちに新規の接続に反映されます。サーバを再起動する必要はありません。既存の接続は、元の設定を使用してアクティブな状態を維持しますが、後続の接続はこの新しい設定を使用します。

注意: **Port**、**ListenAddress**、および **FipsMode** を変更した場合は、再起動が必要です。

サーバは、以下の順番で設定を累加的に処理します。1 つの設定が複数の場所に構成されている場合は、同じ設定について、最後に処理された値が、それ以前のすべての値を置き換えます。

1. **sshd** コマンドラインで **-f** オプションを使用して指定されたグローバル構成ファイル、または代替ファイル
2. **HostSpecificConfig** キーワードを使用して作成および識別されているホスト固有のサブ構成ファイル
3. **UserSpecificConfig** キーワードを使用して作成および識別されているユーザ固有のサブ構成ファイル
4. **sshd** で使用されるコマンドラインオプション

サーバサブ構成ファイル

オプションのサブ構成ファイルを作成および使用して、ユーザまたはクライアントホストのサブセットに適用する設定を構成することができます。サブ構成ファイルは、新規接続ごとに分岐したプロセスによって読み込まれます。これらのファイルは実行時に読み込まれ、行った変更は、後続のすべての接続に反映されます。

ユーザ固有のサブ構成ファイル

ユーザ固有のサブ構成ファイルを構成するには **UserSpecificConfig** キーワードを使用します。このキーワードの構文は以下のとおりです。

`UserSpecificConfig` *`user_expression subconfig_file`*

ユーザ表現 『87ページ』が、接続を試行しているユーザに一致する場合、サーバは指定のサブ構成ファイルを使用します。ファイルのサンプルが以下のディレクトリにインストールされます。

`/etc/ssh2/subconfig/user.example`

`user.example` ファイルには、ユーザ固有のサブ構成ファイルで対応しているキーワードのリストが含まれます。

セキュリティ上の注意: **RequiredAuthentications** について、グローバルな許可リストまたは必須リストとは異なるユーザ固有のリストを構成した場合は、認証を試行する悪意あるユーザが、各種アカウントに関するクライアント/サーバ認証の交渉内容を比較し、許可されている認証のリスト内の相違点を利用して、アカウントが当該システムで有効であり、当該システム上のほかのアカウントとは異なることを判別するおそれがあります。

ホスト固有のサブ構成ファイル

クライアントホストのセブセットに適用される設定を構成するには **HostSpecificConfig** キーワードを使用します。このキーワードの構文は以下のとおりです。

`HostSpecificConfig` *`host_expression subconfig_file`*

ホスト式 『88ページ』がクライアントホストに一致する場合、サーバは指定されたサブ構成ファイルを使用します。ファイルのサンプルが以下のディレクトリにインストールされます。

`/etc/ssh2/subconfig/host.example`

`host.example` ファイルには、ホスト固有のサブ構成ファイルで対応しているキーワードのリストが含まれます。

サブ構成ファイルのサンプル

以下のサンプルファイルは、接続設定を特定のホストおよびユーザに適用するためにサブ構成ファイルをどのように使用できるかの例を示しています。 サーバ構成ファイルのサンプルでは、ホストサブ構成ファイルが **HostSpecificConfig** キーワードを使用して指定されています。 この例では、ホストサブ構成ファイル内の設定が、acme.com ドメインから接続されているすべてのユーザに適用されます。 当該ホストサブ構成ファイルでは、**UserSpecificConfig** キーワードを使用してユーザサブ構成ファイルが指定されています。 このユーザサブ構成ファイルの設定は、acme.com ドメインから接続されているユーザ名 joe からの接続に適用されます。

サーバ構成ファイル

`/etc/ssh2/sshd2_config` の内容のサンプル

```
Port=2222
RequireReverseMapping=yes
ResolveClientHostname=yes
#Specify a host-specific file for the users from acme.com
HostSpecificConfig=.*acme\.com /root/hostsubconfig
#Limit forwarding to user joe and constrain his forwarding rights
ForwardACL=allow remote joe .* peak.acme.com
```

ホストサブ構成ファイル

`/root/hostsubconfig` の内容のサンプル

```
AllowedAuthentications=publickey,password
Ciphers=aes128-cbc
#Allow sftp access only
SessionRestricted=subsystem
#Specify a user-specific file for user joe
UserSpecificConfig=joe /root/joesubconfig
```

ユーザサブ構成ファイル

`/root/joesubconfig` の内容のサンプル

```
RequiredAuthentications=publickey
#Allow both shell and sftp access
SessionRestricted=shell,subsystem
```

第 4 章

対応している暗号化アルゴリズム

この章の内容

暗号化

暗号化は、転送中のデータの機密性を保護します。 この保護はデータ送信前に、秘密鍵と暗号を使用してデータを暗号化することで実現します。 受信データは、同じ鍵と暗号を使用して解読する必要があります。 特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。

Reflection for Secure IT は、以下のデータ暗号化標準に対応しています。

- Arcfour、Arcfour128、および Arcfour256（ストリームモード）
- TripleDES（168 ビット）CBC モード
- Cast（128 ビット）CBC モード
- Blowfish（128 ビット）CBC モード
- AES（Rijndael）（128、192、または 256 ビット）CBC モードおよび CTR モード

データの完全性保証

データの完全性保証は、転送中のデータが変更されないように保証します。

Secure Shell 接続では、MAC (メッセージ認証コード) を使用してデータ保存性を確保します。クライアントとサーバは、別個に、転送されたデータの各パケットのハッシュを計算します。メッセージが送信中に変更されている場合、別のハッシュ値になり、パケットは拒否されます。任意のセッションに使用される MAC は、クライアントの優先順位 (サーバによっても対応される) の最上位にある MAC になります。

Reflection for Secure IT は、以下の MAC 標準に対応しています。

- hmac-sha1
- hmac-md5
- hmac-sha1-96
- hmac-md5-96
- hmac-ripemd-160
- hmac-sha256
- hmac-sha2-256
- hmac-sha512
- hmac-sha2-512

デジタル署名

デジタル署名は、公開鍵認証 (証明書認証など) に使用されます。認証する側は、デジタル署名を使用して、認証を受ける側が正しい秘密鍵を保持していることを確認します。Secure Shell クライアントは、デジタル署名を使用してホストを認証します。Secure Shell サーバは、公開鍵認証が構成されている場合に、デジタル署名を使用してクライアントを認証します。

Reflection for Secure IT は、次のデジタル署名アルゴリズムに対応しています。

- x509v3-rsa2048-sha256
- x509v3-sign-rsa
- x509v3-sign-dss
- ssh-rsa-sha2-256@attachmate.com
- ssh-rsa
- ssh-dss

暗号および MAC の構成

クライアントとサーバは、暗号および MAC の構成について同じキーワードに対応しています。クライアントのキーワードは `ssh2_config` で構成し、サーバのキーワードは `sshd2_config` で構成します。

キーワード	値
Ciphers	許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。 この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。 あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。 既定値は「AnyStdCipher」です。

MACs 許可される値は、「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」、および「hmac-ripemd160」です。
これらすべてに対応するには「AnyMac」を使用します。「AnyStdMac」を使用して「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」を指定します。
hmac-sha256 を指定すると hmac-sha2-256 も有効になります。
hmac-sha512 を指定すると hmac-sha2-512 も有効になります。
複数の MAC をカンマ区切り一覧として指定することもできます。
その他には、「none」、「any」 (AnyMac + 「none」と同等)、および「AnyStd」 (「AnyStdMac」 + 「none」と同等) オプションがあります。MAC について「none」で合意すると、メッセージ認証コードは使用されません。この場合データの整合性は保護されないため、「none」を含むオプションはお勧めしません。

Ciphers はまた、**ssh**、**scp**、および **sftp** コマンドラインで **-c** を使用して定義することもできます。たとえば、以下のようになります。

```
ssh -c blowfish-cbc joe@remote.com
```

MACs はまた、**ssh** および **sftp** コマンドラインで **-m** を使用して定義することもできます。たとえば、以下のようになります。

```
sftp -m hmac-md5 joe@remote.com
```

FIPS モード

アメリカ合衆国政府の連邦情報処理規格 (FIPS) 140-2 では、暗号化モジュールのセキュリティ要件が規定されています。暗号化製品は、米国政府公認の独立した研究所によって、特定の要件を満たしているか検証され 11 のカテゴリにわたってテストされます。この検証結果は National Institute of Standards and Technology (NIST - 国立標準技術研究所) に提出され、内容の確認後、証明書が発行されます。さらに、暗号化アルゴリズムは、その他の FIPS 仕様に基づいて検証および認定が行われる場合もあります。検証された製品の一覧と、ベンダーのセキュリティポリシ (認定されたモジュールの動作の定義) については、http://csrc.nist.gov/groups/STM/cmvp/validation.htm を参照してください。

FIPS モードで動作するように Reflection for Secure IT を構成するには、**FipsMode** キーワードを使用します。このキーワードにはクライアントとサーバの両方で対応しています。

注意: サーバの **FipsMode** 設定を変更した場合は、サーバを再起動して、加えた変更を有効にする必要があります。(SIGHUP 信号は新規セッションを FIPS モードで実行しますが、既存の接続には影響しません。)

FIPS モードは、以下のように適用されます。

- すべての通信で、FIPS 140-2 仕様に準拠したアルゴリズムが使用される必要があります。これらの仕様に準拠していないアルゴリズムは使用できません。ただし、従来のものとの互換性のためにアルゴリズムが NIST によって承認されている場合を除きます。
- ユーザとホストの両方に対する認証で許可される公開鍵の最小サイズが、既定の 512 ビットから最大 1024 ビットに再設定されます。
- Reflection for Secure IT は SecurID、GSSAPI、および RADIUS バイナリの FIPS 状態を確認できないため、これらの認証方式が FIPS 検証済みでない場合は、システム管理者が認証方式を手動で無効にする必要があります。FIPS 検証済みでない *すべての* PAM 認証方式が無効になるようにするには、サーバ構成ファイル () で PAM を無効にします (`UsePAM=no/etc/ssh2/sshd2_config`)。

第 5 章

サーバ認証

この章の内容

公開鍵認証の概要

既定では Reflection for Secure IT は公開鍵ホスト認証を使用します。 インストール時にサーバは新規のホスト鍵を自動的に生成します (または既存のホスト鍵を移行します)。既定の鍵は RSA 2048 ビット鍵です。

公開鍵暗号では、公開/秘密鍵ペアと数値アルゴリズムを併用して、データの暗号化および復号化を行います。鍵の片方は公開鍵で、これは通信相手に自由に配布できます。もう片方の鍵は秘密鍵で、鍵の所有者が安全に保管しておく必要があります。秘密鍵によって暗号化されたデータは公開鍵によってのみ復号化でき、公開鍵によって暗号化されたデータは秘密鍵によってのみ復号化できます。

鍵が認証に使用される時は、認証される側のユーザが、公開/秘密鍵ペアの秘密鍵を使用してデジタル署名を作成します。受信者は、対応する公開鍵を使用して、このデジタル署名の信頼性を確認する必要があります。これは、受信者が、他方のユーザの公開鍵のコピーを所有し、その鍵の信頼性を信頼しなければならないことを意味します。

公開鍵認証は以下のようにして行われます。

ホスト認証のために公開鍵認証が使用される時は、以下の一連のイベントが行われます。

1. Secure Shell クライアントが接続を開始します。
2. サーバが公開鍵をクライアントに送信します。

3. クライアントが、信頼されているホスト鍵ストアからこの鍵を検索します。

クライアントによる鍵の検索結果	発生するイベント
ホスト鍵を見つけ、クライアントコピーが、サーバによって送信された鍵に一致する	認証は次の段階に進みます。
ホスト鍵を検出 *できない*	クライアントは、ホストが不明であるというメッセージを表示し、ホスト鍵の指紋を提供します。ユーザが不明な鍵を受け入れることができるようにクライアントが構成されている場合 (既定)、ユーザは、この鍵を受け入れることができ、認証は次の段階に進みます。 厳格なホスト鍵の確認が履行される場合は、クライアントによって接続が終了されます。
ホスト鍵を見つけたが、クライアントコピーが、サーバによって送信された鍵に一致 *しない*	クライアントは、鍵が既存の鍵に一致しないという警告を表示し、サーバによって送信された鍵の指紋を表示します。ユーザが不明な鍵を受け入れることができるようにクライアントが構成されている場合 (既定値)、ユーザは、この新しい鍵を受け入れることができます。 厳格なホスト鍵の確認が履行される場合は、クライアントによって接続が終了されます。

4. サーバが、受信した公開鍵に対応する秘密鍵を実際に保持していることを確認するために、クライアントはサーバに試行 (任意のメッセージ) を送信して、当該メッセージテキストに基づきハッシュ 『192ページ』を計算します。
5. サーバは、試行メッセージに基づきデジタル署名を作成します。これを行うために、サーバは別個にメッセージのハッシュを計算し、次に、サーバの秘密鍵を使用して、この計算したハッシュを暗号化します。サーバは、当該デジタル署名を元の試行に付加して、この署名付きのメッセージをクライアントに返します。
6. クライアントは、公開鍵を使用して署名を復号化し、クライアント自身が計算したハッシュと当該ハッシュを比較します。値が一致すると、ホスト認証が成功します。

新規のホスト鍵の作成

ほとんどの場合、既定のサーバホスト鍵に変更を加える必要はありません。サーバのインストールパッケージは、既存のホスト鍵ペアがすでに存在しているかどうかを確認します。ホスト鍵が見つからない場合は、パッケージが新しいホスト鍵ペアを作成し、サーバはこの鍵ペアをホスト認証に使用します。ホスト鍵がすでに `/etc/ssh2` に存在している場合、Reflection for Secure IT はこの鍵を使用します。OpenSSH ホスト鍵が `/etc/ssh` に存在する場合、Reflection for Secure IT は鍵を適正な形式に移行してから、この移行後の鍵を使用します。

新規のホスト鍵を作成および使用するには

1 root としてログインします。

2 サーバスクリプトを使用して sshd のインスタンスをすべて終了します(追加情報については、「サーバの起動および停止 『21ページ』」を参照してください)。

3 **ssh-keygen** を使用して新規のホスト鍵を生成します。たとえば、以下のようになります。

```
ssh-keygen -P /etc/ssh2/hostkey2
```

注意: **-P** オプションを使用すると、パスフレーズ保護がない鍵が作成されます。この鍵はホスト鍵用に必要です。

4 (オプション) 新しいホスト鍵の名前/場所を使用する場合は、サーバ構成ファイル (/etc/ssh2/sshd2_config) を編集します。新しい名前および場所を指定するには **HostKeyFile** キーワードを使用します。

```
HostKeyFile=/etc/ssh2/hostkey2
```

既定のホスト鍵名 (/etc/ssh2/hostkey) を引き続き使用する場合、この手順は実行不要です。

5 サービスを再起動します。

クライアントの既知のホストリストへの鍵の追加

既定では、クライアントがサーバへの接続を初めて試行した時に、不明なホストであることを示すメッセージがユーザに表示されます。このメッセージには、ホスト鍵を識別する指紋が含まれています。当該ホスト鍵が実際に正しいホスト鍵であることを確認するには、正しい鍵の指紋を確認できるホストのシステム管理者に問い合わせる必要があります。この検証を行わないと、クライアントは「中間者」攻撃に合う危険があります。初期接続を簡略化するとともに、不明の鍵をユーザが受け入れることができる危険性をなくすために、ユーザは、クライアントの既知のホストリストに手作業でホスト鍵を追加することができます。

クライアントの既知のホストリストへサーバ鍵を追加するには

注意: サーバの公開ホスト鍵の、正しい名前のコピーが必要になります。既知のホスト鍵のクライアントコピーでは、以下のファイル名形式を使用します。

```
key_port_host,IP.pub
```

ここで *port* は、ssh 接続に使用されるポート、*host* はホスト名、*IP* はホストの IP アドレスです (初期のバージョンでは key_*port_host*.pub を使用し、この形式はまだ対応されています)。

正しい名前の鍵を取得する簡単な方法は、サーバへの初期接続を作成してから、クライアントによるホスト鍵の受け入れと名前の指定を許可することです。このようにすると、このホスト鍵のコピーを配布できるようになります。これは、以下の手順で使用されている手法でもあります。

1 サーバから、**ssh-keygen** を使用して、サーバの公開ホスト鍵の指紋を表示します。

```
ssh-keygen -F /etc/ssh2/hostkey.pub
```

2 当該ホストにまだ接続されていないクライアントから、サーバへの接続を開始します。

```
ssh myname@myserver
```

ホスト鍵が、ホスト鍵データベース内にないことを示すメッセージが表示されます。

3 このメッセージに含まれるホスト鍵の指紋が、実際のホスト鍵の指紋に一致することを確認し、「yes」と入力して当該ホスト鍵を受け入れます。

受け入れたホスト鍵の名前と場所を示すメッセージが表示されます。たとえば、以下のようになります。

```
Host key saved to /home/joe/.ssh2/hostkeys/key_22_myserver,10.10.1.123.pub
```

4 この鍵をクライアントコンピュータの既知のホストリストにコピーします。

- このホスト鍵をクライアントコンピュータのすべてのユーザ用に追加するには、ホスト公開鍵ファイルを `/etc/ssh2/hostkeys` にコピーします。

  または、

- このホスト鍵を個別のユーザ用に追加するには、ホスト公開鍵ファイルを `~/.ssh2/hostkeys` にコピーします。

5 (オプション) ユーザが不明なホスト鍵を受け入れることができないように **StrictHostKeyChecking** を有効にします。以下の行をシステム全体にわたる構成ファイル (`/etc/ssh2/ssh2_config`)、またはユーザ固有の構成ファイル (`~/.ssh2/ssh2_config`) に追加できます。

```
StrictHostKeyChecking=yes
```

ホスト公開鍵の指紋の表示

クライアントユーザがサーバへ初めて接続すると、サーバの公開ホスト鍵の指紋が含まれるメッセージが表示されます。サーバ管理者は、サーバ上にホスト鍵の指紋を表示することでこの鍵の有効性を確認できます。

サーバの公開ホスト鍵を識別する指紋を表示するには

1 サーバにログインします。

2 **ssh-keygen** を使用してホスト鍵の指紋を表示します。

```
ssh-keygen -F /etc/ssh2/hostkey.pub
```

サーバ証明書認証の概要

証明書認証は、公開鍵認証で生じるいくつかの問題を解決するための公開鍵認証形式の 1 つです。公開鍵ホスト認証では、システム管理者は、サーバごとにホストの公開鍵を各クライアントの既知のホスト一覧に追加するか、クライアントユーザが未知のホストに接続する時にホストの識別情報を正しく確認するように要求する必要があります。証明書認証では、認証局 (CA) と呼ばれる、信頼されたサードパーティを使用してホストからの受信情報の有効性を確認することで、この問題を回避できます。証明書を使用すると、複数のサーバの一意の公開鍵ではなく単一のトラストアンカを使用して認証を構成することができます。

Reflection PKI Services Manager は PKI 設定の一元管理に対応しています。PKI Services Manager のインスタンスを 1 つインストールして構成し、対応する Attachmate 製品すべてに対して証明書検証サービスを提供することができます。

要件

要件	機能
Reflection PKI Services Manager がインストールされ、適正に構成されていること	PKI Services Manager は、証明書を検証してから、マップファイルを使用し、有効な証明書により認証可能なサーバを判別します。証明書の検証に成功するには、少なくとも 1 つのトラストアンカと 1 つのマッピングルールを構成する必要があります。さらに、中間証明書と証明書破棄情報へのアクセスも構成する必要がある場合があります。
CA によって署名された証明書とそれに関連付けられた秘密鍵がサーバにインストールされていること	サーバは、サーバの認証のため、この証明書をクライアントに送信します。
Reflection for Secure IT UNIX クライアントが、PKI Services Manager 公開鍵のコピーを保持しており、PKI Services Manager に接続するよう構成されていること	クライアントは、PKI Services Manager と通信することで、サーバ証明書の有効性を確認します。

RADIUS 認証は以下のようにして行われます。

1. Reflection for Secure IT サーバは、サーバ認証のためクライアントへ証明書を提示します。
2. Reflection for Secure IT クライアントは Reflection PKI Services Manager に接続します。(Reflection for Secure IT クライアントの **PkidAddress** キーワードを使用してこの接続用のサーバ名とポートを設定します。)
3. Reflection for Secure IT は、インストールされている公開鍵を使用して PKI Services Manager の ID を確認します。(Reflection for Secure IT クライアントの **PkidPublicKey** キーワードを使用して鍵の名前と場所を設定します。)
4. Reflection for Secure IT は証明書とサーバ名を PKI Services Manager に送信します。
5. PKI Services Manager は、証明書が有効かどうかを判断し、PKI Services Manager 管理者によって PKI Services Manager マップファイル (既定では`/opt/attachmate/pkid/config/pki_mapfile`) に構成されているルールに基づき、この証明書を使ってサーバの認証が許可されるかどうかを決定します。この情報が Reflection for Secure IT に返されます。
6. 証明書が有効であり、その証明書を提示したサーバがその証明書で許可される ID である場合、サーバ認証は成功します。

認証証明書の取得

証明書を使用する認証を構成する前に、秘密鍵と、信頼される CA によって署名された関連する証明書が必要とされます。 サーバ認証には、これらをサーバにインストールし構成する必要があります。 ユーザ認証には、これらをクライアントにインストールし構成する必要があります。

秘密鍵および関連する証明書を取得する方法は、複数あります。 既存の秘密鍵用の証明書を取得する方法、新規の秘密鍵を生成して証明書を取得する方法、秘密鍵と証明書の両方を CA から取得する方法があります。

既存の秘密鍵用の証明書を取得するには

1 **ssh-certtool** を使用して、使用している秘密鍵の証明書要求を作成します。例えば、以下のようになります。

```
ssh-certtool -p privatekey pkcs10 "CN=acme,OU=demo,C=US"
```

これにより、要求したファイルは PKCS#10 形式で作成されます。 既定のファイル名は「`output.pkcs10`」です。

2 CA への証明書要求の送信

CA はデジタル署名付きの証明書を返します。

3 返された証明書が PKCS#12 (`*.pfx` または `*.p12`) または PKCS#7 ファイルとしてパッケージ化されている場合は、**ssh-keygen** を使用して、返されたそのパッケージから証明書を抽出できます。

PKCS#12 ファイルの内容を抽出するには **-k** を使用します。

```
ssh-keygen -k package.pfx
```

PKCS#7 ファイルの内容を抽出するには **-7** を使用します。

```
ssh-keygen -7 pkcs7file
```

新規の秘密鍵を生成して、証明書を取得するには

1 **ssh-certtool** を使用して、秘密鍵およびこの秘密鍵の証明書要求を作成します。例えば、RSA 鍵を生成するには以下のようにします。

```
ssh-certtool -n rsa pkcs10 "CN=acme,OU=demo,C=US"
```

これにより、要求したファイルは PKCS#10 形式で作成されます。 既定のファイル名は「`output.pkcs10`」です。

2 CA への証明書要求の送信

CA はデジタル署名付きの証明書を返します。

3 返された証明書が PKCS#12 (`*.pfx` または `*.p12`) または PKCS#7 ファイルとしてパッケージ化されている場合は、**ssh-keygen** を使用して、返されたそのパッケージから証明書を抽出できます。

PKCS#12 ファイルの内容を抽出するには **-k** を使用します。

```
ssh-keygen -k package.pfx
```

PKCS#7 ファイルの内容を抽出するには **-7** を使用します。

```
ssh-keygen -7 pkcs7file
```

秘密鍵と証明書の両方を CA から取得するには

1 要求を CA へ送信します。

CA は、秘密鍵とデジタル署名付きの証明書の両方を含む PKCS#12 (`*.pfx` または `*.p12`) を返します。

2 **ssh-keygen** と **-k** オプションを使用して、返されたパッケージから鍵と証明書を抽出します。例えば、以下のようになります。

```
ssh-keygen -k package.pfx
```

サーバ証明書認証の構成

開始する前に、サーバ証明書認証の概要 『40ページ 』に記載された要件を確認してください。

証明書を使用したサーバ認証を構成するには、Reflection PKI Services Manager をインストールおよび構成して、サーバおよびクライアントを構成する必要があります。構成を開始するには、以下の手順を使用しますが、そのほかに、多くの異なる手順を実行できます。詳細は、「Reflection PKI Services Manager ユーザガイド 」(http://support.attachmate.com/manuals/pki.html) を参照してください。

PKI Services Manager の単一のインスタンスをインストールおよび構成することで、複数の Reflection for Secure IT クライアントおよびサーバからの証明書認証要求に対応できます。ただし、Reflection for Secure IT 設定では PKI Services Manager のアドレスおよびポートとしてただ 1 つのエントリのみが許可されるため、この構成において単一障害ポイントが発生する可能性があります。PKI Services Manager にアクセスできないか、サーバが稼働中でない場合、証明書を使用したすべての認証試行は失敗します。負荷分散およびフェイルオーバ機能を実現するには、PKI Services Manager ホスト名としてラウンドロビン DNS エントリを定義するか、負荷分散機能を持つサーバに PKI Services Manager ホストを配置することができます。PKI Services Manager が Microsoft クラスタ環境で実行されるよう構成することもできます。

注意: 以下に示すパスは、既定のインストールオプションに基づいています。

PKI Services Manager を インストールおよび構成するには

1 Reflection PKI Services Manager サーバに root としてログインします。

2 Reflection PKI Services Manager 『18ページ 』 をインストールします。

3 トラストアンカとして指定する 1 つまたは複数の証明書のコピーを証明書ストアに入れます。既定の PKI Services Manager ストアは以下の場所にあります。

```
/opt/attachmate/pkid/local-store
```

4 テキストエディタで PKI Services Manager 構成ファイルを開きます。既定の名前と場所は次の通りです。

```
/opt/attachmate/pkid/config/pki_config
```

5 **TrustAnchor** キーワードを使用してトラストアンカーを識別します。例:

```
TrustAnchor = trustedca.crt
```

または

```
TrustAnchor = CN=SecureCA,O=Acme,C=US
```

注意: 複数のトラストアンカーを構成するには、**TrustAnchor** の行を追加します。

6 証明書取り消しチェックを構成します。例:

対象	サンプル構成
LDAP サーバに格納されている CRL を使用します。	`RevocationCheckOrder = crlserver` `CRLServers=ldap://crlserver`
OCSP レスポンダを使用します。	`RevocationCheckOrder = ocsp` `OCSPResponders = http://ocspresponder`

注意: 既定では、PKI Services Manager は、ローカルストアで CRL を探します。この構成を使用する場合、ローカルストアに CRL をコピーする必要があります。

7 証明書の信頼のチェーンで中間証明書が必要な場合は、これらの証明書へのアクセスを構成します。例:

対象	サンプル構成
ローカルストアに追加した中間証明書を使用します。	`CertSearchOrder=local`
LDAP サーバに格納されている証明書を使用します。	`CertSearchOrder=certserver` `CertServers=ldap://ldapserver`

8 変更内容を構成ファイルに保存します。

9 テキストエディタで PKI Services Manager マップファイル 『177ページ 』を開きます。既定の名前と場所は次の通りです。

```
/opt/attachmate/pkid/config/pki_mapfile
```

10 ホストの **RuleType** スタンザを作成して、有効な証明書で認証可能なホストを定義する 1 つまたは複数のルールを追加します。例えば、次のように入力します。

```
RuleType = host
  {myhost.com} Subject Contains "myhost"
```

詳細なルールのサンプルについては、「PKI Services Manager のマッピングルールのサンプル 『183ページ 』」を参照してください。

注意: 証明書が有効であると判明した後は、ルールが順番に (ルールの種類を基準に使用した後で、順序に従う) が処理されます。 証明書が、条件式に定義された要件を満たす場合 (または、ルールに条件が含まれていない場合)、そのルールに指定された、許可される ID の認証が許可されます。 最初の一致が検出された後は、追加のルールは適用されません。つまり、条件を指定しないでルールを作成する場合は、そのルールに、許可されるすべての ID が含まれている必要があります。

11 有効な PKI Services Manager 構成をテストします。

```
/usr/local/sbin/pkid -k
```

エラーはありません。構成は有効です。

12 Reflection PKI Services Manager を再起動します。

`/usr/local/sbin/pkid` の再起動

Reflection for Secure IT サーバを構成するには

1 サーバ証明書および関連する秘密鍵 『41ページ 』をインストールします。例えば、次のように入力します。

```
/etc/ssh2/server.key
```

```
/etc/ssh2/server.crt
```

2 ユーザのみの読み取り専用アクセスが可能なようにサーバ鍵の権限を設定します。

```
chmod 400 server.key
```

3 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

4 以下のキーワードを構成します。

```
HostCertificateFile=/etc/ssh2/server.crt
```

```
HostKeyFile=/etc/ssh2/server.key
```

5 サーバを再起動します 『21ページ 』。

Reflection for Secure IT クライアントを構成するには

1 PKI Services Manager が、Reflection for Secure IT クライアントと同じホスト上にインストールされていない場合は、PKI Services Manager 公開鍵を Reflection for Secure IT クライアントにコピーします。PKI Services Manager の鍵の場所は以下のとおりです。

```
/opt/attachmate/pkid/config/pki_key.pub
```

この鍵を Reflection for Secure IT クライアント上の任意の場所へコピーします。例えば、次のように入力します。

```
/etc/ssh2/pki_key.pub
```

2 テキストエディタでクライアント構成ファイル (`/etc/ssh2/ssh2_config`) を開きます。

3 **PkidPublicKey** を編集して、PKI Services Manager 公開鍵を保存した場所を指定します。例えば、次のように入力します。

```
PkidPublicKey=/etc/ssh2/pki_key.pub
```

4 **PkidAddress** を編集して、PKI Services Manager ホストおよびポートを指定します。例えば、次のように入力します。

```
PkidAddress=pkiserver.acme.com:18081
```

注意: ホストを指定してポートを省略した場合、既定の PKI Services Manager ポート (18081) が使用されます。

---

5 **HostKeyAlgorithms** が、ホスト鍵よりも X.509 証明書を優先するよう構成されていることを確認します。これが既定値です。

```
x509v3-rsa2048-sha256,x509v3-sign-rsa,x509v3-sign-dss,ssh-rsa-sha2-256@
attachmate.com,ssh-rsa,ssh-dss
```

Kerberos (GSSAPI) 認証

Kerberos は、クライアント認証とサーバ認証の両方の代替メカニズムを提供するセキュリティプロトコルです。Kerberos 認証は、KDC (Key Distribution Center) と呼ばれる信頼されたサードパーティに依存しています。Secure Shell プロトコルは、GSSAPI (Generic Security Services Application Programming Interface) を介して Kerberos 認証に対応しています。

Kerberos 認証を使用する利点には以下のようなものがあります。

- 信頼されたサードパーティを使用すると、公開鍵認証の使用時に発生する鍵管理タスクを省くことができます。
- Kerberos がサーバ認証に使用される時は、ホスト鍵は必要とされません。つまり、クライアントユーザは、不明のホストを示すメッセージに応答する必要はありません。

GSSAPI を使用したサーバ認証

既定で、Secure Shell 接続は、以下の一連のイベントを使用して確立されます。

1. 鍵交換。クライアントとサーバは、セッションについて共有される秘密鍵、暗号、およびハッシュの交渉を行います。
2. サーバ認証。既定では、この目的でサーバはホスト鍵を提示します。
3. クライアント認証。

GSSAPI がサーバ認証に使用される時は、最初の鍵交換時に Kerberos KDC によってサーバが認証されます。これ以降、サーバ認証は不要になり、サーバがクライアントにホスト鍵を送信することはありません。

GSSAPI を使用したクライアント認証

ユーザが KDC に対して認証されると、その後、当該ユーザは、ほかの Kerberos 対応のアプリケーションで使用可能な Kerberos 資格情報を保持します。 GSSAPI に対応するよう Reflection for Secure IT を構成している場合、サーバはクライアントユーザの認証に Kerberos 資格情報を使用します。 これは、KDC に対して認証されているユーザは、サーバへ接続するために追加の認証を受ける必要がないことを意味します。

Kerberos システム要件

Reflection for Secure IT は、以下の Kerberos 実装に対応しています。

- MIT Kerberos V5、リリース 1.5.4 以降
- Sun Solaris ネイティブ Kerberos ライブラリ

Reflection for Secure IT は、以下の Kerberos ライブラリを使用します。以下に示すキーワードを構成して、システム上のこれらのライブラリへの完全修飾パスを指定する必要がある場合があります (これらの設定の既定値は、使用しているオペレーティングシステムによって異なります)。

ライブラリ	キーワード	使用者
libgssapi_krb5.so	**LibGssKrb5**	クライアント (ssh2_config) およびサーバ (sshd2_config)
libkrb5.so	**LibKrb5**	サーバのみ (sshd2_config)

Kerberos サーバおよびクライアント認証の構成

Kerberos は、相互認証 (クライアントとサーバの両方) またはクライアントのみの認証に使用できます。

- 認証方法が gssapi-keyex の場合は、接続交渉の鍵交換部でサーバとクライアント両方の認証が行われます。この認証に失敗すると、接続が失敗します。これ以降、いずれの認証方法も試行されません。
- 認証方法が gssapi-with-mic の場合、Kerberos はサーバ認証に使用されません。サーバ認証に成功した後に、Kerberos を使用したクライアント認証が試行されます。Kerberos 認証が失敗すると、許可されている他の認証方法が試行されます。

以下に、重要な手順の概略を示します。詳細については、後続の各手順の説明を参照してください。

1. KDC への接続を構成します。
   - ホストプリンシパルを追加し、Secure Shell サーバホストに keytab ファイルをインストールします。
   - クライアントユーザプリンシパルを追加します。
2. (必要に応じて) サーバ構成ファイル内で **AllowedAuthentications** を構成します。
3. (必要に応じて) クライアント構成ファイル内で **AllowedAuthentications** および **GSSAPIDelegateCredentials** を構成します。
4. **kinit** を使用して KDC に対してクライアントユーザを認証してから、Secure Shell 接続を作成します。

KDC への接続を構成するには

1 Secure Shell サーバへログインします。

2 サーバが Kerberos レルムへの認証を行うように構成されていることを確認します。そのように構成されていない場合は、正しく構成された `krb5.conf` ファイルをインストールします。

3 管理特権を持つプリンシパルを使用して Kerberos レルムへの認証を行います。

```
kinit root/admin
```

4 Kerberos 管理ユーティリティを起動します。

```
/usr/krb5/sbin/kadmin
```

5 このサーバ用のホストプリンシパルを追加します。たとえば、ホスト `myhost.sample.com` を追加するには、以下のようになります。

```
addprinc -randkey host/myhost.sample.com
```

6 このサーバ用の keytab ファイルを取り出します。

```
ktadd host/myhost.sample.com
```

7 各クライアントユーザ用のプリンシパルを追加します。たとえば、Joe を追加するには、以下のようになります。

```
addprinc joe
```

サーバに Secure Shell 設定を構成するには

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

2 **AllowedAuthentications** キーワードを編集します。

目的	使用する設定
Kerberos を使用してサーバとクライアントの両方を認証する	`AllowedAuthentications=gssapi-keyex`
Kerberos を使用してクライアントのみを認証する	`AllowedAuthentications=gssapi-with-mic`

クライアントを構成するには

1 テキストエディタでクライアント構成ファイル (`/etc/ssh2/ssh2_config`) を開きます。

2 **AllowedAuthentications** キーワードを編集します。

目的	使用する設定
Kerberos を使用してサーバとクライアントの両方を認証する	`AllowedAuthentications=gssapi-keyex`
Kerberos を使用してクライアントのみを認証する	`AllowedAuthentications=gssapi-with-mic`

3 (オプション) チケットの転送を有効にする場合は、**GSSAPIDelegateCredentials** キーワードを編集します。

```
GSSAPIDelegateCredentials=Yes
```

Kerberos 資格情報を取得するには

Secure Shell サーバに接続するためには、Kerberos 資格情報を取得する必要があります。

1 **kinit** を使用して認証を行います。

```
kinit -f
```

注意: **-f** オプションは必要ありません。このオプションは、転送可能なチケットを要求します。チケットの転送が有効になっている場合 (**GSSAPIDelegateCredentials** を使用) は、このチケットがサーバに転送されます。これは、追加の Kerberos 資格情報を取得する必要なく、ほかの Kerberos 対応のアプリケーションにアクセスできることを意味します。

2 Kerberos KDC 用のパスワードを入力します。

第 6 章

ユーザ認証

この章の内容

Reflection for Secure ITでは、いくつかのクライアント認証方法を用意しています。クライアントとサーバの両方において、構成ファイルを使用してクライアント認証を指定します。最終的には、クライアントはサーバ側の指定した内容に従います。Secure Shell 接続の交渉では、サーバは、許可されている方法と必須の方法のリストを提示します。クライアントとサーバは、このリストを基に交渉を行い、その一環でクライアント認証方式が決定されます。

サーバが許可したユーザ認証方式が複数ある時には、クライアントによって設定された優先順位に従います。接続では、クライアントの優先順位 (サーバによっても許可されている) の最上位にある最初の認証法が使用されます。サーバが、複数の方法を要求するように構成されている場合は、接続を確立するのに複数の認証方法が必要になります。

パスワードおよびキーボード対話型認証

Reflection for Secure IT サーバは、既定でパスワードとキーボード対話形式両方の認証に対応しています。

認証方法	説明
パスワード	クライアントユーザに、Secure Shell サーバホスト上の当該ユーザ用のログインパスワードを入力するよう求めます。 パスワードは、暗号化されたチャネルを介してホストに送信されます。
キーボード対話形式	単純なパスワード認証を含む、キーボードを使用して認証データを入力する手順に対応します。これによって、Secure Shell クライアントは、RSA SecurID トークンまたは RADIUS サーバなどのさまざまな認証機構に対応できるようになります。 例えば、クライアント管理者はキーボード対話型認証を構成して、パスワード更新などの複数のプロンプトが必要な状況を処理できます。 キーボードデータは、暗号化されたチャネルを介してホストに送信されます。

パスワード認証の構成

パスワード認証には既定で対応しており、サーバまたはクライアントのいずれでも、この認証方法を使用するための構成は必要ありません。 既定のサーバまたはクライアント構成を変更する場合は、以下の手順を使用します。

注意: パスワード認証は、優先される方式であるキーボード対話型方式を使用しても実行することができます。

クライアント側でパスワード認証を構成するには

1 テキストエディタでクライアント構成ファイル (`/etc/ssh2/ssh2_config`) を開きます。

2 **AllowedAuthentications** キーワードを編集します。

サーバ側でパスワード認証を構成するには

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

2 **AllowedAuthentications** または **RequiredAuthentications** を編集します。

3 (オプション) **PasswordGuesses** を使用して、ユーザが許可されるパスワード認証の最大試行回数を変更します (既定値は 3 です)。たとえば、以下のようになります。

```
PasswordGuesses=5
```

キーボード対話型認証の構成

キーボード対話型認証には既定で対応しており、サーバまたはクライアントのいずれでも、この認証方法を使用するための構成は必要ありません。

既定のサーバまたはクライアント構成を変更する場合は、以下の手順に従います。

クライアント側でキーボード対話型認証を構成するには

1 テキストエディタでクライアント構成ファイル (`/etc/ssh2/ssh2_config`) を開きます。

2 **AllowedAuthentications** キーワードを編集します。 例えば、キーボード対話型認証を要求するには、以下を使用します。

```
AllowedAuthentications=keyboard-interactive
```

サーバ側でキーボード対話型認証を構成するには

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

2 **AllowedAuthentications** または **RequiredAuthentications** を編集します。 たとえば、以下のようになります。

目的	必要な操作
キーボード対話型認証には対応するが、従来のパスワード認証には対応しない	パスワード (password) を許可一覧から削除します。例えば、以下のようになります。 `AllowedAuthentications=gssapi-keyex,gssapi-with-mic,publickey,keyboard-interactive`
キーボード対話型認証を要求する	以下のコマンドを入力します。 `RequiredAuthentications=keyboard-interactive`

1 (オプション) **AuthKbdInt.Retries** を使用して、キーボード対話型認証でユーザが許可される最大試行回数を変更します (既定値は 3 です)。 たとえば、以下のようになります。

```
AuthKbdInt.Retries=5
```

2 (オプション) **AccountManagement** を使用してアカウント管理を構成します。 詳細については、「プラグ可能な認証モジュール (PAM) 『58ページ 』」を参照してください。

公開鍵認証

公開鍵認証では、公開/秘密鍵ペアを信頼します。 公開鍵認証を構成するには、各クライアントユーザが、鍵ペアを作成して、公開鍵をサーバにアップロードする必要があります。 鍵がパスフレーズで保護されている場合は、公開鍵認証を使用して接続を完了する目的で、クライアントユーザは当該パスフレーズを入力するように求められます。

公開鍵ユーザ認証の構成

公開鍵認証では、クライアントとサーバ両方の構成を必要とします。以下に、重要な手順の概略を示します。詳細については、後続の各手順の説明を参照してください。

1. クライアントで鍵ペアを作成します。
2. クライアントID ファイル (`~/.ssh2/identification`) に、秘密鍵を識別する行を追加します。
3. 公開鍵をサーバ上のユーザのディレクトリ (`~/.ssh2`) にコピーします。
4. サーバ上のユーザの「authorization」ファイル (`~/.ssh2/authorization`) に、公開鍵を識別する行を追加します。

注意: 認証をセキュアに実装し、不正変更、情報の漏洩、スプーフィングを防止するために、クライアントとサーバで使用されるファイルおよびディレクトリにパーミッションと所有者を正しく構成します。この基準に満たない場合は、Secure Shell 接続と公開鍵認証が失敗します。詳細は、「ファイルおよびディレクトリの権限 『135ページ 』」を参照してください。

クライアント側で公開鍵認証を構成するには

1 (オプション) クライアントの **AllowedAuthentications** 設定を変更します。

公開鍵認証は既定で許可されているため、この手順は、当該既定値を変更する場合のみ実行する必要があります。対応している認証を変更するには、クライアント構成ファイル (`/etc/ssh2/ssh2_config`) を開きます。たとえば、公開鍵認証を要求するには、以下を使用します。

```
AllowedAuthentications=publickey
```

2 **ssh-keygen** ユーティリティを使用して公開/秘密鍵ペアを生成します。

例えば、以下のコマンドを実行すると、現在の作業ディレクトリに既定 (2048 ビット RSA) の鍵ペア (`mykey` および `mykey.pub`) が作成されます。この鍵の作成過程ではパスフレーズを入力するように求められます。パスフレーズを提供した場合は、この鍵を使用して認証を行う時に必ず、当該パスフレーズを使用する必要があります。

```
ssh-keygen mykey
```

次の例では、**-P** を使用して、パスフレーズで保護されない鍵を作成します。このオプションは、安全性には欠けますが、スクリプトやバッチファイルとともに使用する場合は推奨されます。**-t** は鍵の種類 (以下の例では DSA) を指定します。鍵の名前を指定していないため、鍵は既定の名前と場所 (この例では `$HOME/.ssh2/id_dsa_1024_myhost_a`。myhost は **hostname** コマンドで返されるシステムのホスト名) を使用して作成されます。

```
ssh-keygen -P -t dsa
```

3 クライアントの ID ファイルを作成 (または編集) します。同ファイルの既定の名前と場所は `~/.ssh2/identification` です。ユーザのみの書き込みアクセスが可能なようにこのファイルを構成します (600 を推奨)。

4 ID ファイル内に、作成した秘密鍵のための行を追加します。鍵の項目の形式は、`IdKey` の後に秘密鍵の名前を続けた形になります。たとえば、以下のようになります。

```
IdKey /home/joe/mykey
```

```
IdKey id_dsa_1024_myhost_a
```

注意: パス情報が提供されていない場合、クライアントは `~/.ssh2/` に一覧表示されている鍵を検索します。

サーバ側で公開鍵認証を構成するには

1 (オプション) サーバの **AllowedAuthentications** または **RequiredAuthentications** 設定を変更します。

公開鍵認証は既定で許可されているため、この手順は、既定の設定を変更する場合のみ実行する必要があります。対応している認証を変更するには、サーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。たとえば、公開鍵認証を要求するには、以下を使用します。

```
RequiredAuthentications=publickey
```

2 クライアントの公開鍵をサーバ上のユーザ固有の構成ディレクトリにコピーします。既定の場所は `~/.ssh2` です。

3 このサーバのこのユーザ用の鍵認証ファイルを作成 (または編集) します。このファイルには、サーバがユーザ認証で受け入れる鍵のリストが含まれます。既定の名前および場所は `~/.ssh2/authorization` です。ユーザのみの書き込みアクセスが可能なようにこのファイルを構成します (600 を推奨)。

4 認証ファイル内に、コピーした公開鍵のための行を追加します。鍵エントリの形式は、`Key` の後に公開鍵の名前が続きます。たとえば、以下のようになります。

```
Key /pathto/mykey.pub
```

```
Key id_dsa_1024_myhost_a.pub
```

リストに示されている鍵はすべて、サーバがユーザ認証に使用できます。鍵は、ユーザが絶対パスを指定しないかぎり、ユーザ固有の構成ディレクトリ (既定では `~/.ssh2/`) 内にあるとみなされます。 クライアントによって提示された鍵が「authorization」ファイル内にあるいずれの鍵にも一致しない場合、公開鍵認証は失敗します。

鍵エージェントの使用

鍵エージェント **ssh-agent** を使用して、認証に使用する秘密鍵を管理できます。 鍵エージェントを使用すると、秘密鍵を保存して、これらの鍵を **ssh**、**scp**、および **sftp** セッションの認証用に使用することが可能になります。 パスフレーズが必要とされるのはエージェントへ鍵を追加する時だけになるため、エージェントの使用によって、**ssh** に依存するスクリプトを簡素化することができます。 既定で、エージェントへの接続は転送可能になっています。つまり、保存された ID はネットワーク内の任意の場所で安全に使用することができます。

注意: エージェント転送には副次的なセキュリティ上のリスクが伴うため、この機能を許可しないこともできます。 クライアントでは **ForwardAgent**、サーバでは **AllowAgentForwarding** を使用します。

現行のシェルでエージェントを起動するには

- 以下のコマンドを使用します。

  ```
  eval `ssh-agent`
  ```

  eval を使用して起動する場合、当該プロセスを手作業で終了する必要があります。PID、または以下に示すように、**-k** を使用できます。

  ```
  ssh-agent
  ```

サブシェルでエージェントを起動するには

- コマンドの引数を使用して、シェルを指定します。例えば、以下のようになります。

  ```
  ssh-agent $SHELL
  ```

  サブシェルでエージェントを起動すると、シェルからログアウトした時にエージェントは自動的に終了します。

エージェントに鍵を追加するには

- 例えば、**ssh-add** を使用して、現行のシェルでエージェントを起動し、ID ファイル内の鍵をエージェントに読み込みます。以下のコマンドシーケンスを使用します。

  ```
  eval `ssh-agent`
  ssh-add
  ```

  鍵がエージェントに追加される時は、パスフレーズを入力するよう求められます。鍵を読み込んだ後は、読み込んだ鍵のいずれかを必要とするサーバに接続できますが、この場合、パスフレーズを入力する必要はありません。

注意:

- コマンドラインで **ssh-agent** を単独で実行すると、必須の環境変数を構成する方法がディスプレイに表示されます。 ただし、これらの必須の変数はまだ構成が済んでいません。 環境変数を構成するために、表示されたテキストをコピーし、それをコマンドラインに貼り付けてから、コマンドを実行できます。この作業を行うまで、**ssh-add** を使用することはできません。`eval` または `$SHELL` を使用する時は (前述の例を参照)、この追加の手順を実行する必要はありません。
- X11 を使用する場合は、「`< /dev/null`」を使用して **ssh-add** を呼び出し、**ssh-askpass** 確認ウィンドウをアクティブ化します。 このウィンドウはパスフレーズの入力要求に使用されます。
- X.509 証明書に関連付けられた秘密鍵を使用している場合は、**ssh-add -x** オプションを使用して、これらの鍵を鍵エージェントに追加します。`ssh-add -x`

ユーザの証明書認証

クライアント認証に証明書を使用することで、公開鍵認証で生じるいくつかの問題を解決できます。 公開鍵認証では、各クライアントが毎回、サーバに公開鍵のコピーをアップロードする必要があります。 証明書認証では、信頼されたサードパーティである認証局 (CA) を使用してクライアントからの受信情報の有効性を確認するため、この問題を回避できます。 証明書を使うと、クライアントの複数の一意の公開鍵ではなく単一のトラストアンカを使用して認証を構成することができます。

注意: Reflection PKI Services Manager は PKI 設定の一元管理に対応しています。 PKI Services Manager のインスタンスを 1 つインストールして構成し、対応する Attachmate 製品すべてに対して証明書検証サービスを提供することができます。

要件

要件	機能
Reflection PKI Services Manager がインストールされ、適正に構成されていること	PKI Services Manager は、証明書を検証してから、マップファイルを使用し、有効な証明書により認証可能なユーザを判別します。証明書の検証に成功するには、少なくとも 1 つのトラストアンカと 1 つのマッピングルールを構成する必要があります。さらに、中間証明書と証明書破棄情報へのアクセスも構成する必要がある場合があります。
CA によって署名された証明書とそれに関連付けられた秘密鍵がクライアントにインストールされていること	クライアントは、ユーザの認証のため、この証明書をサーバに送信します。
Reflection for Secure IT サーバが、PKI Services Manager 公開鍵のコピーを保持しており、PKI Services Manager に接続するよう構成されていること	サーバは、PKI Services Manager と通信することで、ユーザ証明書の有効性を確認します。

ユーザの証明書認証は以下のようにして行われます。

1. Reflection for Secure IT クライアントは、ユーザ認証のためサーバへ証明書を提示します。
2. Reflection for Secure IT サーバは Reflection PKI Services Manager に接続します (Reflection for Secure IT サーバの **PkidAddress** キーワードを使用してこの接続用のサーバ名とポートを設定します)。
3. Reflection for Secure IT は、インストールされている公開鍵を使用して PKI Services Manager の ID を確認します(Reflection for Secure IT サーバの **PkidPublicKey** キーワードを使用して鍵の名前と場所を設定します)。
4. Reflection for Secure IT は証明書とユーザ名を PKI Services Manager に送信します。
5. PKI Services Manager は、証明書が有効かどうかを判断し、PKI Services Manager 管理者によって PKI Services Manager マップファイル (既定では `/opt/attachmate/pkid/config/pki_mapfile`) に構成されているルールに基づき、この証明書を使ってユーザの認証が許可されるかどうかを決定します。この情報が Reflection for Secure IT に返されます。

6. 証明書が有効であり、その証明書を提示したユーザがこの証明書で許可された ID を示している場合、Reflection for Secure IT サーバは、ユーザのデジタル署名を検証して、このユーザの証明書に含まれる公開鍵に関連付けられた秘密鍵がクライアントによって処理されたことを確認します。デジタル署名が確認されると、ユーザ認証は成功です。

ユーザの証明書認証の構成

開始する前に、ユーザの証明書認証 『54ページ 』 に記載された要件を確認してください。

証明書を使用したユーザ認証を構成するには、Reflection PKI Services Manager をインストールおよび構成して、サーバおよびクライアントを構成する必要があります。構成を開始するには、以下の手順を使用しますが、そのほかに、多くの異なる手順を実行できます。詳細は、「Reflection PKI Services Manager ユーザガイド (http://support.attachmate.com/manuals/pki.html)」を参照してください。

PKI Services Manager の単一のインスタンスをインストールおよび構成することで、複数の Reflection for Secure IT クライアントおよびサーバからの証明書認証要求に対応できます。ただし、Reflection for Secure IT 設定では PKI Services Manager のアドレスおよびポートとしてただ 1 つのエントリのみが許可されるため、この構成において単一障害ポイントが発生する可能性があります。PKI Services Manager にアクセスできないか、サーバが稼働中でない場合、証明書を使用したすべての認証試行は失敗します。負荷分散およびフェイルオーバ機能を実現するには、PKI Services Manager ホスト名としてラウンドロビン DNS エントリを定義するか、負荷分散機能を持つサーバに PKI Services Manager ホストを配置することができます。PKI Services Manager が Microsoft クラスタ環境で実行されるよう構成することもできます。

注意: 以下に示すパスは、既定のインストールオプションに基づいています。

PKI Services Manager を インストールおよび構成するには

1 Reflection PKI Services Manager サーバに root としてログインします。

2 Reflection PKI Services Manager 『18ページ 』 をインストールします。

3 トラストアンカとして指定する 1 つまたは複数の証明書のコピーを証明書ストアに入れます。既定の PKI Services Manager ストアは以下の場所にあります。

```
/opt/attachmate/pkid/local-store
```

4 テキストエディタで PKI Services Manager 構成ファイルを開きます。既定の名前と場所は次の通りです。

```
/opt/attachmate/pkid/config/pki_config
```

5 **TrustAnchor** キーワードを使用してトラストアンカーを識別します。例:

```
TrustAnchor = trustedca.crt
```

または

```
TrustAnchor = CN=SecureCA,O=Acme,C=US
```

注意: 複数のトラストアンカーを構成するには、**TrustAnchor** の行を追加します。

6 証明書取り消しチェックを構成します。例:

対象	サンプル構成
LDAP サーバに格納されている CRL を使用します。	`RevocationCheckOrder = crlserver` `CRLServers=ldap://crlserver`
OCSP レスポンダを使用します。	`RevocationCheckOrder = ocsp` `OCSPResponders = http://ocspresponder`

注意: 既定では、PKI Services Manager は、ローカルストアで CRL を探します。この構成を使用する場合、ローカルストアに CRL をコピーする必要があります。

7 証明書の信頼のチェーンで中間証明書が必要な場合は、これらの証明書へのアクセスを構成します。例:

対象	サンプル構成
ローカルストアに追加した中間証明書を使用します。	`CertSearchOrder=local`
LDAP サーバに格納されている証明書を使用します。	`CertSearchOrder=certserver` `CertServers=ldap://ldapserver`

8 変更内容を構成ファイルに保存します。

9 テキストエディタで PKI Services Manager マップファイル 『177ページ』を開きます。既定の名前と場所は次の通りです。

`/opt/attachmate/pkid/config/pki_mapfile`

10 ユーザの **RuleType** スタンザを作成して、有効な証明書で認証可能なユーザを定義する 1 つまたは複数のルールを追加します。たとえば、以下のようになります。

```
RuleType = user
  { %UPN.user% } UPN.host Equals "acme.com"
  { fred root } Subject.CN Contains "Fred"
```

詳細なルールのサンプルについては、「PKI Services Manager のマッピングルールのサンプル 『183ページ』」を参照してください。

注意: 証明書が有効であると判明した後は、ルールが順番に (ルールの種類を基準に使用した後で、順序に従う) が処理されます。 証明書が、条件式に定義された要件を満たす場合 (または、ルールに条件が含まれていない場合)、そのルールに指定された、許可される ID の認証が許可されます。 最初の一致が検出された後は、追加のルールは適用されません。

11 有効な PKI Services Manager 構成をテストします。

`/usr/local/sbin/pkid -k`

エラーはありません。構成は有効です。

12 Reflection PKI Services Manager を再起動します。

`/usr/local/sbin/pkid` の再起動

Reflection for Secure IT サーバを構成するには

1 PKI Services Manager が、Reflection for Secure IT サーバと同じホスト上にインストールされていない場合は、PKI Services Manager 公開鍵を Reflection for Secure IT サーバにコピーします。

PKI Services Manager の鍵の場所は以下のとおりです。

```
/opt/attachmate/pkid/config/pki_key.pub
```

この鍵を Reflection for Secure IT ホスト上の任意の場所へコピーします。たとえば、以下のようになります。

```
/etc/ssh2/pki_key.pub
```

注意: この鍵ファイルは、所有者が root である必要があり、root 以外のユーザは書き込むことができません。

2 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

3 **PkidPublicKey** を編集して、PKI Services Manager 公開鍵を保存した場所を指定します。たとえば、以下のようになります。

```
PkidPublicKey=/etc/ssh2/pki_key.pub
```

4 **PkidAddress** を編集して、PKI Services Manager ホストおよびポートを指定します。たとえば、以下のようになります。

```
PkidAddress=pkiserver.acme.com:18081
```

注意: ホストを指定してポートを省略した場合、既定の PKI Services Manager ポート (18081) が使用されます。

5 公開鍵認証を許可または要求するよう **AllowedAuthentications**/**RequiredAuthentications** を構成します。以下に示す既定値では、公開鍵認証を許可しますが、要求しません。

```
AllowedAuthentications=gssapi-with-mic,publickey,keyboard-interactive,
password

RequiredAuthentications=
```

Reflection for Secure IT クライアントを構成するには

1 ユーザ証明書および関連する秘密鍵を取得します。 『41ページ 』

2 証明書および秘密鍵をインストールします。たとえば、以下のようになります。

```
~/.ssh2/userkey

~/.ssh2/userkey.crt
```

注意: 証明書は、秘密鍵と同じディレクトリ内にあり、同じベース名に `.crt` ファイル拡張子が付けられた名前である必要があります。

3 ユーザのみの読み取り専用アクセスが可能なようにユーザ鍵の権限を設定します。

```
chmod 400 userkey
```

4 クライアントの ID ファイルを作成 (または編集) します。
(既定は `~/.ssh2/identification` です。)ユーザのみの書き込みアクセスが可能なようにこのファイルを構成します。

```
chmod 600 identification
```

5 クライアントID ファイルに、秘密鍵を識別する行を追加します。**CertKey** キーワードを使用します(鍵が ~/.ssh2/ ディレクトリ内にある場合、パス情報はオプションです)。たとえば、以下のようになります。

```
CertKey userkey
```

6 テキストエディタでクライアント構成ファイル (/etc/ssh2/ssh2_config) を開きます。

7 クライアントの設定の構成で、**AllowedAuthentications** に publickey が含まれていること、および **IdentificationFile** に、手順 3 で構成したファイルが指定されていることを確認します。既定値を以下に示します。

```
AllowedAuthentications=gssapi-with-mic,publickey,keyboard-interactive,
password
```

```
IdentificationFile=~/.ssh2/identification
```

PAM (プラグ可能な認証モジュール)

キーボード対話型認証とともにプラグ可能な認証モジュール (PAM) を使用するように Reflection for Secure IT サーバを構成することができます。PAM では、認証関連サービスを提供する、実行時にプラグ可能なモジュールが使用されます。これらのモジュールは、認証、アカウント管理、セッション管理、パスワード管理の 4 つの種類に分けられます。

PAM が構成されると、Reflection for Secure IT により、認証のコントロールが PAM ライブラリへ転送されます。 PAM ライブラリは、PAM 構成ファイルに指定されたモジュールを読み込んでから、Reflection for Secure IT に対して、認証が正常終了したことを確認するように求めます。

以下のサーバキーワードによって、サーバでの PAM 認証が構成されます。

サーバキーワード	構成情報
AuthKbdInt.Required	PAM を認証およびパスワード管理に使用します。 `AuthKbdInt.Required=pam`
AccountManagement	PAM をアカウント管理に使用します。 `AccountManagement=pam`
UsePamSessions	PAM をセッション管理に使用します。 `UsePamSessions=yes`
PamServiceName	PAM サービスの名前を指定します。既定値は以下のとおりです。 `PamServiceName=ssh`
PamServiceNameForInternal Processes	PAM サービスを内部処理で使用するように指定します。たとえば、以下のようになります。 `PamServiceNameForInternalProcesses ssh-shell`

PamServiceNameForSubsystems	PAM サービスをサブシステムで使用するように指定します。たとえば、以下のようになります。 `PAMServiceNameforSubsystems sftp ssh-sftp`

PAM 認証の構成

PAM が構成されると、Reflection for Secure IT により、認証のコントロールが PAM ライブラリへ転送されます。

サーバ側で PAM 認証を構成するには

1 必須モジュール*auth*、*account*、*password*、および *session* に対応するように PAM 構成設定を編集します。必須モジュールが定義されていない場合、接続は拒否されます。

Linux システムでは、以下のファイルがサーバとともにインストールされます。

```
/etc/pam.d/ssh
```

このファイルには、既定の構成情報が含まれます。例えば、SLES システム上の `ssh` ファイルには以下が含まれます。

```
#%PAM-1.0
auth     include        common-auth
auth     required       pam_nologin.so
account  include        common-account
password include        common-password
session  include        common-session
```

その他のシステムでは、`/etc/pam.conf` を作成 (または構成) します。たとえば、HP-UX では、以下のようになります。

```
ssh auth     required  /usr/lib/security/libpam_unix.1
ssh account  required  /usr/lib/security/libpam_unix.1
ssh password required  /usr/lib/security/libpam_unix.1
ssh session  required  /usr/lib/security/libpam_unix.1
```

2 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

3 **AllowedAuthentications** (または **RequiredAuthentications**) に、許可されている認証方法として `keyboard-interactive` が含まれている (既定値) ことを確認します。

4 PAM サービスの名前を識別するように **PamServiceName** を構成します。

- PAM モジュールが `/etc/pam.d/ssh` に定義されている場合は、既定値 (`ssh`) を使用します。

  または、

- PAM モジュールが `pam.conf` に定義されている場合、**PamServiceName** の値は、使用しているサービス名 (上記の例では `ssh`) と一致していなければなりません。`ssh` が `pam.conf` に定義されていない場合は、既定のサービス名 `other` を使用できます。

5 PAM を使用するようにサーバを構成します。

PAM の用途	サーバ構成ファイルに追加する情報
認証およびパスワード管理	`AuthKbdInt.Required=pam`
アカウント管理	`AccountManagement=pam`
セッション管理	`UsePamSessions=yes`

6 (オプション) 認証時にクライアントユーザに表示されるプロンプトに「PAM authentication」という語句を含めるには、以下を含めます。

```
AuthKbdInt.Verbose=yes
```

クライアント側で PAM 認証を構成するには

- **AllowedAuthentications** に、許可されている認証方法として `keyboard-interactive` が含まれている（既定値）ことを確認します。

RADIUS 認証

RADIUS は、UNIX パスワードファイル、Active Directory、LDAP、ユーザ/パスワードのペアを含む単純なテキストファイルなどのパスワードデータベースとの統合によって、ユーザを認証する認証/許可/清算サービスです。Reflection for Secure IT は認証用途でのみ RADIUS に対応しています。

要件

1 台以上の RADIUS 認証サーバを構成する必要があります。Reflection for Secure IT を構成する場合は、RADIUS サーバの名前、RADIUS 通信に使用されるポート (通常、1812 または 1645)、および RADIUS サーバで使用される共有秘密が必要です。 この情報は、RADIUS 構成ファイルを作成するために使用します。

RADIUS 認証は以下のようにして行われます。

Reflection for Secure IT サーバは、ユーザを認証する目的で RADIUS クライアントとして機能します。要求は、RADIUS ファイルに構成されたすべての RADIUS サーバに送信されます。

1. Reflection for Secure IT サーバは、クライアントからキーボード対話型認証要求を受信します。
2. RADIUS 認証が有効になっている場合、Reflection for Secure IT サーバは、User-Name および Password 属性/値ペアを含む ACCESS-REQUEST メッセージを、構成した最初の RADIUS サーバへ送信することでユーザの認証を試行します。
3. Reflection for Secure IT サーバは、RADIUS 認証サーバからの ACCESS-ACCEPT または ACCESS-REJECT メッセージを待機します。

4. Reflection for Secure IT サーバが ACCESS-ACCEPT メッセージを受信した場合、クライアント接続が許可され、Reflection for Secure IT サーバで、現在のサーバ構成に基づくユーザアクセスが可能になります。 サーバが ACCESS-REJECT メッセージを受信したか、応答の受信に失敗した場合、サーバは、構成済みのその他の RADIUS サーバに対して認証を試みます。 ACCESS-ACCEPT メッセージがどの RADIUS サーバからも受信されない場合、RADIUS 認証は失敗し、Reflection for Secure IT サーバは、ほかの許可される認証を試行します。

   注意: RADIUS認証サーバに対してユーザの認証が可能であっても、そのユーザ用のアカウントが Reflection for Secure IT サーバ上に存在しない場合、認証は失敗します。

RADIUS 認証の構成

RADIUS が構成されると、Reflection for Secure IT により、認証のコントロールが RADIUS 認証サーバへ転送されます。

Reflection for Secure IT サーバを構成するには

1 以下のファイルを作成し、所有者のみの読み取り/書き込みアクセス権 (権限 = 600) を設定します。

```
/etc/ssh2/radius_config
```

2 テキストエディタでこのファイルを開きます。 RADIUS サーバごとに、当該サーバ、そのサーバ上で RADIUS に使用されるポート、およびそのサーバに対する認証のために RADIUS クライアントで必要となる共有秘密を識別する行を追加します。例えば、以下のようになります。

```
server1:1812:secret1

server2:1812:secret2
```

   注意: RADIUS サーバは、認証要求に対する応答が受信されるまで上から下へ順番に確認されます。

3 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。以下のキーワードを編集します。

```
AllowedAuthentications=keyboard-interactive

AuthKbdInt.Required=radius

RadiusFile=/etc/ssh2/radius_config
```

クライアントを構成するには

- キーボード対話型認証を有効にします。(これは、Reflection for Secure IT クライアントの既定値です。)

RSA SecurID 認証

RSA SecurID は、ハードウェアトークンまたはソフトウェアトークンをベースとする、RSA Security, Inc 社の 2 要素認証ソリューションです。 SecurID の使用前に、認証マネージャのドキュメントを確認するようお勧めします。

Reflection for Secure IT は PAM を使用する RSA SecurID 認証に対応しています。

必須要素	機能
RSA 認証マネージャ	認証要求を確認し、認証ポリシを中心的に管理します。
RSA 認証エージェント	認証要求を受け取り、認証のため、それらの要求を認証マネージャに送信します。
	注意: PAM 用 RSA 認証エージェントは、Reflection for Secure IT サーバと同じコンピュータ上で動作している必要があります。
ハードウェアトークン	ワンタイム認証コードを生成する、鍵フォッブや PIN カードなどのハードウェアデバイス。
PAM 用 RSA 認証エージェント	認証のコントロールを RSA へ転送します。

SecurID 認証の構成

Reflection for Secure IT は、PAM 用 RSA 認証エージェントに対応しています。このエージェントを使用すると、サーバへの接続時に RSA SecurID トークンの使用が可能になります。 PAM 用 RSA 認証エージェントは、Reflection for Secure IT サーバと同じホスト上で動作している必要があります。

クライアントを構成するには

- キーボード対話型認証を有効にします。(これは、Reflection for Secure IT クライアントの既定値です。)

サーバを構成するには

1 Reflection for Secure IT サーバを実行しているコンピュータ上に RSA 認証エージェントをインストールします。

2 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

3 キーボード対話型認証を有効にして、認証およびパスワード管理に PAM を使用するようサーバを構成します。

```
AllowedAuthentications=keyboard-interactive

AuthKbdInt.Required=pam
```

サーバを起動するには

注意: Secure Shell サーバを起動する前に環境変数 VAR_ACE および LD_LIBRARY_PATH を設定する必要があります。 VAR_ACE を、sdconf.rec ファイルが含まれている PAM 用 RSA エージェントのインストールディレクトリに設定します。 LD_LIBRARY_PATH を、RSA/サーバまたは RSA/エージェントがインストールされているディレクトリに設定します。

- 環境変数を設定し、サーバを起動するには

```
$ VAR_ACE=/opt/ace/data LD_LIBRARY_PATH=/opt/ace/prog /usr/sbin/sshd2
```

注意: 再起動後も環境変数の変更を維持するには、サーバ起動スクリプト『21ページ』を変更するか、root ユーザの既定のプロファイルを変更できます。

HP-UX 高信頼性システムでのアカウント管理の構成

Reflection for Secure IT において、HP-UX 高信頼性システムによるログイン規制が正しく遵守されるようにするには、PAM アカウント管理を使用するようサーバを構成する必要があります。

注意: HP-UX 高信頼性システムを使用して、許可されたユーザのみが接続できるようにするには、PAM アカウント管理の構成が必要です。PAM アカウント管理を使用するようサーバが構成されていないと、状況によっては、アクセスが拒否される必要があるユーザがシステムへアクセスできるようになる場合があります。

サーバ上に PAM アカウント管理を構成するには

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

2 アカウント管理に PAM を使用するようサーバを構成します。

```
AccountManagement=pam
```

注意: PAM アカウント管理に対応するには、ホストシステム上で PAM が適正に構成されている必要があります。

第 7 章

安全なファイル転送

この章の内容

Reflection for Secure IT は、安全なファイル転送に使用可能なプログラム **sftp** および **scp** に対応しています。いずれのコマンドも、コンピュータ間でのファイル転送を効率的かつ安全に実行します。

sftp を使用すると、ローカルコンピュータとリモートホスト間で安全なファイル転送を行うことができます。また、ファイルの作成やパーミッションの変更など、ほかのファイル管理コマンドを実行することもできます。**sftp** は、対話型で使用することも、操作を自動化するためにバッチファイルと組み合わせて使用することもできます。

scp を使用すると、ローカルコンピュータとリモートホスト間または 2 つのリモートホスト間で安全なファイルコピーを行うことができます。

安全なファイル転送 (sftp)

安全なファイル転送 (**sftp**) は、**ftp** の安全な代替機能として使用できます。**sftp** は、対話型で使用することも、自動化された安全なファイル転送のためにバッチファイルと組み合わせて使用することもできます。

sftp は、**ssh** によって提供された認証および暗号化を使用するため、リモートコンピュータ上で Secure Shell サーバが実行されていなければなりません。**sftp** 接続の設定は、**ssh** クライアント構成ファイルによって制御されます。これらの設定の詳細については、「クライアント構成キーワード 『107ページ 』」を参照してください。また、**sftp** コマンドライン上で **-o** オプションを使用して設定を構成することもできます。

注意: コマンドラインオプションは、構成ファイルの設定を置き換えます。

コマンドラインオプションの詳細については、「sftp コマンドラインオプション 『152ページ 』」を参照してください。**sftp** コマンドリファレンスについては、「対応している sftp コマンド 『155ページ 』」を参照してください。

対話型 sftp の使用

対話型 sftp セッションを使用して、リモートコンピュータ上で 1 つまたは複数のファイル管理コマンドを安全に実行することができます。

対話型 sftp セッションを開くには、以下の手順に従います。

1 リモートホストへ接続します。たとえば、以下のようになります。

```
sftp joe@myhost.com
```

注意: Secure Shell サーバ上で使用している名前が現在のユーザ名と同じである場合は、ユーザ名を省略できます。

接続の確立が成功すると、以下のプロンプトが表示されます。

```
sftp>
```

2 以下のいずれかの操作を実行します。

操作	使用コマンド
対応しているコマンドのリストの表示	**help** 例えば、以下のように入力します。 `sftp> help`
対応しているコマンドに関する詳細の表示	**help** *command* 例えば、以下のように入力します。 `sftp> help put`
ファイルの転送および管理	使用可能なコマンド 『155ページ』 例えば、ファイル `demo` をローカルの作業ディレクトリからリモートの作業ディレクトリに転送するには、以下のように入力します。 `sftp> put demo`
セッションの終了	**quit** 例えば、以下のように入力します。 `sftp> quit`

注意: ホスト名を指定せずに **sftp** を起動した場合は、**open** と **close** の対話型コマンドを使用して、対話型セッション中に 1 つまたは複数のホストに接続できます。

sftp バッチファイルの実行

sftp バッチファイルを使用すると、安全な方法でファイル管理を自動化できます。

sftp バッチファイルを作成および実行するには

1 GSSAPI や、パスフレーズ保護なしの公開鍵など、非対話型クライアント認証方法に対応するようにクライアントを構成します。

注意: **sftp** のバッチファイルオプション (**-B**) は、対話処理が必要な認証方法には対応していません。

2 クライアントで、**sftp** バッチファイルを作成してください。バッチファイルでは、**sftp** 『155ページ』 で使用可能な任意の対話型コマンドが使用できます。たとえば、以下のようなコマンドによって `demo` というファイルを作成できます。

```
get path/file1
get webfiles/*.htm
```

3 **sftp** を使用して、リモートホストに接続し、バッチファイルを実行します。たとえば、以下のようになります。

```
sftp -B demo myname@myhost.com
```

クライアントは、バッチファイル内のコマンドを実行して、終了します。

注意:

- ログインが正常に行われると、**sftp** によってバッチファイル内のコマンドが順に実行され、**bye**、**exit**、または **quit** コマンドが検出されると接続が終了します。
- バッチファイル内のコマンドの実行に失敗しても、**sftp** は残りのコマンドの実行を続行し、最初に失敗したコマンドのエラーコードを返します。ただし、接頭語に「-」（ダッシュ）が付いたコマンドは、失敗しても常に 0 を返します。転送コマンドごとに個別のエラーを返すバッチファイルを構成する方法は、**scp** を参照してください。

sftp 転送方式 (ASCII またはバイナリ) の構成

SFTP は 2 つの転送方式に対応しています。 ASCII およびバイナリです。 ASCII モードでは、個々の英字、数字、および文字は、ASCII 文字コードを使用して転送され、受信側のコンピュータで、そのシステムに対して適正なテキスト形式で保存されます。 UNIX と Windows コンピュータ間での ASCII 転送では、改行文字は各システム用に適宜変換されます。 (必要に応じて、改行文字の変換を手動で構成することもできます。) バイナリ転送では、データはバイト単位でサーバに転送され、データの変換は行われません。

Reflection for Secure IT では、さらに、*auto* というスマート転送オプションを使用できます。 自動モードでは、転送方式はファイル拡張子によって決まります。 指定されたファイル拡張子を持つファイルには ASCII 転送が使用され、それ以外のファイルにはバイナリ転送が使用されます。 ASCII ファイルの種類の既定の一覧は「`txt, htm*, pl, php*`」です。 指定の **sftp** セッション用にこの一覧を変更するには、**setext** コマンドを使用します。 既定のファイル拡張子一覧を変更するには、クライアントキーワード **FileCopyAsciiExtensions** を使用します。

sftp コマンド	機能
`ascii -s`	現在の転送モードを表示します。
`binary`	現在の転送モードをバイナリにします (既定)。
`auto`	現在の転送モードを自動に設定します。
`getext`	自動モードが有効になっている場合に ASCII ファイル転送を使用するファイル拡張子の現在の一覧を表示します。

sftp コマンド	機能
`setext`	自動モードが有効になっている場合に ASCII ファイル転送を使用するファイル拡張子の現在の一覧を指定します。 複数の拡張子を指定するには、カンマまたは空白区切りの一覧を使用します。 このコマンドでは値は累積されません。 ワイルドカード (zsh-glob) 文字を指定できます。 ファイル拡張子の前にピリオドを付けてはなりません。 空白を含む拡張子を指定するには、その拡張子を引用符で囲むか、エスケープ文字としてバックスラッシュを使用します。
`ascii`	現在の転送モードを ASCII に設定します。 リモートの改行コードが明示的に指定されている場合、クライアントは、サーバから改行変換を取得しようと試みます。 サーバがこの機能に対応していない場合、クライアントはリモートの改行コードを CRLF に設定します。
`ascii dos`	リモートの改行コードを CRLF に設定します。
`ascii unix`	リモートの改行コードを LF に設定します。

クライアントキーワード	機能
FileCopyAsciiExtensions	自動モード転送が有効になっている場合に ASCII ファイル転送用のファイルの種類の既定の一覧を指定します。 既定は「txt, htm*, pl, php*」です。

安全なファイルのコピー (scp)

scp を使用すると、ローカルコンピュータとリモートホスト間または 2 つのリモートホスト間で安全なファイルコピーを行うことができます。

転送元ファイル名と転送先ファイル名の両方には、ホストとユーザの指定を含めることができます。これは、ファイルが転送元ホストから転送先ホストへコピーされることを示すためです。2 つのリモートホスト間のコピーは許可されます。ワイルドカードに対応しています。繰り返しがオフの場合 (既定値)、名前の置換はファイル名のみに適用され、ディレクトリには適用されません。(**-r** を使用して) 繰り返しを有効にすると、名前の置換がファイルとディレクトリに適用されます。既定で既存のファイルは上書きされます。上書きの動作を制御するには、**--overwrite** を使用します(同じファイルについては、この設定値に関係なく、ファイル転送処理は行われません)。

scp は、**ssh** によって提供された認証および暗号化を使用するため、リモートコンピュータ上で Secure Shell サーバが実行されていなければなりません。**scp** 接続の設定は、**ssh** クライアント構成ファイルによって制御されます。これらの設定の詳細については、「クライアント構成キーワード 『107ページ 』」を参照してください。

また、**scp** コマンドライン上で **-o** オプションを使用して設定を構成することもできます。コマンドラインオプションは、構成ファイルの設定を置き換えます。

例

以下の例は、ローカルコンピュータとリモートホスト間または 2 つのリモートホスト間で安全なファイル転送を行うために **scp** をどのように使用できるかを示しています。

目的	例
リモートファイル (file1) を指定のローカルファイル (file2) および場所へ転送する	`scp joe@myhost:/source/file1 /destination/file2`
すべての *.htm ファイルをローカルコンピュータ上の現在の作業ディレクトリから myhost.com 上の joe の既定のディレクトリにコピーする	`scp *.htm joe@myhost.com:`
指定のファイルをリモートの host1 からリモートの host2 にコピーする	`scp joe@host1:/dir/scr_file joe@host2:/dir/dest_file` 注意: 認証は 2 回、必要になります。

スマートコピーおよびチェックポイント再開

Reflection for Secure IT UNIX クライアントおよびサーバは、繰り返される不要なデータ転送にかかる時間を最小限に抑えるための機能に対応しています。

同一のファイル (スマートファイルコピー)

クライアントユーザが転送を開始し、サーバ上に全く同じ名前のファイルがすでに存在している場合、サーバは、そのファイルのサーバコピーのハッシュを計算し、この値をクライアントに送信します。 クライアントはそのファイルのクライアントコピーのハッシュを計算し、その値をサーバから送信された値と比較します。 2 つのハッシュが同一である場合、これは、ファイルが同一のもので、データ転送が行われないことを示しています。 **scp -p** オプションを使用して転送する場合を除き、転送先ファイルのタイムスタンプが更新されます。

既定では、スマートファイルコピーが有効になっています。クライアントから無効にするには、**SmartFileCopy** を「no」に設定します。サーバから無効にするには、**SmartFileTransfer** を「no」に設定します。スマートファイルコピーを無効にすると、既存のファイルが上書きされます。

中断されたファイル転送の自動再開 (チェックポイント再開)

Reflection for Secure IT クライアントおよびサーバは、中断されたファイル転送を、転送が中断されたポイントから再開することができます。 例えば、接続がファイルのアップロード時に切断された場合、クライアントユーザはその転送を再開できます。 Reflection for Secure IT クライアントはサーバ上のファイルのサイズを判別し、そのファイルのハッシュをサーバに要求します。 クライアントは、サーバ上で保持されている長さになるまで、ローカルファイルのハッシュを計算します。 これらのハッシュが同じである場合に、転送がそのファイルの当該ポイントで再開されます。

注意: 行末が異なるシステム間での ASCII 転送の場合は、ファイルの比較のためのハッシュの計算で有効なデータは生成されません。したがって、この場合、ハッシュの比較は省略され、常に、ファイル全部が転送されます。

既定では、チェックポイント再開が有効になっています。クライアントから無効にするには、**CheckpointResume** を「no」に設定します。サーバから無効にするには、**SmartFileTransfer** を「no」に設定します。チェックポイント再開を無効にすると、中断後にファイル転送が開始されます。

注意: 混雑したネットワークでファイルを転送する場合、この機能を利用するメリットよりも、ネットワーク上でハッシュ値を送信するのに要する時間によるデメリットのほうが上回ることがあります。このような状況では、スマートコピーおよびチェックポイント再開機能を無効にすることにより、パフォーマンスを向上できる場合があります。

アップロードおよびダウンロードアクセスの構成

既定で、ユーザには、ユーザのログインアカウントで許可されるすべてのディレクトリへの完全なアクセス権が付与されます。**AllowSftpCommands** を使用すると、**sftp** および **scp** の使用時にユーザが実行可能な操作の種類を制限することができます。このキーワードには、カンマ区切りのリストとして、「**all**」、「**none**」、「**browse**」、「**download**」、「**upload**」、「**delete**」、「**rename**」の中から 1 つまたは複数の項目を指定できます。**upload** オプションを使用すると、ユーザは、サーバ上でファイルの変更、ファイルの作成、ディレクトリの作成、ファイル属性の変更が可能になります。**download** オプションでは、ユーザはファイルの内容を読み込むことができます。

AllowSftpCommands によって、SFTP サブシステムを使用するコマンドからのアクセスを制御します。これには、Reflection for Secure IT クライアントからの **scp** と **sftp** の両コマンドと、OpenSSH クライアントからの **sftp** コマンドが含まれます。このキーワードは OpenSSH クライアントからの **scp** コマンドには適用されません。OpenSSH **scp** コマンドは SFTP サブシステムを使用せず、安全なチャネルを介して **rcp** コマンドを実行するからです。

注意: クライアントユーザは、数種類の方法でサーバファイルおよびディレクトリにアクセスできます。 サーバを構成する際に考慮すべき要素としては、システム上に構成されているセッションアクセス、トンネリングアクセス、ファイルおよびディレクトリ権限などがあります。

アップロードおよびダウンロードの権限を構成するには

注意: このキーワードによる変更は **scp** 転送と **sftp** 転送の両方に影響を与えます。

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。(このキーワードをサブ構成ファイルで構成することもできます。)

2 **AllowSftpCommands** キーワードを編集します。例えば、以下のようになります。

ユーザにファイルの表示とダウンロードを許可するが、サーバファイルへの変更は許可しないようにするには

```
AllowSftpCommands = browse, download
```

ユーザにファイルの参照とアップロードを許可するが、サーバ上のファイルの内容の表示は許可しないようにするには

```
AllowSftpCommands = browse, upload
```

3 端末セッションまたはリモートでのコマンド実行 (OpenSSH **scp** を含む) によるファイルへのアクセスを防ぐには、**SessionRestricted** キーワードを使用できます。

```
SessionRestricted = subsystem
```

ダウンロード済みファイルのファイル権限の設定

sftp または **scp** を使用してクライアントへファイルをダウンロードした場合、そのダウンロード済みファイルのファイル権限は、クライアントの構成とソースファイルの権限の両方によって決まります。

ファイルがすでにクライアント上に存在する場合

- クライアントファイルの権限は転送後もそのまま維持されます。転送によって、ファイルの内容は更新されますが、既存のファイル権限は変更されません。

ファイルがクライアント上に存在しない場合、転送されたファイルに設定される権限は以下の要因の影響を受けます。

- ダウンロード済みファイルにはソースファイルと同じ権限が与えられますが、ただし、それらの権限を持つファイルを作成できないようにする設定がクライアント上で有効になっていない場合に限られます。
- 新規に作成されたファイルの権限を制限するローカル設定が有効になっている場合は、それらの設定が、ダウンロード済みファイルに適用されます。 これらの設定は、グローバルに構成することも、**umask** コマンドを使用して現在のセッション用に変更することもできます。

umask を使用してダウンロード済みファイルの権限を設定するには

1 新規に作成されたファイルに対して必要な制限を指定するには **umask** を使用します。 例えば、以下の同じ意味のコマンドのいずれかを使用して、新しいファイルをユーザのみが読み取り/書き込みできるように制限できます。

```
$ umask 066
```

または、

```
$ umask u=rwx,g=x,o=x
```

2 サーバに接続して、**sftp** または **scp** のいずれかを使用してダウンロードします。

上記の **umask** のサンプルでは、ダウンロード済みのファイルが、グループまたはその他のアクセス権なしでクライアント上に作成されます。

以下のセッションは、**sftp** を使用してダウンロードされたファイルに権限を設定するための **umask** の使用例を示しています。 1 つ目のファイル (`file1`) は、ユーザ、グループまたはその他のアクセス権なしで、サーバ上で読み取り/書き込み (666) できるようにします。 2 つ目のファイル (`file2`) は、ユーザがサーバ上で読み取り/書き込みできるようにし、グループまたはその他のアクセス権なしに対しては、読み取り (644) できるようにします。 ダウンロード後は、どちらのファイルも、ユーザのみがクライアント上で読み取り/書き込み (600) できるようにします。

```
$ umask 066

$ sftp joe@myserver.com

Authentication successful.

sftp> ls -l file1

-rw-rw-rw-    0 joe   users     108 Sep 30 02:52 file1

sftp> get file1

/home/joe/file1                   108  0.0KB/s  00:00  100%

sftp> lls -l file1

-rw-------    0 joe   users     8 Sep 30 11:47 file1

sftp> ls -l file2

-rw-r--r--    0 joe   users     225 Sep 30 02:56 file2
```

```
sftp> get file2
/home/joe/file2                    225  0.0KB/s  00:00 100%
sftp> lls -l file2
-rw-------    0 joe   users     225 Sep 30 11:47 file2
sftp> exit
$
```

アップロード済みファイルのファイル権限の設定

ここに記載されているオプションは、**sftp** または **scp** を使用してアップロードしたファイルの権限に影響します。

ForceSftpFilePermissions を使用して権限を設定する

ForceSftpFilePermissions は、**sftp** または **scp** を使用してサーバにアップロードされるすべてのファイルのファイル権限値の指定に使用できるサーバキーワードです。この設定は、他のすべての権限設定の動作を上書きします。3 桁の許可モード値を使用します。次に例を示します。

```
ForceSftpFilePermissions=600
```

この例では、アップロードするすべてのファイルが 600 に設定されます (`-rw-------`)。また、既存ファイルに対する権限を変更しようとすると、クライアントユーザが要求する権限値に関係なく、そのファイルも 600 に設定されます。

ForceSftpFilePermissions が構成されている場合は、次のようになります。

- ファイルを新規作成するか既存ファイルに上書きするかにかかわらず、アップロードするファイルはすべて指定値に設定されます。
- システムの `UMASK` 設定は無視されます。
- **sftp** ユーザが実行する **chmod** コマンドでは、ユーザが指定する値は無視されます。また、ファイルの権限は **ForceSftpFilePermissions** が設定する値に変更されます。
- **-p** オプションは、**sftp** および **scp** コマンドラインで使用される場合は無視されます。

次のセッションでは、joe というユーザが demoserver.com に接続しています。ここでは、**ForceSftpFilePermissions** は 600 に設定されています。Joe のローカルファイル `sample.txt` の権限は 666 です。転送後、サーバ上の `sample.txt` のファイル権限は 600 に設定されます。Joe は、サーバ上の自分のファイル `test.txt` の権限を一覧表示し、権限が 666 であることを確認します。Joe は、chmod で権限を 644 に変更しようとします。**ForceSftpFilePermissions** の値が Joe の設定した値より優先されるため、このコマンドの実行後にファイルの権限は 600 に設定されます。

```
joe@abchost:~> ls -l sample.txt
-rw-rw-rw- 1 joe users 9668 2011-06-18 17:41 sample.txt
joe@abchost:~> sftp demoserver.com
Authentication successful.
sftp> put sample.txt
sample.txt                    668     9.4KB/s   00:00     100%
sftp> ls -l sample.txt
```

```
-rw-------    1 joe    joe   9668 Mar 18  2011 sample.txt
sftp> ls -l test.txt
-rw-rw-rw-    1 joe    joe   73 Jan 15 17:49 test.txt
sftp> chmod 644 test.txt
Changing mode on /home/joe/test.txt
sftp> ls -l test.txt
-rw-------    1 joe    joe    73 Jan 15 17:49 test.txt
sftp>
```

システム設定を使用して権限を制御する

ForceSftpFilePermissions が構成されていない場合、アップロードされるファイルの権限はシステム設定によって決定されます。

ファイルがすでにサーバ上に存在する場合

- サーバファイルの権限は転送後もそのまま維持されます。転送によって、ファイルの内容は更新されますが、既存のファイル権限は変更されません。

ファイルがサーバ上に存在しない場合、転送されたファイルに設定される権限は以下の要因の影響を受けます。以下の一覧では、下方の項目が、上方の項目より優先されます。

1. アップロード済みファイルにはソースファイルと同じ権限が与えられますが、ただし、それらの権限を持つファイルを作成できないようにする設定がサーバ上で有効になっていない場合に限られます。
2. クライアントが **SetRemoteEnv** キーワードを使用して UMASK 値を要求している場合は、それらの権限による制限が適用されます。
3. ファイルの新規作成に関するシステム全体の設定が適用されます。(例えば、これらの設定は /etc/default/login や /etc/environment などの標準のシステムファイルで構成されていたり、PAM を使用して構成されていたりします。)
4. UMASK 値がグローバルな Reflection for Secure IT 環境ファイル (`/etc/ssh2/environment`) で構成されている場合は、それらの権限による制限が適用されます。
5. UMASK 値がユーザ固有の Reflection for Secure IT 環境ファイル (`~/.ssh2/environment`) で構成されている場合は、それらの権限による制限が適用されます。

注意: UMASK は、既定で、**SettableEnvironmentVars** で許可される環境変数の一覧に含まれます。UMASK がこの一覧に含まれていない場合は、サーバ上の `environment` ファイルでも、クライアントキーワード **SetRemoteEnv** でも UMASK 値を変更することはできません。

環境ファイルを使用して、サーバ上のアップロード済みファイルに権限を設定するには

1 環境ファイルを作成 (または編集) します。

設定の種類	使用するパスおよびファイル名
ユーザ固有の設定	`~/.ssh2/environment`
グローバル設定	`/etc/ssh2/environment`

2 アップロード済みファイルに適用する UMASK 値を指定する行を追加します。例えば、以下のようになります。

```
UMASK=066
```

クライアントで SetRemoteEnv を使用して、アップロード済みファイルに権限を設定するには

- テキストエディタでクライアント構成ファイル (/etc/ssh2/ssh2_config) を開きます。 アップロード済みファイルに適用する UMASK 値を指定するための SetRemoteEnv の行を追加します。例えば、以下のようになります。

```
SetRemoteEnv=UMASK=066
```

または、

- コマンドラインで **SetRemoteEnv** を使用して UMASK 値を指定します。例えば、以下のようになります。

```
sftp -oSetRemoteEnv=UMASK=066 joe@myserver.com
```

以下のセッションは、**scp** を使用してアップロードされたファイルに権限を設定するための **SetRemoteEnv** の使用例を示しています。ソースファイル (demo) は、ユーザ、グループまたはその他のアクセス権なしで、クライアント「abchost」上で読み取り/書き込み (644) できるようにします。ダウンロード後、このファイルは、ユーザのみがサーバ「xyzhost」上で読み取り/書き込み (600) できるようにします。

```
joe@abchost:~> ls -l demo

-rw-r--r-- 1 joe users 30 2008-10-02 12:07 demo

joe@abchost:~> scp -oSetRemoteEnv=UMASK=066 demo joe@10.10.3.232:

Authentication successful.

demo                                  30     0.0KB/s   00:00     100%

joe@abchost:~> ssh joe@10.10.3.232

Authentication successful.

Last login:Thu Oct  2 16:56:22 2008 from 150.215.83.121

[joe@xyzhost ~]$ ls -l demo

-rw-------  1 joe joe 30 Oct  2 16:57 demo

[joe@xyzhost ~]$
```

第 8 章

ポート転送

この章の内容

トンネリングとしても知られるポート転送は、アクティブなセッションにおける Secure Shell チャネルを通じて通信をリダイレクトするための方法を提供します。ポート転送が構成されると、指定のポートへ送信されるすべてのデータは、保護されたチャネルを通じてリダイレクトされます。 以下のいずれかを構成できます。

- ローカルポート転送によって、Secure Shell クライアントホスト上で実行されているアプリケーションクライアントとリモートアプリケーションサーバ間で安全にデータを移動します。
- リモートポート転送によって、Secure Shell サーバホスト上で実行されているアプリケーションクライアントとローカルアプリケーションサーバ間で安全にデータを移動します。
- FTP 転送によって、すべての FTP 通信を Secure Shell トンネルを介して転送できます。
- X11 転送によって、Secure Shell クライアントホスト上で実行されている X サーバと、Secure Shell サーバホスト上で実行されている X クライアント間で安全に X プロトコルデータを移動します。これは、特別な種類の動的リモートポート転送であり、各種設定を使用して構成されます。

用語集

ポート転送には、クライアントアプリケーションとサーバアプリケーションの 2 つのセット (Secure Shell クライアントとサーバ、およびデータが転送されるクライアント/サーバのペア) が必要です。このガイドでは、ポート転送に関連して以下のように定義された用語を使用します。

用語	定義
Secure Shell サーバ	Reflection for Secure IT サーバデーモン
Secure Shell サーバホスト	Secure Shell サーバを実行しているコンピュータ
Secure Shell クライアント	Reflection for Secure IT クライアントアプリケーション
Secure Shell クライアントホスト	Secure Shell クライアントを実行しているコンピュータ
アプリケーションクライアント	そのデータを転送したいクライアント/サーバのペアのクライアントプログラム。例えば、メールクライアントまた

	は Web ブラウザです。
アプリケーションクライアントホスト	アプリケーションクライアントを実行しているコンピュータ。通常は Secure Shell サーバホストまたは Secure Shell クライアントホストのいずれかになりますが、3 番目のホストにすることもできます。
アプリケーションサーバ	メールサーバまたは Web サーバなど、アプリケーションクライアントと通信するサーバプログラム
アプリケーションサーバホスト	サーバアプリケーションを実行しているコンピュータ。Secure Shell サーバホストまたは Secure Shell クライアントホストのいずれか、または 3 番目のホストにすることもできます。

ローカルポート転送

ローカルポート転送を使用して、Secure Shell クライアントと同じコンピュータ上で実行されているアプリケーションクライアントからデータを安全に転送できます。ローカルポート転送を構成する時は、データの転送に使用する任意のローカルポート、およびデータを受信する着信先ホストとポートを指定します。ローカルポート転送は次のように動作します。

1. Secure Shell 接続が確立されると、Secure Shell クライアントは、指定のローカルポートを使用して、ローカルコンピュータ (Secure Shell クライアントを実行しているコンピュータ) 上のリスニングソケット 『192ページ』を開きます。 ほとんどの場合、Secure Shell クライアントホストを実行しているアプリケーションのみがこのソケットを使用できます。

   ゲートウェイポート設定は、ローカルに転送されるポートをリモートアプリケーションが使用できるかどうかを制御します。既定ではこの設定は無効で、クライアントは、ローカルポート転送のためにソケットを開いた時に、ループバックアドレス (「localhost」または 127.0.0.1) を使用します。これにより、他のコンピュータで実行中のアプリケーションは、転送されるポートに接続できなくなります。ゲートウェイポートを有効にすると、リモートアプリケーションクライアントは、Secure Shell クライアントの Ethernet アドレス (IP アドレス、URL、DNS 名など) を使用してソケットを開くことができます。例えば、acme.com で実行中の Secure Shell クライアントがポート 8088 を転送するよう構成されているとします。ゲートウェイポートが無効な場合、転送されるソケットは localhost: 8088 です。ゲートウェイポートが有効な場合、転送されるソケットは acme.com: 8088 です。

   注意: ゲートウェイポートを有効化すると、使用しているクライアントホスト、ネットワーク、接続のセキュリティの低下を招きます。この理由は、リモートアプリケーションが、認証なしで、システム上の転送されたポートを使用することが可能になるからです。

2. アプリケーションクライアントは、(アプリケーションサーバホストおよびポートに直接接続するのでなく) 転送ポートに接続するように構成されます。 そのクライアントが接続を確立すると、すべてのデータがリスニングポートに送信され、Secure Shell クライアントにリダイレクトされます。
3. Secure Shell クライアントはデータを暗号化し、Secure Shell チャネルを通じて Secure Shell サーバに安全にデータを送信します。

4. Secure Shell サーバはデータを受信して解読し、アプリケーションサーバによって使用される送信先ホストおよびポートにリダイレクトします。

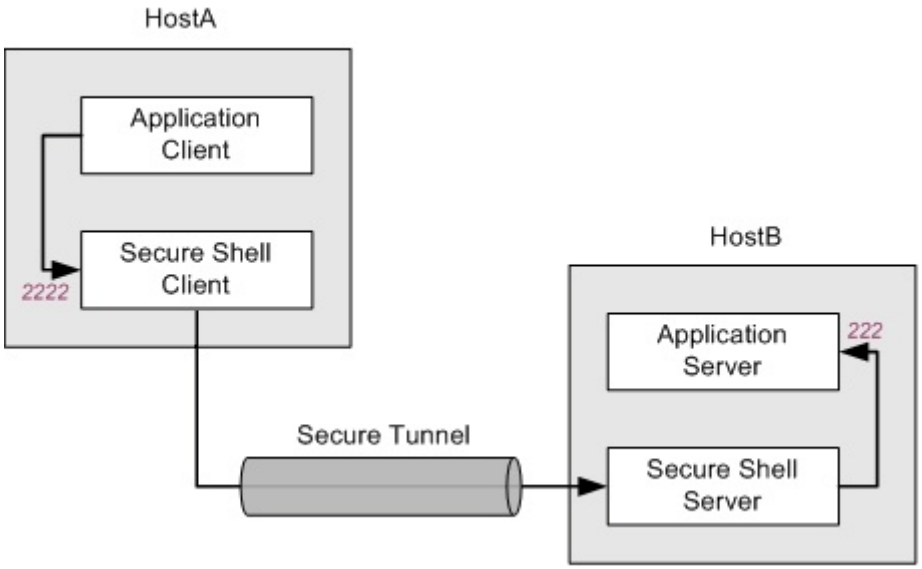
   注意: 最終宛先ホストおよびポートが Secure Shell サーバホスト上にない場合、Secure Shell ホストとアプリケーションサーバホスト間でデータは平文で送信されます。

5. アプリケーションサーバからの戻りデータは Secure Shell サーバに送られ、Secure Shell サーバは戻りデータを暗号化し、SSH トンネルを通じて Secure Shell クライアントに安全に送信します。 Secure Shell クライアントはデータを解読し、元のアプリケーションクライアントにデータをリダイレクトします。

ローカルポート転送の一般的なコマンドライン構文は以下のとおりです。

```
ssh -L listening_port:app_host:hostport user@sshserver
```

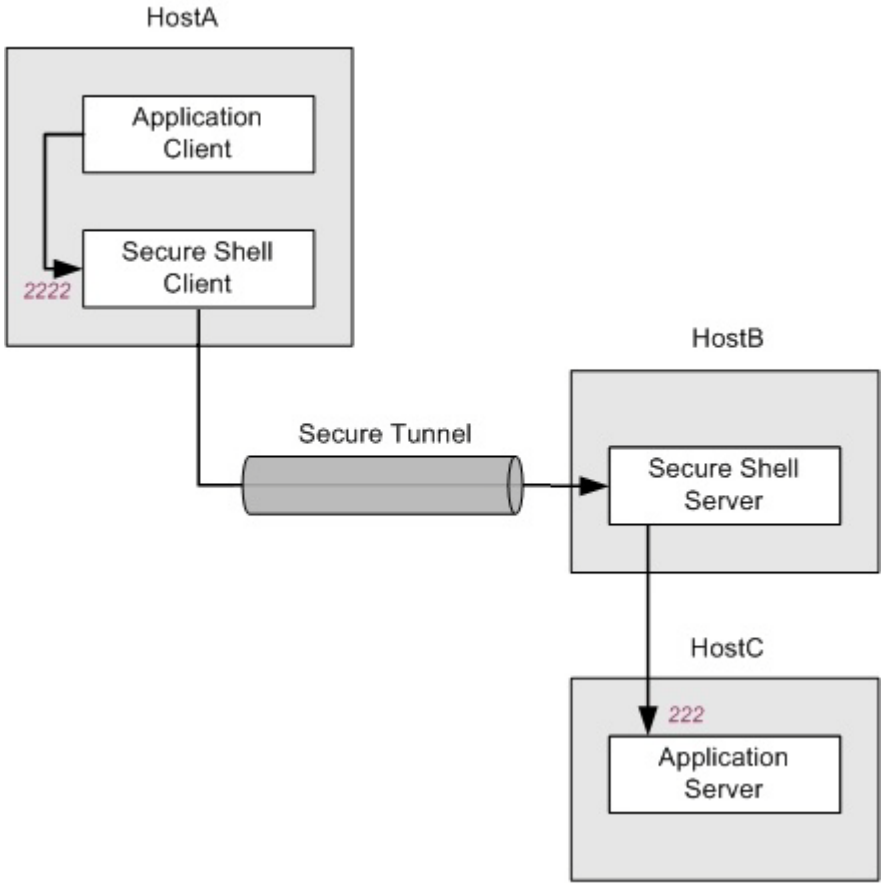
以下に示す図は、ローカルポート転送の 2 とおりの使用法を示しています。



上記の構成では、アプリケーションクライアントと Secure Shell クライアントの両方がホスト A で実行されます。Secure Shell サーバとアプリケーションサーバの両方はホスト B で実行されます。ホスト A のポート 2222 へ送信されたすべてのデータはホスト B のポート 222 へ転送されます。この配置では、転送中のすべてのデータが安全に暗号化されます。これは、以下のコマンド (localhost はホスト B のループバックアドレスを識別します) によって構成します。

```
ssh -L 2222:localhost:222 user@HostB
```

以下の図は、3 番目のホストへのローカルポート転送を示しています。この構成では、アプリケーションサーバが、Secure Shell サーバとは異なるホストで実行されます。ホスト A のポート 2222 へ送信されたすべてのデータはホスト C のポート 222 へ転送されます。



これは、以下のコマンドによって構成します。

```
ssh -L 2222:HostC:222 user@HostB
```

注意: ホスト B とホスト C 間で送信されるデータは暗号化されません。

リモートポート転送

リモートポート転送を使用して、Secure Shell サーバホストで実行されているアプリケーションクライアントからデータを安全に転送できます。リモートポート転送を構成する時は、データの転送に使用する任意のリモートポート、およびデータを受信する着信先ホストとポートを指定します。リモートポート転送は次のように動作します。

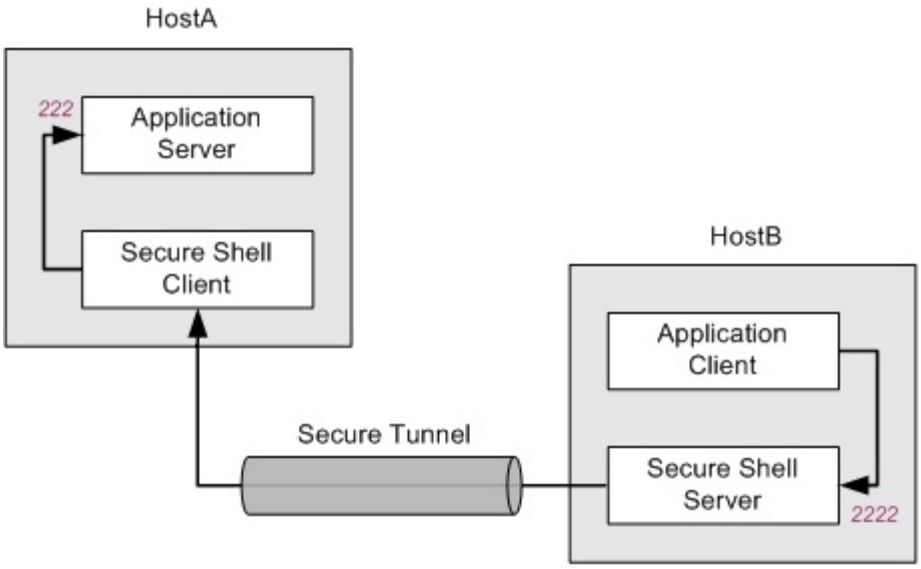
1. Secure Shell 接続が確立されると、Secure Shell サーバは、指定したリスニングポートを使用して、Secure Shell サーバホスト上のリスニングソケット 『192ページ』を開きます。
2. Secure Shell サーバホストで動作するクライアントアプリケーションは、(アプリケーションサーバホストおよびポートに直接接続する代わりに) リスニングポートに接続するように構成されます。そのクライアントが接続を確立すると、すべてのデータがリスニングポートに送信され、Secure Shell サーバにリダイレクトされます。
3. Secure Shell サーバはデータを暗号化し、SSH トンネルを通じて安全に Secure Shell クライアントにデータを送信します。

4. Secure Shell クライアントは、データを受信し、解読し、サーバアプリケーションが使用する (Secure Shell クライアントホスト上の) 宛先ホストおよびポートにリダイレクトします。
5. サーバアプリケーションからの戻りデータは Secure Shell クライアントに送られ、Secure Shell クライアントは戻りデータを暗号化し、SSH トンネルを通じて Secure Shell サーバに安全に送信します。Secure Shell サーバはデータを解読し、元のクライアントアプリケーションにデータをリダイレクトします。

リモートポート転送の場合の一般的なコマンドライン構文は次のとおりです。

```
ssh -R listening_port:app_host:hostport user@sshserver
```

次の図に、リモートポート転送構成の例を示します。



HostA では、アプリケーションサーバと Secure Shell クライアントが動作しています。HostB では、Secure Shell サーバとアプリケーションクライアントが動作しています。HostB のポート 2222 に送信されるすべてのデータが、HostA のポート 222 に転送されます。この配置では、通過中のすべてのデータが安全に暗号化されます。これを構成するコマンドは次のとおりです。

```
ssh -R 2222:localhost:222 user@HostB
```

ポート転送の構成

ポート転送トンネルを確立するには、**ssh** コマンドラインか、またはクライアント構成ファイル (`/etc/ssh2/ssh2_config`) のいずれかを使用します。

ローカルポート転送を構成および使用するには

1 転送に使用するローカルポートを選択します(この手順では例として 2110 を使用します)。

注意: これは使用可能ないずれのポートでもかまいませんが、1024 未満のポート値は使用しないでください。これらのポートは、慣例によりサービス用に予約されており、使用不可になっています。

2 アプリケーションクライアント (電子メールクライアントなど) を、リモートアプリケーションサーバのソケットではなくローカルホスト上の転送されたポートに接続するように構成します。この例では、以下のようになります。

転送されたローカルポート	リモートアプリケーションサーバのソケット
localhost: 2110	mailserver.com: 110

3 Secure Shell クライアントに接続します。

ローカルポート転送を使用して、転送されたローカルポートからリモートアプリケーションサーバへデータを送信します。一般的なコマンドライン構文は以下のようになります。

```
ssh -L listening_port:app_host:hostport user@sshserver
```

この例では、メールサーバが、Secure Shell サーバと同じホスト上で実行されます。この場合、アプリケーションホストは mailserver.com 上の「localhost」です。コマンドライン構成は以下のようになります。

```
ssh -L 2110:localhost:110 joe@mailserver.com
```

4 アプリケーションクライアントを通常どおり使用します。

データは、クライアントホスト上のリスニングポート (`localhost:2110`) から、保護されたチャネルを介して mailserver.com 上のリモートアプリケーションサーバのリスニングソケット (`localhost:110`) へ安全に転送されます。

3 番目のホストへの転送

前述の例では、アプリケーションサーバと Secure Shell サーバは同一のホスト上で実行されます。転送されるデータは、転送の全期間にわたって暗号化されます。アプリケーションサーバが、異なるホストで実行されている場合にもポート転送を使用することができます。たとえば、以下のようになります。

```
ssh -L 2110:mailserver.com:110 user@sshserver.com
```

この場合、データは、保護されたトンネルを介して sshserver.com へ転送されます。次に、データは、暗号化されずに mailserver.com 上のポート 2110 へ転送されます。

FTP 転送

Secure Shell トンネルを介して FTP 通信を転送するよう Reflection for Secure IT を構成できます。 FTP 転送は、アクティブモードとパッシブモード両方の転送に対応しています。

FTP 転送を使用すると、以下のようなメリットがあります。

- FTP アプリケーションを継続して使用できます。 すべての通信 (FTP コマンドチャネルやすべてのデータチャネルを含む) は Secure Shell クライアントと Secure Shell サーバ間で安全に暗号化されます。
- Secure Shell サーバと FTP サーバが同じコンピュータ上で実行されている場合は、Secure Shell ポート (22) のみをファイアウォール内で開く必要があります。トンネリングが設定されていない場合は、FTP 通信で、パッシブモードの転送用に FTP ポート (21) と広範囲の非特権ポートを開くことが必要とされます。
- アクティブモードの転送では、FTP クライアントコンピュータはファイアウォール内で開かれているポートを必要としません。

使用しているハードウェアリソースによっては、Secure Shell チャネルを使用して FTP 接続を転送すると、通常の FTP 接続と比べて、転送速度に変動が生じることがあります。 ネットワークの速度が CPU の速度よりも速い場合は、FTP トンネリングを設定すると、暗号化処理のため転送速度が低下することがあり、 ネットワークの速度が CPU の速度よりも遅い場合は、Secure Shell 圧縮を使用すると、転送速度が向上することがあります。

ローカル FTP 転送

ローカル FTP クライアントで使用されているポートからリモートの FTP サーバへ FTP 通信を転送するには、ローカルリスニングポートの前に接頭語「`ftp/`」を付けます。

以下の例では、(Secure Shell クライアントと同じコンピュータ上の) FTP クライアントから送信された FTP 通信は、myhost.com 上で動作している FTP サーバに転送されます。この構成では、FTP クライアントを `localhost:2121` に接続するよう構成します。

```
ssh -L ftp/2121:myhost.com:21 user@myhost.com
```

または、

```
LocalForward=ftp/2121:myhost.com:21
```

注意: FTP クライアントは、Reflection for Secure IT クライアントと同じサーバ上になければなりません。 Reflection for Secure IT サーバとは異なるホスト上の FTP サーバへのローカル FTP 転送を構成することはできますが、この場合、Reflection for Secure IT サーバから FTP サーバへの転送中はデータが暗号化されません。

リモート FTP 転送

リモート FTP クライアントで使用されているポートからローカルの FTP サーバへ FTP 通信を転送するには、リモートリスニングポートの前に接頭語「`ftp/`」を付けます。

以下の例では、(Secure Shell サーバと同じコンピュータ上の) FTP クライアントから送信された FTP 通信は、(Secure Shell クライアントと同じコンピュータ上の) FTP サーバに転送されます。この構成では、FTP クライアントをポート 3333 に接続するよう構成します。

```
ssh -R ftp/3333:localhost:21 user@myhost.com
```

または、

```
RemoteForward=ftp/3333:localhost:21
```

注意: FTP サーバは、Reflection for Secure IT クライアントと同じホスト上になければならず、FTP クライアントは、Reflection for Secure IT サーバと同じホスト上になければなりません。

X プロトコル転送

X ウィンドウシステムは、UNIX システム上のグラフィカルな表示に対応しています。X プロトコル転送を使用すると、X クライアントとリモートの X サーバ間の通信を保護することができます。X 転送は既定で有効になっています。X 転送は、以下のように機能します。

1. X 転送が有効になっている場合、Secure Shell クライアントは、サーバへの接続時に X 転送を要求します。
2. X 転送にサーバが対応している場合、サーバは、サーバ自体をサーバホスト上のプロキシ X サーバとして設定し、プロキシ X ディスプレイを示すようにクライアントシェルの DISPLAY 環境変数を設定します。

3. サーバホストで X クライアントプログラムを実行すると、当該プログラムはプロキシディスプレイに接続します。
4. Secure Shell クライアントはプロキシ X クライアントとして動作し、クライアントホスト上の X サーバに接続します。
5. すべての X プロトコル情報が、Secure Shell チャネルを介して送信されます。

X11 設定の動作

クライアント設定 **ForwardX11** により、X11 の転送が有効化または無効化されます(既定は「yes」です)。クライアント設定 **TrustX11Applications** では、X サーバが転送された X11 クライアントアプリケーションを信頼できるものとして扱うかどうかが指定されます(既定は「no」です)。

状況によっては、これらの設定の構成が X クライアントアプリケーションの起動速度に影響することがあります。この状況は、2 つ以上のシステムが同時に動作する場合に発生します。たとえば、以下のようになります。

> System1 で X サーバと Secure Shell クライアントが動作しています。
> System2 で X クライアントアプリケーション、Secure Shell クライアント、Secure Shell サーバが動作しています。
> System3 で X クライアントアプリケーションと Secure Shell サーバが動作しています。

X11Forwarding=yes (既定)、TrustX11Applications=no (既定) という状態で、System1 から System2 への **ssh** 接続を行う場合、X アプリケーションの起動で遅延は発生しません。

X11Forwarding=yes (既定)、TrustX11Applications=no (既定) という状態でその後に新しいシェルから System3 への **ssh** 接続を行うと、ユーザ認証後に長い遅延 (6 秒程度) が発生し、その間は System3 から起動された X アプリケーションが System1 で動作する X サーバに表示されません。xauth アプリケーションにより、X サーバとの通信と新しい cookie の登録が行われるため、これによってさらに遅延が長くなります。この遅延を防止し、System3 から X アプリケーションを実行するため、2 つめの接続に対して TrustX11Applications=yes と設定してください。

注意: 2 つめの接続に対して TrustX11Applications=yes という設定を行っても、System1 で動作する X サーバの安全上のリスクは拡大しません。xauth アプリケーションによる登録が、最初の (System1 からの) X11 転送によって System2 上に作成された既存の cookie (TrustX11Applications=no) に対して行われるためです。

ポート転送の設定

以下のキーワードまたはコマンドラインオプションを使用して、ポート転送を構成します。

コマンドラインオプション

ssh コマンドラインで以下のオプションを使用できます。

オプション	説明
-L *listening_port:host:hostport*	Secure Shell クライアントホスト上の指定のポート (*listening_port*) をリッスンし、データを着信先の *host* と *hostport* へ転送します。
-R *listening_port:host:hostport*	Secure Shell サーバホスト上の指定のポート (*listening_port*) をリッスンし、データを着信先の *host* と *hostport* へ転送します。
-X	X11 接続の転送を有効にし、X11 クライアントを信頼されないものとして扱います。ゲストのリモート X11 クライアントは、信頼される X11 クライアントに属するデータを不正に変更できません。
-x	X11 接続の転送を無効にします。
-Y	X11 接続の転送を有効にし、X11 クライアントを信頼関係があるクライアントとして扱います。

クライアント構成キーワード

次の設定をクライアント構成ファイルで設定できます。(グローバルファイルは `/etc/ssh2/ssh2_config` です。ユーザ固有のファイルは `~/.ssh2/ssh2_config` です。)

キーワード	説明
ClearAllForwardings	ローカル転送、リモート転送、または動的転送されたポートのうち、既に処理されたすべてのポートを構成ファイルまたはコマンドラインからクリアします。 scp と **sftp** は、この設定の値に関係なく、転送されたすべてのポートを自動的にクリアします。既定値は no です。
ForwardX11	**-X** に相当します。
GatewayPorts	Secure Shell クライアントホスト上の転送されたポートをリモートアプリケーションで使用可能にするかどうかを制御します。既定値は no であり、ほかのコンピュータ上で実行されているアプリケーションが、転送されたポートに接続できないようにします。
LocalForward *listening_port*:*host*:*hostport*	**-L** に相当します。
RemoteForward *listening_port:host:hostport*	**-R** に相当します。
TrustX11Applications	X サーバが、転送された X11 クライアントアプリケーションを信頼されるものとして扱うかどうかを指定します。既定値は no です。
XauthPath	**xauth** プログラムのフルパスを指定します。既定は `/usr/bin/xauth` です。

サーバ構成キーワード

サーバ構成ファイル (`/etc/ssh2/sshd2_config`) で以下の設定を構成できます。

オプション	説明
AllowTCPForwarding	すべてのポート転送を有効/無効にします。 既定値は yes です。
AllowX11Forwarding	X11 転送を許可するかどうかを指定します。 既定値は yes です。
AllowTCPForwardingForGroups **DenyTCPForwardingForGroups**	指定されたグループ用のポート転送を許可または拒否します。正規表現に対応しています。
AllowTCPForwardingForUsers **DenyTCPForwardingForUsers**	指定されたユーザのみに使用されるポート転送を許可または拒否します。正規表現に対応しています。
ForwardACL	ポート転送に対する詳細な制御を可能にします。詳細については、「サーバ構成キーワード 『118ページ 』」を参照してください。
GatewayPorts	リモートホストがクライアントに転送されるポートに接続できるかどうかを指定します。既定値は no です。
X11UseLocalHost	サーバが X11 転送をループバックアドレスにバインドする必要があるかどうかを指定します。既定値は yes です。

第 9 章

アクセスおよび権限の制御

この章の内容

アクセス制御の設定

以下の表に、サーバへのクライアントアクセスを制御するためのサーバの設定の概要を示します。

既定では、サーバホストのアカウントを持つすべてのクライアントユーザは、サーバに接続し、端末セッションを開始し、クライアントコンピュータによりユーザアカウントに許可されているすべてのローカルファイルとディレクトリにアクセスすることができます。クライアントユーザ、グループ、コンピュータがアクセスできるように、サーバ構成ファイル (`/etc/ssh2/sshd2_config`) をカスタマイズできます。

目的	使用コマンド
最大接続数を設定する	**MaxConnections**
TCP のシングル接続に対応する多重セッションの最大数を設定します。接続の再使用を無効にするには、このキーワードを 1 に設定します。	**MaxSessions**
指定されたセッションの種類のみへのアクセスを許可する	**SessionRestricted**
クライアントユーザからのアクセスを制御する	**AllowUsers** **DenyUsers** **UserSpecificConfig**
クライアントグループからのアクセスを制御する	**AllowGroups** **DenyGroups** **UserSpecificConfig**
クライアントホストからのアクセスを制御する	**AllowHosts** **DenyHosts** **HostSpecificConfig**
TCP ラッパを使用してアクセスを制御する	**LibWrap**

目的	使用コマンド
sftp および **scp** ユーザまたはグループを、限定されたディレクトリツリーに制限する	**ChrootSftpUsers** **ChrootSftpGroups**
sftp および **scp** ユーザのアップロード/ダウンロードアクセス権を制御する	**AllowSftpCommands**
ポート転送を制限する	**AllowTcpForwardingForGroups** **DenyTcpForwardingForGroups** **AllowTcpForwardingForUsers** **DenyTcpForwardingForUsers** **ForwardACL** **GatewayPorts** **AllowX11Forwarding** **X11UseLocalHost**
PAM 認証を構成する	**AccountManagement** **AuthKbdInt.Required** **PamServiceName** **UsePamSessions**

Allow および Deny キーワードの使用

ユーザ、グループ、クライアントホストコンピュータへのアクセスを制御するために以下のキーワードを使用できます。

AllowUsers、**DenyUsers**、**AllowGroups**、**DenyGroups**、**AllowHosts**、**DenyHosts**、 **AllowTcpForwardingForUsers**、**DenyTcpForwardingForUsers**、**AllowTcpForwardingForGroups**、**DenyTcpForwardingForGroups**、**ForwardACL**

これらキーワードのいずれかに対してユーザ、グループ、またはホストを指定するには、値のカンマ区切りリストを含む単一のキーワードインスタンスを使用するか、複数のキーワードインスタンスを含めます。後者の場合、最後に指定された値がすべてのインスタンス上に累積的に追加されます。

サーバは、以下のロジックを使用して、接続を許可するかどうかを決定します。

1. 所定のアクセスの種類 (ホスト、ユーザ、グループ、または TCP 転送) に対して「Deny」キーワードが構成されているかどうかを調べて確認し、クライアントがいずれかの拒否式に一致する場合はアクセスを拒否します。
2. 同じ種類に対して「Allow」キーワードが構成されているかどうかを調べて確認します。
   - 「Allow」キーワードが構成されていない場合、アクセスは許可されます。
   - *いずれかの*「Allow」キーワードが構成されている場合、サーバは、クライアントが許可式に一致する場合*のみ*、アクセスを許可します。

例

以下の例は、どのようにしてアクセスを許可、アクセスを拒否、または許可と拒否を組み合わせて使用できるかを示しています。

目的	例
名前が「abc」で始まるユーザに対してのみアクセスを許可します。	`AllowUsers= abc.*`
IP アドレスが 123.156.78 で始まるクライアントホストに対してアクセスを拒否し、それ以外のクライアント上のユーザに対してアクセスを許可します。	`DenyHosts=123\.156\.78\..*`
badpc 以外の acme.com ドメイン内のすべてのホストに対してアクセスを許可し、それ以外のドメインのクライアントに対してアクセスを拒否します。	`AllowHosts=.*\.acme\.com` `DenyHosts=badpc\.acme\.com`

mypc を含めて acme.com ドメイン内のすべてのホストに対してアクセスを拒否し、それ以外のドメインのクライアントに対してアクセスを許可します。	`DenyHosts=.*\.acme\.com`
	`AllowHosts=mypc\.acme\.com`
	`AllowHosts=.*`
	注意: 最後の行が存在しないと、いずれのクライアントへのアクセスも許可されません。この理由は、いずれかのクライアントが許可リストに追加されると、クライアントが許可式に一致する場合にかぎりクライアントへのアクセスが許可されるようになるためです。

注意: サブ構成ファイル 『30ページ 』を使用して、ユーザ固有設定やホスト固有設定を構成することもできます。

ユーザアクセスの構成

サーバへのアクセスを制御するにはサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を編集します。 キーワード **AllowUsers**、**DenyUsers**、**AllowTcpForwardingForUsers**、**DenyTcpForwardingForUsers**、**ForwardACL**、**ChrootSftpUsers**、**UserSpecificConfig** を使用してユーザアクセスを構成します。ユーザ名だけを指定することも、以下の構文を使用してグループやホスト情報を含めることもできます。

user[%*group*][@*host*]

ここで、*user* はユーザの正規表現 (数値の UID には対応していません)、*group* はグループの正規表現 (数値の GID には対応していません)、および *host* はホストの正規表現 (ドメイン名、IP アドレス、サブネットマスクのいずれか) になります。たとえば、以下の構文の例では、myhost.com の interns グループのすべてのメンバーへのアクセスを拒否します。

```
DenyUsers=.*%interns@myhost.com
```

グループアクセスの構成

サーバへのアクセスを制御するにはサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を編集します。 キーワード **AllowGroups**、**DenyGroups**、**AllowTcpForwardingForGroups**、**DenyTcpForwardingForGroups**、**ChrootSftpGroups** を使用してグループアクセスを構成します。これらのキーワードは任意の有効な正規表現に対応しています。数値の GID には対応していません。たとえば、以下のようになります。

```
DenyGroups=interns
```

クライアントホストアクセスの構成

サーバへのアクセスを制御するにはサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を編集します。 キーワード **AllowHosts**、**DenyHosts**、**HostSpecificConfig** を使用して、クライアントホストコンピュータの設定を構成します。ホストを指定するには、IP アドレスまたはドメイン名のいずれかを使用します。サーバは、最初に、クライアントの IP アドレスを使用して一致しようと試みます。これに失敗した場合、サーバはドメイン名を使用して一致しようと試みます。

注意: **ResolveClientHostname** 設定は、サーバが、ドメイン名へのクライアント IP アドレスの解決を試行するかどうかを制御します。既定値は「yes」です。クライアントの解決されたドメイン名は、常に完全修飾ドメイン名になります。 これは、ドメイン名を使用してホストを許可リストまたは拒否リストに追加する時は、ホストのドメイン名が適正に処理されるように完全修飾ドメイン名または正規表現のいずれかを使用する必要があることを意味します。 たとえば、クライアント「mypc」へのアクセスを拒否する場合、クライアント `mypc.myhost.com` は接続可能です。 当該クライアントに対するアクセスの拒否を確実なものにするには、明示的に「`mypc\.myhost\.com`」へのアクセスを拒否するか、「`mypc\..*`」などの正規表現を使用する必要があります。

さらに、IP アドレスと一致させるようにサーバを構成することもできます。特定の IP アドレスと一致させるには、後続の i を伴うバックスラッシュ (`\i`) を使用したホスト表現を開始します。たとえば、以下のようになります。

```
DenyHosts = \i123.45.78.9
```

CIDR (Classless Inter-Domain Routing) サブネットを使用して IP アドレスの範囲を一致させるには、後続の m を伴うバックスラッシュ (`\m`) を使用したホスト表現を開始します。たとえば、以下のようになります。

```
DenyHosts = \m123.123.0.0/16
```

注意: `\i` または `\m` のいずれかを使用する場合、正規表現は IP アドレス内では対応していません。

注意

- 許可リストまたは拒否リストにローカルホストを構成するには、すべての外部インタフェースの IP アドレスと、ローカルループバックアドレス (127.0.0.1 および 0: 0: 0: 0: 0: 0: 0: 1) を記述します。
- 許可リストまたは拒否リストにアドレスを構成するには、IPv4 と IPv6 の両方のアドレスを指定する必要があります。拒否リストによってアクセスがブロックされるように構成する場合は、このことは特に重要です。

第 10 章

監査

この章の内容

ファイル転送監査

ファイル転送動作の詳細レコードを保持するため、サーバがファイル転送ログを作成するよう構成できます。この記録は既定で有効になっています。

ファイル転送監査を有効にするには

1 テキストエディタでサーバ構成ファイル (`/etc/ssh2/sshd2_config`) を開きます。

2 次のように、**AuditLog** を「sftp」に設定します。

```
AuditLog = sftp
```

3 (オプション) **AuditLog.Directory** や **AuditLog.Sftp.WithHash** の値を編集します。指定可能な値は、次のとおりです。

```
AuditLog.Directory=/etc/ssh2/logs
```

```
AuditLog.Sftp.WithHash=yes
```

4 構成ファイルに変更内容を保存します。

監査ログの記録が有効になっていると、Reflection for Secure IT は、指定した監査ログディレクトリに新しいログを毎日作成します。監査ログでは、名前形式として `sshd2-audit-YYYYMMDD.log` が使用されます。ここで、YYYYMMDD は日付を示します。

ログファイルはカンマ区切り形式で作成されます。監査ログファイルの最初の行には、次のようにログの内容が表示されます。

「UserID」、「ClientIP」、「Action」、「ServerFilename」、「StartTime」、「EndTime」、
「ServerFileModificationTime」、「ServerFileSize」、「BytesTransferred」、
「Result」、「Reason」、「ServerFileHash」。

注意:

- サーバでは、SCP2 (SFTP サブシステムを使用するファイル転送実装) を使用する sftp 転送と scp 転送が記録されます。SCP1 (OpenSSH が使用する SCP プロトコルの初期実装) を使用する転送は記録されません。

- **SmartFileTransfer** が有効になっている（既定）場合は、クライアントファイルとサーバファイルが同一であると、サーバが監査レコードを作成しない場合があります。同一ファイルの転送で監査レコードが作成されるようにするには、**SmartFileTransfer** を `no` に設定します。
- `sftp` プロトコル接続について、ユーザが自分のホームディレクトリに限定されている場合（**ChrootSftpUsers** または **ChrootSftpGroups**を使用する）、監査ログディレクトリがユーザのホームディレクトリ（`AuditLog.Directory = /home/joe/logs` など）にないと転送が失敗します。

監査 (メッセージの記録)

次の監査サービスは常に有効です。

- ログイン履歴
- 現在ログインしているユーザ
- 失敗したログイン

ファイル転送動作のレコードを保持する追加監査も構成できます。この監査は、既定では有効になっていません。詳細については、「ファイル転送監査の構成 『89ページ 』」を参照してください。

以下の表に、監査の構成用のキーワードをまとめます。

キーワード	説明
AuditLog	監査ログを作成するかどうかを指定します。有効な値は「sftp」および「none」 です。 「sftp」を指定すると、ファイル転送活動の詳細レコードを含むカンマで区切られたログファイルが、**AuditLog.Directory** で指定された場所に作成されます。監査ログファイルの最初の行には、次のようにログの内容が表示されます。UserID,ClientIP,Action, ServerFilename,StartTime,EndTime,ServerFileModificationTime, ServerFileSize,BytesTransferred,Result,Reason,ServerFileHash 既定値は「none」です。
AuditLog.Directory	監査ログの出力場所です。日ごとの新しいログが sshd2-audit-YYYYMMDD.log の名前形式で作成されます。ここで、YYYYMMDD は日付です。AuditLog = sftp の場合、クライアントユーザによる最初のファイル転送時か、サーバの再起動時にこのファイルが作成されます。 既定値は次のとおりです (`/etc/ssh2/logs`)。
AuditLog.Sftp.WithHash	sftp ログエントリにファイルハッシュが含まれるかどうかを指定します。ハッシュ値を使用して、同一ファイルの転送を示す複数のレコードを特定することができます。変更されていないファイルが転送された場合、ログ中のハッシュ値は毎回同じになります。ファイルが変更されている場合は、ハッシュ値が異なります。 既定値は「yes」です。

キーワード	説明
SyslogFacility	**sshd** メッセージのファシリティコードです。**SyslogFacility** の値は、`syslog.conf` に指定した機能に対応していなければなりません。 既定値は「AUTH」です。
SftpSysLogfacility	**sftp-server** メッセージのファシリティコードです。値を構成しない場合 (既定)、**sftp-server** では **sshd** 用に構成された現在の機能が使用されます。sftp サーバの記録用に代替機能を指定するには、**SftpSysLogFacility** を使用してください。sftp メッセージを別の機能に送信することが、監査のために有益なことがよくあります。 **SftpSysLogFacility** の値は、`syslog.conf` に指定した機能に対応していなければなりません。
SftpLogCategory	**SftpSysLogFacility** で指定した機能に送信される **sftp** サーバメッセージの種類を指定します。 既定値は「loginlogout,directorylistings,downloads,modifications,uploads」です。これですべての種類の記録が構成されます。これらのオプションか、「all」または「none」を指定することができます。
LogLevel	**SysLogFacility** と **SftpSysLogFacility** の記録レベルです。構成ファイルが読み込まれた後、`syslog.conf` で定義されたルールに従ってメッセージが処理されます。 このレベルは、**sshd** と **sftp** の両方の記録に適用されます。

ログファイルの場所

ファイル転送監査ログ

ファイル転送監査を有効にしておくと、ログは既定で次の場所に作成されます (`/etc/ssh2/logs`)。**AuditLog.Directory** を使用して既定以外の場所を構成できます。

ログイン情報

ログイン情報の出力場所は、プラットフォームによって異なります。詳細については、以下の表を参照してください。

プラットフォーム	ログイン履歴	現在のログイン	失敗したログイン
HPUX (11.11)	`/var/adm/wtmp`	`/etc/utmp`	`/var/adm/btmp`
HPUX (11.23, 11.31)	`/var/adm/wtmps`	`/etc/utmpx`	`/var/adm/btmps`
AIX	`/var/adm/wtmp` `/etc/security/lastlog`	`/etc/utmp`	`/etc/security/failedlogin` `/etc/security/lastlog`

Solaris	/var/adm/wtmpx	/var/adm/utmpx	/var/adm/loginlog
RHEL	/var/log/lastlog /var/log/wtmp	/var/run/utmp	/var/log/btmp
SLES	/var/log/wtmp	/var/run/utmp	/var/log/btmp

注意:

- プラットフォームの中には、複数のファイルに書き込みを行うものもあります。
- 一部の Linux システムには btmp が存在していません。サーバは、このデータベースが存在している場合に当該データベースへの書き込みを行います。

sshd および **sftp-server** メッセージの出力は、Reflection for Secure IT 構成と **syslogd** 構成の両方の影響を受けます。例えば、/etc/syslog.conf 内の次の項目は、local6 という名前の機能を構成し、その機能から /var/adm/rsit_log に出力を送信します。

```
local6.info        /var/adm/rsit_log
```

注意: 上記の構文では、2 つの項目をタブで区切る必要があります。

local6 機能に **sshd** メッセージを送信するように Reflection for Secure IT を構成するには、サーバ構成ファイルに次の行を記述します (/etc/ssh2/sshd2_config)。

```
SysLogFacility local6
```

Solaris の監査への対応

Solaris の動作環境は、ファイルアクセス、プロセスオペレーション、ネットワークアクティビティなどのシステムイベントの監査に対応しています。監査を有効にすると、選択したイベントの監査証跡がログファイルの形式で作成され、システムの不正使用を検出するためにこれを監視することができます。Solaris の動作環境での監査は、Basic Security Module (BSM) によって行われます。BSM の構成の詳細は、Solaris のマニュアルを参照してください。

Secure Shell 接続用の監査レコードを生成するには

1 Secure Shell (SSH) 監査イベントが正しい監査クラスにマッピングされていることを確認します。/etc/security/audit_event ファイル内の以下の行は、すべての Secure Shell イベントがイベントのlogin/logout クラスに属すよう定義します。

```
6172:AUE_ssh:login - ssh:lo
```

注意: Solaris 8 を使用している場合、この項目は存在しませんが、手作業で追加する必要があります。

2 /etc/security/audit_control ファイルを編集します。このファイルでは、すべてのユーザ用にシステム全体の監査設定を定義できます。flags:セクションに login/logout イベントクラスを追加してください。

```
flags:lo
```

3 (オプション) 一部のユーザに特別な監査設定が必要な場合、または一部のユーザに対して監査を削除する場合は、/etc/security/audit_user ファイルを編集します。これらの項目は以下のような形式となります。

```
user:always_audit-flags:never_audit_flags
```

例えば、/etc/security/audit_user ファイル内の以下の項目では、ユーザ「joe」の監査が無効になります。

```
joe::lo
```

監査ログを表示するには

1 以下の場所で監査ログを探します。

```
/var/audit
```

2 バイナリ形式のファイルを読み込むには、**praudit** コマンドを使用します。

```
praudit audit_file
```

Secure Shell の login/logout イベント用の監査項目では、ログインまたはログインを試行したユーザ、それに使用されたホスト、接続が成功したかどうかが分かります。

例 1

ユーザ「joe」の項目、ホスト sphinx.company.com からログイン:

```
header,94,2,login - ssh,,Tue May 13 10:49:44 2010, + 863 msec
subject,joe,joe,other,joe,other,7763,7763,0 2805 sphinx.company.com
text,sshd login joe on /dev/pts/4
return,success,0
```

この場合、ユーザは正常にシステムにログインし、/dev/pts/4 で Secure Shell の端末セッションが行われました。

例 2

リモートコマンドや、**scp** または **sftp** によるファイル転送など、端末セッションが不要な Secure Shell ログインでは、端末または tty 番号の代わりに、サーバがユーザの代理で実行するコマンドが示されます。たとえば、以下のようになります。

```
header,116,2,login - ssh,,Tue May 13 10:49:58 2010, + 361 msec
subject,joe,joe,other,joe,other,7774,7774,0 2806 sphinx.company.com
text,sshd login joe on (no tty)
text,remote command:sftp
return,success,0
```

例 3

ログイン試行の失敗例:

```
header,81,2,AUE_ssh,,Tue May 13 11:22:51 2003, + 462 msec
subject,joe,joe,other,joe,other,8006,8006,0 0 sphinx.company.com
text,invalid password
return,failure:Interrupted system call,-1
```

第 11 章

デバッグのログ記録

この章の内容

Reflection for Secure IT ログは、問題解決に役立つ詳細情報を提供します。クライアントログに含まれる情報は、サーバログに含まれる情報と異なっており、多くの場合、両方のログを保持することが有用です。

クライアントのデバッグ

ssh、sftp、および scp コマンドラインでデバックを構成できます。さらに、これらすべてのセッションの種類に適用されるデバッグをクライアント構成ファイル (`/etc/ssh2/ssh2_config`) で構成することもできます。

コマンドラインオプション

クライアント側のデバッグを構成するには以下のコマンドラインオプションを使用します。

オプション	使用者	説明
-d *debug_level*	**ssh**、**ssh-agent**	デバッグレベルを設定します。値を上げると、表示される情報量が増えます。 1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。
-D *debug_level*	**scp**、**sftp**	**ssh -d** に相当します。 注意: **scp** と **sftp** は大文字の D を使用します。
-v	**ssh**、**scp**、**sftp**	デバッグレベルを冗長モードに設定します。これは、デバッグレベルを 2 に設定することと同じです。
-q	**ssh**	クワイエットモードを有効にします。このモードでは、バナーを含むすべての警告および診断メッセージが表示されません。

構成ファイルキーワード

次の設定をクライアント構成ファイルで設定できます。(グローバルファイルは `/etc/ssh2/ssh2_config` です。ユーザ固有のファイルは `~/.ssh2/ssh2_config` です。)

キーワード	説明
LogLevel	**SyslogFacility** によって指定された機能に送信されるメッセージの冗長レベルを設定します。
QuietMode	警告および診断メッセージ (バナーを含む) を抑制します。
VerboseMode	デバッグレベルを「冗長」モードに設定します。**-v** および **LogLevel** = verbose に相当します。
SyslogFacility	**ssh**、**sftp**、および **scp** メッセージの記録に使用する機能コードを指定します。既定値は「USER」です。クライアントの監査を無効にするには、値の設定を「none」にします。

サーバのデバッグ

サーバイベントメッセージは、さまざまなソースから生成され、それぞれの構成オプションによって制御されます。以下の表では、記録に影響する **sshd** コマンドラインオプションおよびサーバ構成キーワードの要約、および出力先の場所を示します。

目的	使用コマンド	出力場所	注意
単一のクライアント接続をデバッグする	**-d** *debug_level*	`stderr`	1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。 このオプションを使用した場合、**sshd** は、最初のクライアント接続が閉じた後に終了します。 このオプションは **LogLevel** の設定に依存しません。
継続的なデバッグを有効にする	**-D** *debug_level*	`/etc/ssh2/logs`	以下の名前の形式でデバッグファイルが作成されます。 `debugYYMMDD_HHMMSS_uniqueID` このオプションは **LogLevel** の設定に依存しません。
デバッグメッセージを抑制する	**-q** **QuietMode**	N/A — syslog 出力のみに影響します。	このオプションは **LogLevel** に優先します。
サーバ起動メッセージを表示する	**LogLevel**	`stderr`	`stderr` への出力には、`sshd2_config` の解析中に検出されたエラーおよび警告が含まれます。

第 12 章

問題解決

この章の内容

公開鍵認証の問題解決

問題: 公開鍵認証が構成されているが、公開鍵認証を使用してクライアントユーザの接続ができない。

クライアントの構成の確認

1 クライアントに秘密鍵/公開鍵のペアがあることを確認し、秘密鍵の名前と場所をメモしてください。

2 クライアントの構成ファイルを開いてください(ユーザ固有のファイルがある場合、グローバルファイルとユーザファイルの両方を確認してください)。

- **AllowedAuthentications** に「publickey」が含まれていることを確認してください。
- **IdentificationFile** 設定を確認してください。ファイルの名前と場所をメモしてください(既定値は `~/.ssh2/identification`)。

3 ID ファイルを開く

- このファイルに、クライアントの秘密鍵を識別する行が記述されていることを確認してください。例えば、次のように入力します。
  `IdKey /home/joe/mykey`
- 鍵の名前が鍵のペアの秘密鍵に完全に一致していることを確認してください。(例えば、秘密鍵にファイル拡張子がある場合、この拡張子も含めて照合する必要があります。)
- パスが指定されていない場合、鍵が Secure Shell のユーザディレクトリにあることを確認してください。 (`~/.ssh2/`)

4 ファイルとディレクトリの権限を確認してください。(クライアントで **StrictModes** が有効になっている場合、2 番目と 3 番目の項目が必要です。これが既定です。)

- 秘密鍵の読み取りは所有者のみが可能か (600)?
- ID ファイルは、ユーザにのみ書き込みアクセスを許可するよう構成されているか (600 または 644)?
- ユーザディレクトリとすべての親ディレクトリは、ユーザにのみ書き込みアクセスを許可するよう構成されているか (755 以下)?

サーバ構成の確認

1 サーバのユーザ固有の構成ディレクトリにユーザの公開鍵のコピーがあるか確認してください。既定の場所は次のとおりです (`~/.ssh2`)。

2 サーバの構成ファイルを開いてください。

- **AllowedAuthentications** に「publickey」が含まれていることを確認してください。
- **AuthorizationFile** 設定を確認してください。ファイルの名前と場所をメモしてください(既定値は `~/.ssh2/authorization`。)

3 認証ファイルを開いてください。

- このファイルに、クライアントの公開鍵のサーバ側のコピーを識別する行が記述されていることを確認してください。例えば、以下のようになります。
  `Key /home/joe/mykey.pub`
- 鍵の名前が、ファイル拡張子も含めて公開鍵に完全に一致していることを確認してください。
- パスが指定されていない場合、鍵が Secure Shell のユーザディレクトリにあることを確認してください(既定値は `~/.ssh2/`)。これは、サーバで **UserConfigDirectory** キーワードを使用して構成することができます)。

4 ファイルとディレクトリの権限を確認してください。(サーバで **StrictModes** が有効になっている場合、2 番目の項目が必要です。これが既定です)。

- 認証ファイルは、ユーザにのみ書き込みアクセスを許可するよう構成されているか (600 または 644)?
- ユーザディレクトリとすべての親ディレクトリは、ユーザにのみ書き込みアクセスを許可するよう構成されているか (755 以下)?

ファイルの転送速度低下の問題解決

ファイルの転送速度は、クライアントおよびサーバシステムの CPU の能力、転送用に利用可能な帯域幅 『193ページ』、ネットワークの待ち時間 『193ページ』など、さまざまな要因の影響を受けます。多くの場合、Reflection for Secure IT の既定の設定を使用することで、最良のパフォーマンスが得られます。場合によっては、以下の設定が転送速度に影響することもあります。

圧縮設定の確認

クライアントとサーバの両方で **Compression** キーワードを使用して、圧縮を構成することができます。圧縮値は 0 ～ 9 を指定できます。サーバの既定値は 6 です。クライアントの既定値は 0 です。値を上げると、圧縮量が増えます。大きな値を使用すると、ネットワークの帯域幅が小さくなりますが、CPU サイクルは大きくなります。

- **Compression** の値を小さくするか、または設定を 0 にすると、ファイルがすでに圧縮されている場合、ネットワークの帯域幅が大きい場合、あるいはコンピュータの CPU の能力が限られている場合に、パフォーマンスが向上することがあります。
- **Compression** の値を大きくすると、ファイルが圧縮されていない場合、ネットワークの帯域幅が小さい場合、あるいは CPU の能力が制約要因とならない場合に、パフォーマンスが向上することがあります。

スマートコピーおよびチェックポイント再開の設定の確認

スマートコピーおよびチェックポイント再開 『69ページ』は、無駄なデータ転送の反復に費やす時間を最小限に抑えるのに役立ちます。これらの機能では、クライアントとサーバの間で送信される一連のハッシュに基づいて、ファイルの一部または全部が同じであるかどうかを判断しています。同じ内容は転送されません。この機能は既定で有効になっています。混雑したネットワークでファイルを転送する場合、この機能を利用するメリットよりも、ネットワーク上でハッシュ値を送信するのに要する時間によるデメリットのほうが上回ることがあります。

- 待ち時間の長いネットワークで大容量のファイルを転送する場合、スマートコピーおよびチェックポイント再開機能を無効にすることにより、パフォーマンスを向上できる場合があります。これらの機能をクライアントから無効にするには、**SmartFileCopy** および **CheckpointResume** の設定を「no」にします。これらの機能をサーバから無効にするには、**SmartFileTransfer** の設定を「no」にします。

High Performance Network (HPN) 設定の確認

Reflection for Secure IT は、ファイル転送のパフォーマンスを最大にする HPN 機能に対応しています。この機能は既定で有効になっています。

- 最良のパフォーマンスを得るため、クライアントとサーバの両方で **HPNDisabled** の設定が「no」 (既定) になっていることを確認してください。

注意: クライアントとサーバで HPN を有効にすることによる転送速度の向上は、待ち時間
『193ページ 』が長く、帯域幅 『193ページ 』が広いネットワークで効果が最も高くなります。

SELinux を実行するシステムのトラブルシューティング

問題: Security-Enhanced Linux (SELinux) を実行している Linux システムで、Reflection for Secure IT をインストールまたは実行しようとすると、次のようなエラーメッセージが表示される。

```
error while loading shared libraries:/usr/lib/libssccm.so.2.0.40:cannot restore segment prot after reloc:Permission denied
```

```
error while loading shared libraries:libssccm.so.2.0.40:cannot enable executable stack as shared object requires:Permission denied
```

これらのエラーは、SELinux が有効になっており、Targeted ではなく Enforcing に設定されていると発生します。Reflection for Secure IT 暗号化モジュールで、テキストの位置を変更する必要があります。次の手順で、この要件に対応できます。

SELinux が有効なシステムで Reflection for Secure IT をインストールおよび実行するには

1 Reflection for Secure IT のインストール後に上記のようなエラーメッセージが表示される場合は、次のコマンドを実行します。指定するライブラリ名は、エラーメッセージ中のライブラリ名と一致する必要があります。

```
chcon -t textrel_shlib_t /usr/lib/libssccm.so.2.0.40
```

2 ホスト鍵を作成します。 『38ページ 』

3 サーバを起動します。 『21ページ 』

注意: Reflection for Secure IT のアップグレードのたびに、手順 1 と 3 を繰り返す必要があります。

付 録

付録

この章の内容

付 録 A

クライアントで使用されるファイル

`$HOME/.ssh2/ssh2_config`

ユーザ固有の構成ファイル。形式は、システム全体にわたる構成ファイルと同じです。 推奨されるパーミッション = 644。

`/etc/ssh2/ssh2_config`

システム全体にわたる構成ファイル。このファイルは、Reflection for Secure IT のインストール時にインストールされます。インストールされたファイルには、既定値がコメントアウト行として表示されます。システム全体にわたる設定を変更するには、このファイルを編集します。キーワードおよび対応している値については、ssh2_config(5) を参照してください。 推奨されるパーミッション = 644。

`$HOME/.ssh2/hostkeys/key_*.pub`

このディレクトリには、現在のユーザによって信頼されているホストの公開鍵が含まれます。既定では、不明なホストのプロンプトに対してユーザが「yes」と応答すると、鍵はこの場所に自動的に追加されます。(この動作は、次を使用して変更できます。 **StrictHostKeyChecking** キーワード (構成ファイル内)。)バージョン 7.0 から、ホスト鍵は次のファイル名形式を使用します。

key_*port_host*,*IP*.pub

ここで、*port* は SSH 接続に使用されるポート、*host* はホスト名、IP はホストの IP アドレスです。(以前のバージョンは key_*port_host*.pub を使用していました。この形式にも引き続き対応しています。)

注意: 既定では、このディレクトリに追加された鍵にはグループおよびグループ外の読み取りアクセス (644) があります。セキュリティを向上させるには、これらのファイルに権限を設定し、所有者 (600) のみが読み取りできるようにします。

`/etc/ssh2/hostkeys/key_*.pub`

システム全体で既知のホスト。この一覧に鍵があるホストは、コンピュータのすべてのユーザから信頼されます。この場所に鍵が自動的にインストールされることはありません。システム全体にわたって信頼されるホストを追加するには、このディレクトリを作成し、ホストの公開鍵のコピーを配置します。上記で説明されているファイル名形式を使用してください (`$HOME/.ssh2/hostkeys/key_*.pub`)。

`$HOME/.ssh2/identification`

公開鍵または証明書によってユーザ認証を行う場合、ID ファイルが必要です (これは、既定のファイル名と場所です。ID ファイルの名前や場所を再定義できます。**ssh** コマンドラインで次を使用します (**-i** または **IdentificationFile** キーワード))。ID ファイルには、クライアントユーザが保有する 1 つまたは複数の秘密鍵の一覧が含まれます。クライアントは、一覧にある任意の鍵をユーザ認証に使用できます。複数の鍵が一覧にある場合、クライアントはまず一覧の先頭の鍵を試し、続いて順序どおりに他の鍵を試します。パス情報がない場合、クライアントは `$HOME/.ssh2/` 内のキーのリストを検索します。このファイルは、ユーザのみの書き込みアクセスが可能 (権限が 600 または 644) なように設定されていなければなりません。

標準の鍵の場合、リストに鍵を追加するための構文は以下のとおりです。

`IdKey <keyname>`

例えば、次のように入力します。

`IdKey id_dsa_2048_a`

n X.509 証明書に関連する鍵の場合、構文は以下のとおりです。

`CertKey <keyname>`

関連する証明書は、指定した鍵と同じディレクトリ内にあり、 .crt ファイル拡張子が付いた同じベース名を使用する必要があります。

注意: 公開鍵の認証では、サーバも構成する必要があります。証明書の認証では、Reflection PKI Services Manager をインストールおよび構成して、サーバを構成する必要があります。

注意: ここで **ChrootSftpUsers** または **ChrootSftpGroups** が有効になっている場合、接続されているユーザに対して、ホームディレクトリに追加されたサブディレクトリ (すべてのプラットフォーム上のetc および AIX 上の dev) も表示されます。これらのディレクトリは、移動したり削除したりすることができません。その etc ディレクトリには、2 つの必須ファイルが含まれます。その rsit.conf ファイルは、Reflection for Secure IT で必要となるファイルのインストール場所を特定するためのファイルです。その localtime ファイルは、記録などのプロセスが現在の時刻を取得するために必要となるファイルです。システムの localtime ファイルは、chroot したユーザがアクセスできない場所にあります。AIX 上で作業しているユーザは /dev/nullも必要です。これは、syslog に正しく記録するために必要です。

付 録 B

サーバで使用されるファイル

サーバでは、すべての接続にシステム全体のファイル (場所は `/etc/ssh2`) を使用します。ユーザ固有のディレクトリ内のファイル (既定の場所は `~/.ssh2`) は、個々のクライアントユーザからの接続に使用されます。

システム全体のサーバファイル

`/etc/ssh2/sshd2_config`

グローバルサーバ構成ファイル。このファイルは、グループなどによって書き込み可能であってはなりません。ファイル形式および対応している設定については、sshd2_config(5) を参照してください。推奨されるパーミッション = 644。

`/etc/ssh2/hostkey`

クライアントにサーバを認識させるのに使用する公開/秘密鍵ペアの既定の秘密鍵。このファイルはルートのみから読み書きできる必要があります。このファイルのアクセスは、ユーザのみの読み書きアクセスに制限する必要があります。権限の制限が十分でない場合、公開鍵の認証は失敗します。推奨されるパーミッション = 600。

`/etc/ssh2/hostkey.pub`

クライアントにサーバを認証させるのに使用する公開/秘密鍵ペアの既定の公開鍵。推奨されるパーミッション = 644。

`/etc/ssh2/subconfig`

オプションのユーザ固有のサブ構成ファイルおよびホスト固有のサブ構成ファイル用のディレクトリ。推奨されるパーミッション = 700。

`/etc/ssh2/subconfig/<subconfig_file>`

ユーザ固有およびホスト固有のサブ構成ファイル。詳細については、sshd2_config(5) の「SUBCONFIGURATION FILES」を参照してください。

`/etc/ssh2/environment`

このファイルが存在する場合、このサーバへのすべての Secure Shell クライアントの接続に使用できるように、環境変数が設定されます(キーワード **SettableEnvironmentVars** によっても、設定可能な環境変数が制御されます)。推奨されるパーミッション = 644。注意: このファイルで指定された環境変数の設定は、`/etc/default/login` や `/etc/environment` などの標準のシステムファイルで構成される値より優先されます。このグローバルファイルとユーザ固有の環境ファイル (`~/.ssh2/environment`) で同じ環境変数が構成されている場合、グローバル値よりユーザ固有の値のほうが優先されます。ポンド記号 (#) の付いた行はコメント行です。構文は次のとおりです。

`environment_variable=value`

`/etc/nologin`

ログインをルートに制限します。このファイルが存在する場合、ルートのみがログインを許可されます。ログインを試行した以外のユーザには、`nologin` というテキストが表示されます。

`<piddir>/sshd2_22.pid`

接続を受け付けているプロセスの PID が含まれます。PID ディレクトリは、ご使用のオペレーティングシステムによって決まります。この名前に符号化されるポート番号 (既定は 22) は、**Port** キーワードの値によって決まります。**PidFile** キーワードを使用して、別の名前または場所を指定することもできます。

`/etc/motd`

その日のメッセージファイル。このファイルのテキストは、ユーザのログイン時に表示されます。この表示をオフにするには、**PrintMotd** キーワードを使用します。

`/etc/ssh2/radius_config`

1 台以上の RADIUS 認証サーバを記述した、ユーザ作成のファイル。上で提案したファイル名は必要ありません。このファイルの作成後に、**RadiusFile** キーワードを使用してファイル名を指定することができます。各 RADIUS サーバに対して、名前、ポート、共有秘密を入力する必要があります。推奨されるパーミッション = 600。構文は次のとおりです。

server1:*port1*:*shared_secret1*

server2:*port2*:*shared_secret2*

ユーザ固有のサーバファイル

`~/.ssh2`

サーバ上のユーザ固有のファイルの既定ディレクトリ(**UserConfigDirectory** キーワードを使用して、別の場所を指定することもできます)。推奨されるパーミッション = 700。

`~/.ssh2/authorization`

既定のクライアント「authorization」ファイル(**AuthorizationFile** キーワードを使用して、別のファイルを指定することもできます)。このファイルは、クライアントユーザの Secure Shell 公開鍵認証に必要です。各ユーザは、そのユーザのディレクトリに「authorization」ファイルを所有している必要があります。このファイルのアクセスは、ユーザのみの書き込みアクセスに制限する必要があります。権限の制限が十分でない場合、公開鍵の認証は失敗します。推奨されるパーミッション = 600。

このファイルには、公開鍵認証中にサーバが使用する鍵ファイルの一覧が含まれます。クライアントによって提示された鍵が「authorization」ファイル内にあるいずれの鍵にも一致しない場合、公開鍵認証は失敗します。キーワードには大文字と小文字の区別がなく、ポンド記号 (#) の付いた行はコメント行です。対応しているキーワードは次のとおりです。

key

このユーザに対してサーバが受け入れる鍵を指定します。鍵エントリの形式は、「key」の後に公開鍵を含むファイルの名前を続けます。絶対パスを指定する場合を除いて、鍵はユーザ固有の構成ディレクトリ (既定は `~/.ssh2`) 内にあることが前提となります。たとえば、次の行によって、ユーザは、指定した鍵のいずれかを使用して認証することを許可されます。

```
key mykey.pub
key id_rsa_2048_a.pub
```

options

直前の鍵に適用されるオプションを指定するには、このオプションキーワードを使用します。特定の鍵のすべてのオプションを、1 行に構成する必要があります。空白を使用することができます。オプションは、鍵を含む行の直後の行に構成する必要があります。形式は次のとおりです。

```
Options option_keyword="arg", [option_keyword="arg"],...
```

3 種類の **Options** キーワードが利用できます。**command**、**allow-from**、および **deny-from** です。

command *command*

指定されたコマンドがリモートホストで実行され、その後接続が閉じられます。たとえば、次の構成では、mykey.pub が認証に使用されるたびに、スクリプト「myscript」が実行されます。

```
key mykey.pub
options command="sh myscript"
```

allow-from *IP-address*

鍵は、指定した IP アドレスからの接続用にのみ使用できます。例えば、次の構成では、指定した鍵を「150.」と「10.10.」で始まる IP アドレスからの接続にのみ使用できるようにします。

```
Key /home/joe/.ssh/mykey.pub
options allow-from="150\..*,10\.10\..*"
```

deny-from *IP-address*

鍵は、指定した IP アドレスからの接続用に使用できません。

注意: 許可リストまたは拒否リストにアドレスを構成するには、IPv4 と IPv6 の両方のアドレスを指定する必要があります。拒否リストによってアクセスがブロックされるように構成する場合は、このことは特に重要です。 許可リストまたは拒否リストにローカルホストを構成するには、すべての外部インタフェースの IP アドレスと、ローカルループバックアドレス (127.0.0.1 および 0: 0: 0: 0: 0: 0: 0: 1) を記述します。

`~/.hushlogin`

このファイルが存在する場合、ユーザの最後のログイン、その日のメッセージ、およびメール確認は表示されません。

`~/.ssh2/environment`

このファイルが存在する場合、このユーザのログイン時に設定するための環境変数が設定されます(キーワード **SettableEnvironmentVars** によっても、設定可能な環境変数が制御されます)。推奨されるパーミッション = 644。注意: このファイルで指定された環境変数の設定は、`/etc/default/login` や `/etc/environment` などの標準のシステムファイルで構成される値や、グローバルファイル (`/etc/ssh2/environment`) 内で構成された設定より優先されます。ポンド記号 (#) の付いた行はコメント行です。構文は次のとおりです。

```
environment_variable=value
```

付 録 C

クライアント構成キーワード

次の設定をクライアント構成ファイルで設定できます。(グローバルファイルは `/etc/ssh2/ssh2_config` です。ユーザ固有のファイルは `~/.ssh2/ssh2_config` です。)また、これらの設定を **ssh** コマンドラインで **-o** オプションを使用して構成することもできます。

AddressFamily

クライアントで対応可能なアドレス形式を指定します。指定可能な値は、「any」(使用するアドレスファミリをシステムで決定)、「inet」 (IPv4 のみが使用可能)、および「inet6」 (IPv6 が好ましいが IPv4 も使用可能) です。既定は「inet」です。また、次を使用して、優先するアドレスファミリを構成することもできます (**-4** および **-6** コマンドラインオプション)。

AllowedAuthentications

クライアントが試行する認証方法と、その順番を指定します。対応方法には、「gssapi-keyex」、「gssapi-with-mic」、「publickey」、「keyboard-interactive」、および「password」があります。対応方法を指定するには、カンマ区切り一覧を使用します。クライアントは先頭から末尾への順序で認証方法を試行します。接続に使用される認証方法は、サーバで許可されている方法のうち、クライアントの優先順序で最も順位が高い方法です。サーバが複数の方法を要求するように構成されている場合、接続を確立するためには複数の認証方法が必要です。自動化スクリプトに対応するためには、対話が最も少ない方法を一覧の先頭に置く必要があります。既定は「gssapi-with-mic」、「publickey」、「keyboard-interactive」、「password」です。

AuthenticationSuccessMsg

認証が正常に完了した時に、次のメッセージを表示するかどうかを指定します。「Authentication successful.」 許可される値は「yes」および「no」です。既定は「yes」です。

BatchMode

パスワードやパスフレーズの入力を求めるなどのユーザ入力を求めるすべてのクエリーを無効にするかどうかを指定します。これは、スクリプトジョブやバッチジョブで役立ちます。ここで **StrictHostKeyChecking** の設定を「ask」、 **BatchMode** の設定を「yes」にした場合、クライアントではホスト鍵が不明なクエリーに対する応答がすべて「no」となります。 許可される値は「yes」および「no」です。既定は「no」です。

CheckHostIP

公開鍵ファイル名にエンコードされたホスト名および IP アドレスを使用してホスト IP アドレスの確認を実行するかどうかを指定します。ユーザが新しいホスト鍵を受け入れると、その鍵は `key_port_host,IP.pub` 形式を使用して既知のホストストアに追加されます。 ここで **CheckHostIP** を有効にしている場合に、指定されたホストの実際の IP がそのホストのエンコードされた IP アドレスに一致しないと、ホスト認証は失敗します。この設定を有効にすると、ホスト鍵が変更された場合に DNS スプーフィングを検出するのに役立ちます。 許可される値は「yes」および「no」です。既定は「no」です。

注意: v. 7.0 より前のバージョンを使用してホスト鍵ストアに追加されたホスト鍵には、ホストの IP アドレスが含まれません。古い形式の鍵を使用する場合は、**CheckHostIP** を無効にしてください。

CheckpointResume

この設定を「yes」 (既定) にすると、中断したファイル転送が中断の時点から再開されます。この設定を「no」にすると、転送は必ず最初から開始されます。注意: また、サーバで次を使用することにより、チェックポイント再開を無効にすることができます (**SmartFileTransfer** キーワード)。

Ciphers

対応する 1 つ以上の (カンマで区切った) 暗号化アルゴリズムを指定します。 特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。 許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。

この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。

あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと 128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。

また、暗号化アルゴリズムを指定することができます。 **ssh** コマンドラインで次を使用します (**-c** オプション)。既定値は「AnyStdCipher」です。

ClearAllForwardings

ローカル転送、リモート転送、または動的転送されたポートのうち、既に処理されたすべてのポートを構成ファイルまたはコマンドラインからクリアします。 許可される値は「yes」および「no」です。既定は「no」です。 注意: **scp** および **sftp** は、この設定に関係なく、すべての転送ポートを自動的にクリアします。

Compression

圧縮のレベルを指定します。0 ～ 9 の圧縮の値を指定することができます。値を大きくすると、圧縮量も大きくなります。大きな値を使用すると、ネットワークの帯域幅が小さくなりますが、CPU サイクルは大きくなります。レベル 6 は「yes」に相当します。レベル 0 は「no」に相当します。 既定値は「no」 (0) です。

注意: 次を使用することにより、圧縮を無効にすることができますが (**ssh** コマンドラインで次を使用します (**-C** オプション))、有効にするにはこのキーワードが必要です。

ConnectionReuse

新しい **ssh** 、 **scp** 、および **sftp** セッションで確立済みの接続の再使用を許可するかどうかを指定します。この機能を使用すると、再認証せずに新しいセッションを開始できます。 許可される値は「yes」および「no」です。既定は「no」です。 「yes」に設定すると、ターゲットホスト、ポート、ユーザがすべて確立済みの接続のものと同じ場合に、新しいセッションで既存のトンネルが再使用されます。「no」に設定すると、クライアントはセッションごとに新しい接続を確立します。つまり、新しい接続はそれぞれ認証プロセスを繰り返し、変更された接続固有のすべての設定 (転送や暗号など) を適用します。

注意: サーバ管理者が次を使用してディレクトリへのアクセスを制限している場合、接続の再使用は失敗することがあります (**ChrootSftpGroups** または **ChrootSftpUsers**)。

ConnectionTimeout

クライアントがサーバへの接続を試行する際に、クライアントが待機する最大時間 (秒単位) を指定します。既定では 0 (ゼロ) に設定されます。つまり、クライアントは制限を設定せず、実際の制限はオペレーティングシステムによって決定されます。

DefaultDomain

既定のドメイン名を指定します。コマンドラインでは短いホスト名を入力し、接続には完全修飾ドメイン名を送信できるようにするには、構成ファイルにこの設定を追加します。ここで **DefaultDomain** の値を構成していて、「.」(ドット) を含まないホスト名を入力した場合、 **DefaultDomain** の値が結合されて「.」を使用したホスト名になります。(注意: ここで **DefaultDomain** 文字列の最初にオプションでドットを組み込むことができますが、この文字列の最初の「.」は無視されます。)任意の英数文字が値として使用可能です。例えば、 **DefaultDomain** を「acme.com」または「.acme.com」に設定すると、コマンド「ssh joe@myhost」は「ssh joe@myhost.acme.com」として送信されます。

DontReadStdin

stdin を /dev/null からリダイレクトするため、stdin からの読み取りを防止します。また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-N** オプション)。 許可される値は「yes」および「no」です。既定は「no」です。

EscapeChar

端末セッションのエスケープ文字を設定します。既定の文字はチルダ (~) です。エスケープ文字を「none」に設定すると、使用できるエスケープ文字はなくなり、チルダは他の文字と同様に機能します。 詳細については、次の「エスケープシーケンス」を参照してください (**ssh** マニュアルページ)。また、エスケープ文字を設定することができます。 **ssh** コマンドラインで次を使用します (**-e** オプション)。

ExitOnForwardFailure

要求されたすべてのダイナミック、ローカル、およびリモートのポート転送を構成することができない場合に **ssh** が接続を終了するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

FileCopyAsciiExtensions

自動モード転送が有効になっているときに、 **sftp** セッション中に ASCII 転送を使用するファイルの種類を指定します。指定した以外のファイルはバイナリ転送となります。カンマまたは空白区切りのリストを指定します。ワイルドカード (zsh-glob) 文字を指定できます。ファイル拡張子の前にピリオドを付けてはなりません。空白を含む拡張子を指定するには、その拡張子を引用符で囲むか、エスケープ文字としてバックスラッシュを使用します。既定は「txt, htm*, pl, php*」です。(**setext** を使用して、 **sftp** セッション中にそのセッションの別のファイルリストを指定することができます。次を使用します (**getext**)。現在のリストを表示します。)

注意: この設定は、自動転送が有効になっている場合にのみ関係するものです。既定では転送モードはバイナリに設定されています。自動転送を有効にするには、**sftp** コマンドの「auto」を使用します。現在の転送モードを表示するには、「ascii -s」を使用します。

FipsMode

すべての接続で、FIPS 140-2 仕様に準拠したセキュリティプロトコルおよびアルゴリズムを使用するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

ForcePTTYAllocation

コマンドが指定されている場合も TTY を強制的に割り当てます。 許可される値は「yes」および「no」です。既定は「no」です。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-t** オプション)。

ForwardAgent

認証エージェントへの接続が確立されている場合に、その接続をリモートマシンに転送するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

ForwardX11

X11 接続の転送を有効にし、X11 クライアントを信頼されないものとして扱います。ゲストのリモート X11 クライアントは、信頼される X11 クライアントに属するデータを不正に変更できません。 許可される値は「yes」および「no」です。既定は「yes」です。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-X** オプション)。

GatewayPorts

ゲートウェイポート設定は、ローカルに転送されるポートをリモートアプリケーションが使用できるかどうかを制御します。既定ではこの設定は無効で、クライアントは、ローカルポート転送のためにソケットを開いた時に、ループバックアドレス (「localhost」または 127.0.0.1) を使用します。これにより、他のコンピュータで実行中のアプリケーションは、転送されるポートに接続できなくなります。ゲートウェイポートを有効にすると、リモートアプリケーションクライアントは、Secure Shell クライアントの Ethernet アドレス (IP アドレス、URL、DNS 名など) を使用してソケットを開くことができます。例えば、acme.com で実行中の Secure Shell クライアントがポート 8088 を転送するよう構成されているとします。ゲートウェイポートが無効な場合、転送されるソケットは localhost: 8088 です。ゲートウェイポートが有効な場合、転送されるソケットは acme.com: 8088 です。 許可される値は「yes」および「no」です。既定は「no」です。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-g** オプション)。

注意: クライアントではリモートホスト接続の認証が行われないため、このオプションの使用には細心の注意が必要です (インターネット接続用のネットワークアダプタで使用しないでください)。この接続が転送されるアプリケーションで独自の認証を行わない場合、すべてのリモートホスト接続においてそのアプリケーションへの無制限のアクセスが許可されます。

GoBackground

ポート転送が設定されていて Scure Shell セッションをバックグラウンドで実行したい場合に、このキーワードを使用します。許可される値は、「yes」、「no」、および「oneshot」です。既定は「no」です。少なくとも 1 つのポート転送ルールが構成されている場合、「yes」および「oneshot」の設定では、認証の完了後にセッションがバックグラウンドに送信されます。「yes」を指定した場合、プロセスを手動で終了するまで、Secure Shell セッションはバックグラウンドに留まり、無限に転送要求を受け付けます(これは、 **-f** を **ssh** コマンドラインで使用するのと同様です。) 「oneshot」を指定した場合、バックグラウンドセッションは転送された 1 つだけの接続の開始を待機し、転送された接続が閉じられるとすぐに終了します(これは、 **-fo** を **ssh** コマンドラインで使用するのと同様です。)

GSSAPIDelegateCredentials

GSSAPI 資格情報をサーバに転送 (委任) するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

Host

接続に使用する実際のホスト名または IP アドレスを指定します。既定は空の文字列です。このキーワードは、ホストスタンザ式と組み合わせて使用すると、ホストに接続するための代替名を作成できます。このキーワードがすべてのスタンザの外に出現する場合、接続の既定ホストを指定するのに使用できます。

HostCA

このキーワードはお勧めしません。これは、次の同義語です (**TrustAnchor**)。

HostCANoCRLs

このキーワードはお勧めしません。これは、次の同義語です (**TrustAnchor**)。注意: 証明書破棄の確認は、Reflection for Secure IT 構成ファイルでは構成できません。破棄の確認を構成するには、Reflection PKI Services Manager を使用してください。

HostCertNameCheck

証明書を使用したサーバ認証でホスト名の確認が必要かどうかを指定します。ここで **HostCertNameCheck** の設定が「yes」の場合、接続用に指定されたホスト名または IP アドレスが証明書用の許可される ID のセットに含まれている場合のみ、認証に成功します。(許可される ID を構成するには、PKI Services Manager マップファイルを使用してください。)ここで **HostCertNameCheck** の設定が「no」の場合、クライアントは許可される ID のセットを無視し、有効な証明書をすべて受け入れます。ここで **HostCertNameCheck** の設定が「ask」 (既定) の場合、サーバ名が許可される ID でなければプロンプトが表示され、続行するかどうか確認されます。

HostKeyAlgorithms

クライアントが提案するホスト鍵アルゴリズムを、優先順に指定します。既定値は「x509v3-rsa2048-sha256, x509v3-sign-rsa, x509v3-sign-dss, ssh-rsa-sha2-256@attachmate.com, ssh-rsa,ssh-dss」です。この設定は、証明書と標準ホスト鍵認証の両方をサーバに構成する場合に便利です。既定値では、通常の SSH 鍵アルゴリズムの前に x509 アルゴリズムが指定されています。Secure Shell プロトコルでは、1 回のホストの認証試行しか行えません。ホストが証明書を提示し、証明書を使用したホスト認証がクライアントに構成されていない場合、接続に失敗します(これは、複数の認証方式と試行に対応しているユーザ認証とは異なります)。この場合、優先順位を「,ssh-rsa-sha2-256@attachmate.com, ssh-rsa, ssh-dss, x509v3-rsa2048-sha256, x509v3-sign-rsa,x509v3-sign-dss」に変更して、証明書よりも SSH 鍵が優先されるようにクライアントを構成します。

HostKeyAlias

ホスト鍵をクライアントの既知のホスト鍵のディレクトリに保存する場合に実際のホスト名の代わりに使用するエイリアスを指定します。ホスト鍵は、この命名形式 `key_port_host,IP.pub` 形式を使用して既知のホストストアに追加されます。指定する値により、保存されるホスト鍵の名前の `host` の部分が置換されます。このオプションは、Secure Shell 接続のトンネリングを行う場合や、1 台のホスト上で複数のサーバが動作している場合に便利です。

HPNDisabled

パフォーマンスの向上のために Reflection for Secure IT で HPN ダイナミック TCP ウィンドウ機能を使用するかどうかを指定します。HPNDisabled = 「no」 (既定) と設定すると、最適なパフォーマンスが得られるように、Reflection for Secure IT によって TCP ウィンドウと TCP 受信バッファが調節されます。**HPNDisabled** を「yes」にすると、受信バッファは 64KB に設定されます。

IdentificationFile

公開鍵認証に使用する代替 ID ファイルを指定します。ファイルの場所は、完全修飾パスまたは相対パスを指定しないかぎり現在の作業ディレクトリと見なされます。既定の ID ファイルは `~/.ssh2/identification` です。 詳細は、以下の「ファイル」のセクションを参照してください。また、この構成を行うことができます。**ssh** コマンドラインで次を使用します (**-i** オプション)。

IdentityFile

このキーワードはお勧めしません。次を使用してください (**IdentificationFile**)。

KeepAlive

クライアントがサーバに TCP キープアライブメッセージを送信するかどうかを指定します。このキーワードはお勧めしません。次を使用します (**ServerAliveInterval** を代わりに使用)。 許可される値は「yes」および「no」です。既定は「yes」です。

KEXs

クライアントが対応している鍵交換アルゴリズムを指定します。 対応している値は、「diffie-hellman-group-exchange-sha256」、「diffie-hellman-group-exchange-sha1」、「diffie-hellman-group14-sha1」、「diffie-hellman-group1-sha1」です。カンマ区切りのリストとして、複数のアルゴリズムを指定することができます。既定値は「diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1,diffie-hellman-group1-sha1」です。

LibGssKrb5

GSSAPI (Kerberos 5) 認証を使用する場合は、この設定を使用します。この設定は、libgssapi_krb5.so という名前の Kerberos ライブラリへの完全修飾パスを指定します。

LocalForward

このキーワードは、クライアントの任意のポートから、安全なトンネルを介して接続を転送する場合に使用します。この設定を構成するための構文は、次のとおりです。

[*protocol/*][*listening_host*:]*listening_port*: *host*: *hostport*

Secure Shell 接続が確立されている場合、Secure Shell クライアントは指定のローカルポート (*listening_port*) を使用して、Secure Shell クライアントホストのソケットを開きます(複数のインタフェースを使用しているクライアントホストでは、*listening_host* を使用してインタフェースを指定してください)。(転送するデータが属する) アプリケーションクライアントは、転送されるソケットにデータを送信するように構成します (宛先のホストおよびポートに直接送信しません)。そのクライアントが接続を確立すると、転送されるポートに送信されるデータはすべて、安全なトンネルを介して Secure Shell サーバにリダイレクトされます。このサーバではデータが復号化され、宛先のソケット (host,hostport) に送信されます。ゲートウェイポートオプションが有効でないかぎり、転送されるローカルポートは、Secure Shell クライアントと同じコンピュータで実行されているクライアントのみが使用できます。オプションの *protocol* は、**tcp** または **ftp** にすることができます。

注意: 最終宛先ホストおよびポートが Secure Shell サーバホスト上にない場合、Secure Shell ホストとアプリケーションサーバホスト間でデータは平文で送信されます。

次の例は、Secure Shell クライアントと同じコンピュータで実行されているメールクライアントと、Secure Shell サーバと同じコンピュータで実行されているメールサーバの間の電子メール通信を安全にするためにローカル転送を使用しています。ローカルメールクライアントは、ローカルポート 14300 に通信を送信するように構成されています。ポート 14300 で受信したデータは、安全なトンネルを介してサーバに転送され、ここでさらにポート 143 に転送されます。

```
LocalForward=14300:localhost:143
```

以下の例では、(Secure Shell クライアントと同じコンピュータ上の) FTP クライアントから送信された FTP 通信は、myhost.com 上で動作している FTP サーバに転送されます。この構成では、FTP クライアントを localhost:2121 に接続するよう構成します。

```
LocalForward=ftp/2121:myhost.com:21
```

また、ローカル転送を構成することができます。 **ssh** コマンドラインで次を使用します (**-L** オプション)。

LogLevel

syslog にログ記録される **ssh** メッセージに使用される冗長レベルを設定します。使用可能な値は、「fatal」、「error」、「quiet」、「info」、「verbose」、「debug1」 (「debug」は 1 に相当)、「debug2」 (2 に相当)、「debug3」 (3 に相当)、および「trace」 (「debug99」は 99 に相当) です。これらの値と関連する syslog レベルは、fatal は CRIT、error および quiet は ERROR、info および verbose は INFO、debug1、debug2、debug3、および trace はDEBUG です。 既定値は「info」です。

注意: 記録の設定を「trace」にすると、セキュリティのリスクが高まります。このレベルでは、暗号化されていないプロトコル情報が書き出されることがあるため、情報の漏洩が懸念されます。また、大量に書き込まれる情報によりディスクスペースが急速に占有され、ホストや Reflection for Secure IT の応答が停止する可能性があります。

MACs

クライアントが対応する MAC (メッセージ認証コード) を優先順に指定します。許可される値は、「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」、および「hmac-ripemd160」です。これらすべてに対応するには「AnyMac」を使用します。「AnyStdMac」を使用して「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」を指定します。hmac-sha256 を指定すると hmac-sha2-256 も有効になります。hmac-sha512 を指定すると hmac-sha2-512 も有効になります。複数の MAC をカンマ区切り一覧として指定することもできます。その他には、「none」、「any」 (AnyMac + 「none」と同等)、および「AnyStd」 (「AnyStdMac」 + 「none」と同等) オプションがあります。MAC について「none」で合意すると、メッセージ認証コードは使用されません。この場合データの整合性は保護されないため、「none」を含むオプションはお勧めしません。

また、MAC を構成することができます。 **ssh** コマンドラインで次を使用します (**-m** オプション)。既定は「AnyStdMac」です。

NoHostAuthenticationForLocalHost

このオプションは、クライアントをローカルホストに接続する場合にホスト認証を無効にします。ホームディレクトリを複数のコンピュータで共有している場合に便利です。この場合、ローカルホストは各コンピュータ上の異なるホストを指し、クライアントユーザはホスト鍵の変更に関する多数の警告を受けます。この設定を「yes」にすることにより、ローカルホストの認証が無効になり、このような警告が表示されなくなります。 許可される値は「yes」および「no」です。既定は「no」です。

NumberOfPasswordPrompts

パスワードプロンプトに対して応答できる回数を指定します。注意: サーバ側でも許可されるパスワードの最大試行回数を設定できます。
ここで **NumberOfPasswordPrompts** をサーバで構成されている値よりも大きい値に設定すると、サーバ側の制限に達した時に接続は失敗します。既定値は 3 です。

PasswordPrompt

パスワード認証用に表示するプロンプトを指定します。2 つの変数が使用でき、%r はユーザ名に置換され、%h はホスト名に置換されます。
既定値は「`%r@%h's password:`」です(この設定はキーボード対話型型認証には影響しません)。

PkidAddress

PKI Services Manager への接続に使用するポートを指定します。*`host`*`:` *`port`* という形式を使用します。既定は `localhost:18081` です。ホストを指定してポートを省略した場合、既定の PKI Services Manager ポート (18081) が使用されます。

PkidPublicKey

Reflection PKI Services Manager の ID を確認するために使用される公開鍵の名前と場所を指定します。既定は `/opt/attachmate/pkid/config/pki_key.pub` です。

Port

サーバ上の接続先ポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-p** オプション)。

ProxyCommand

サーバへの接続に使用するコマンドを指定します。コマンド文字列は行末まで拡張され、ユーザのシェルによって実行されます。このコマンドでは 2 つの変数が使用でき、「%h」はホスト名に置換され、「%p」はポートに置換されます。コマンドとしては、stdin からの読み込みと stdout への書き込みが可能な任意のものが使用できます。コマンドは最終的に Reflection for Secure IT サーバに接続する必要があります。次を使用して (**ProxyCommand** を、プロキシに対応し **nc** (または **netcat**) などのコマンドと一緒に使用することができます。例えば、以下のコマンドは、 **nc** を使用して 198.168.2.1 の HTTP プロキシ経由で接続します。

```
ProxyCommand /usr/bin/nc -X connect -x 198.168.2.1:8080 %h %p
```

既定は「none」で、このオプションを無効にします(空の文字列を指定するのと同じです)。

注意: **CheckHostIP**は、プロキシコマンドで確立した接続には使用できません。

QuietMode

クワイエットモードを有効にします。このモードでは、バナーを含むすべての警告および診断メッセージが表示されません。 許可される値は「yes」および「no」です。既定は「no」です。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-q** オプション)。

RekeyIntervalSeconds

クライアントが新しいセッション鍵の交渉を開始する前に待機する秒数を指定します。値は整数である必要があります。既定は 3600 です。この鍵は、 **RekeyLimit** と組み合わせて使用することができます。この場合、最初の制限に達するとクライアントで新しい鍵の交換が開始されます。

RekeyLimit

クライアントが新しいセッション鍵の交渉を開始する前に、送信できる最大データ量を指定します。引数はバイト数で、接尾語「K」、「M」、または「G」をオプションで使用できます。これらは、それぞれキロバイト、メガバイト、ギガバイトを示します。既定値を使用するには、この値を 0 (ゼロ) に設定します。既定は「1G」と「4G」の間で、暗号によって決まります。この鍵は、 **RekeyIntervalSeconds**と組み合わせて使用することができます。この場合、最初の制限に達するとクライアントで新しい鍵の交換が開始されます。

RelaySignals

クライアントがサーバにリレーすべき信号を指定します。 **RelaySignals** にカンマ区切りのリストとして指定できる信号は、ABRT、ALRM、FPE、HUP、ILL、INT、PIPE、QUIT、SEGV、TERM、USR1、USR2 です。信号 KILL および STOP は、捕捉、ブロック、無視することができないため、これらの信号には対応していません。既定では、信号のリレーは行われません。

RemoteForward

このキーワードは、サーバ上の任意のポートから、安全なトンネルを介して接続を転送する場合に使用します。この設定を構成するための構文は、次のとおりです。

[*protocol*/][*listening_host*:]*listening_port*:*host*:*hostport*

Secure Shell 接続が確立されている場合、Secure Shell サーバは指定のリモートポート (*listening_port*) を使用して、そのホスト (Secure Shell サーバが動作しているコンピュータ) のソケットを開きます(複数のインタフェースを使用しているサーバホストでは、*listening_host* を使用してインタフェースを指定してください)。(転送するデータが属する) クライアントアプリケーションは、転送されるソケットにデータを送信するように構成します (宛先のホストおよびポートに直接送信しません)。そのクライアントが接続を確立すると、転送されるポートに送信されるデータはすべて、安全なトンネルを介して Secure Shell クライアントにリダイレクトされます。ここではデータが復号化され、宛先のソケット (host,hostport) に送信されます。オプションの *protocol* は、**tcp** または **ftp** にすることができます。

以下の例では、(Secure Shell サーバと同じコンピュータ上の) FTP クライアントから送信された FTP 通信は、(Secure Shell クライアントと同じコンピュータ上の) FTP サーバに転送されます。この構成では、FTP クライアントをポート 3333 に接続するよう構成します。

```
RemoteForward=ftp/3333:localhost:21
```

また、リモートポート転送を構成することができます。 **ssh** コマンドラインで次を使用します (**-R** オプション)。

SendNOOPPackets

クライアントが Secure Shell チャネル経由でサーバに NOOP メッセージを送信するかどうかを指定します。この設定を「yes」にした場合は、 **ServerAliveCountMax** を 3 に、 **ServerAliveInterval** を 600 に設定した場合と同じです。 許可される値は「yes」および「no」です。既定は「no」です。

ServerAliveCountMax

応答しなくなったサーバに対するセッションを閉じるには、この設定を使用します。これは、 **ServerAliveInterval** を 0 以外の値に設定した場合のみ該当する項目です。**ServerAliveCountMax** は、クライアントがサーバからの戻りメッセージを確認できない場合に送信する、サーバアライブメッセージの最大数を設定します。このしきい値に達すると、クライアントはセッションを終了します。既定は 3 です。例えば、**ServerAliveInterval** の設定値が 600 、 **ServerAliveCountMax** の設定値が 3 の場合、サーバからの応答がない場合に、クライアントからサーバに 10 分ごとに最大 3 通のメッセージを送信します。つまりクライアントは約 30 分後に、応答のない接続を閉じます。

ServerAliveInterval

Secure Shell チャネルを介してサーバに NOOP メッセージを送信する間隔を秒単位で指定します。クライアントは、指定された間隔の間にサーバからデータが受信されないと、サーバにメッセージを送信します。このキーワードをゼロ以外の値に設定すると、Secure Shell サーバおよび TCP スタックにクライアントが引き続きアライブであることを知らせ、またすべてのネットワークハードウェア (ルータや NAT など) に Secure Shell 接続が引き続きアライブであることを知らせることができるほか、ネットワークの問題やアプリケーションの問題を検出することができます。無応答になったサーバへの接続を終了するには、 **ServerAliveCountMax** と一緒にこの設定を使用します。既定は 0 です。この場合、クライアントはメッセージを送信しません。

SetRemoteEnv

シェルまたはコマンドの実行前に、サーバに設定する環境変数を指定します。値の形式は、VAR=val とします。ここで val は空のままにすることもできます。引数は大文字で指定する必要があります。

注意: このキーワードで設定される値は累積されます。このキーワードを 1 つまたは複数の構成ファイルで複数回構成することで、複数の値を設定できます。

SftpVersion

クライアントが対応するSFTP プロトコルの最大バージョンを指定します。有効な値は 3 および 4 (既定値) です。サーバが、この設定で指定するバージョンより古いバージョンにしか対応していない場合、サーバが指定するバージョンが使用されます。

SmartFileCopy

ファイル転送の実行前に、Reflection for Secure IT でファイルの同一性の確認を行うかどうかを指定します。この設定を「yes」 (既定) にすると、ファイルの同一性が確認され、ファイルが同じ場合はデータ転送が行われません。この設定を「no」にすると、同一性の確認が行われず、ファイルは常に上書きされます。注意: また、サーバで次を使用することにより、スマートファイルコピーを無効にすることができます (**SmartFileTransfer** キーワード)。

StrictHostKeyChecking

このキーワードは、認証の際にホストが不明な鍵を提示した時のクライアントの動作を決定します。指定可能な値は、

「yes」 - ホスト鍵が手動でユーザのホスト鍵ディレクトリ (`~/.ssh2/hostkeys`)、または次のシステム全体のホスト鍵ディレクトリにコピーされた場合のみ、接続に成功します (`(/etc/ssh2/hostkeys)`)。クライアントはホスト鍵をユーザのコンピュータに追加しません。これは、最も安全なオプションです。

「ask」 - これが既定値です。クライアントは、不明なホストからの鍵を受け入れるかどうかをユーザにたずねるプロンプトを表示します。このプロンプトにはホスト鍵の指紋が表示されるため、ホストの識別情報を確認するのに使用できます。ユーザが「yes」を指定すると、クライアントでホスト鍵がユーザのディレクトリ (`~/.ssh2/hostkeys`) 内の既知のホスト鍵に追加され、その後の接続でホストの ID の確認のためにこの鍵が使用されます。

「no」 - ユーザのホスト鍵ディレクトリ (`~/.ssh2/hostkeys`) に未知のホスト鍵が自動的に追加され、その後の接続でホストの ID の確認のために使用されます。ユーザは不明なホスト鍵がいつ提示されたかを知ることはできません。

StrictModes

公開鍵認証に必要なファイルとディレクトリのパーミッションを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。 設定を「yes」にする場合、ユーザのディレクトリ (`~/.ssh2/`) とすべての親ディレクトリの書き込みと実行がそのユーザによってのみ可能になるようにする必要があります (モード 755 が設定可能)。ユーザのディレクトリの推奨のパーミッションは 700 です。ユーザ ID ファイル (既定では`~/.ssh2/identification`) は、ユーザのみの読み/書きアクセスが可能となるよう構成する必要があります (600 を推奨、644 も設定可能)。設定を「no」にする場合、このファイルパーミッションが必須でないため、ファイルや情報の機密保持が保証されない可能性があります。

注意: 秘密鍵には、これ以外のファイルパーミッションの制限が課されます。現在の **StrictModes** の設定に関係なく、ユーザのみの読み取りアクセスが可能なように鍵を構成する必要があります。秘密鍵へのアクセスの制限が十分でない場合、公開鍵の認証は常に失敗します。秘密鍵の推奨のパーミッションは 600 です。

SysLogFacility

次の接続用のログメッセージに使用する機能コードを指定します (**ssh**、 **sftp**、および **scp**)。既定は「USER」です。この値が「none」の場合、Reflection for Secure IT は syslog への記録を無効にします。その他の有効な値はプラットフォームに依存します。syslog(3) および syslog.conf(5) を参照してください。

TrustAnchor

このキーワードはオプションで、サーバ認証に証明書をする場合のみ該当します。既定では、Reflection PKI Services Manager は、構成したすべてのトラストアンカを使用して、認証用に提供された証明書を検証します。証明書の検証に使用する Reflection PKI Services Manager トラストアンカを制限するには、このキーワードを使用します。PKI Services Manager ストアでの証明書の Subject DN (識別名) を指定するか、または証明書のファイル名を使用することができます。指定するトラストアンカは、(PKI Services Manager **TrustAnchor** キーワードを使用して) Reflection PKI Services Manager にも構成する必要があります。

TrustX11Applications

X サーバが、転送された X11 クライアントアプリケーションを信頼されるものとして扱うかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。 リモートの X11 クライアントに X11 ディスプレイへのフルアクセスを許可するには、この設定を「yes」にします。このキーワードを「no」に設定すると、X11 アプリケーションは信頼されないものとして扱われます。そのため、侵害されたホストへの接続によって、ホスト上のアプリケーションが X11 接続を使用して入力操作を傍受するリスクを回避できます。

User

リモートサーバ用のユーザ名を指定します。この設定をホスト固有のスタンザで定義することで、ホストごとに異なるユーザ名を構成できます。既定は、環境変数 $USER の現在の値です。

VerboseMode

デバッグレベルを冗長モードに設定します。これは、デバッグレベルを 2 に設定することと同じです。 また、この構成を行うことができます。 **ssh** コマンドラインで次を使用します (**-v** オプション)。 許可される値は「yes」および「no」です。既定は「no」です。

XauthPath

xauth(1) プログラムの場所を指定します。既定値 (例えば `/usr/X11R6/bin/xauth`) はシステムによって異なります。

付 録 D

サーバ構成キーワード

サーバ構成ファイルで、以下の設定項目を構成することができます。既定のファイルは `/etc/ssh2/sshd2_config` です。

AccountManagement

アカウント管理システムを構成します。 **sshd** でユーザアカウントの確認のために使用します。アカウント管理サービスは、アカウントが有効かどうか、およびパスワードが引き続き有効かどうかを判断します。許可される値は、「password」、「pam」、「aix」、および「none」です。既定値は「pam、password」で、両方のシステムによる検証を通過するにはユーザアカウントが必要です。

pam。アカウント管理に PAM を使用します。PAM アカウント管理は、使用されている認証方法にかかわらず、すべてのセッションに適用されます。アカウントがロックされている場合、接続は拒否されます。

password。アカウントの検証にパスワードデータベースを使用します。

aix。AIX システムで使用します。ユーザが公開鍵認証での認証に成功すると、サーバはパスワードアカウントの制限事項 (期限切れのパスワードの更新要件など) を無視します。AIX の account_locked フラグによってアカウントがロックされている場合は、公開鍵認証が成功してもログインは許可されません。注意: AIX 以外のシステムでは、この値は「none」と同等です。

none。アカウント検証は使用されません。この値は、トラブルシューティングのみに使用してください。

AddressFamily

リッスン、セッション、転送用の TCP ソケットを作成するとき、サーバによってこの設定が使用されます。指定可能な値は、「any」 (使用するアドレスファミリをシステムで決定)、「inet」 (IPv4 のみが使用可能)、および「inet6」 (IPv6 のみが使用可能) です。既定値は「inet」です。注意: ここで **ListenAddress** の現在の値も、サーバで IPv4 または IPv6 アドレスによる接続が可能かどうかに影響することがあります。

AllowAgentForwarding

エージェント転送を許可するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

AllowedAuthentications

サーバが対応する認証方法を指定します クライアントとサーバは、クライアント構成とサーバ構成の両方に基づいて、初期接続プロセスの際に 1 つまたは複数の認証方法について合意します。 (**RequiredAuthentications** を使用して、1 つ以上の認証方法を要求します **RequiredAuthentications** は **AllowedAuthentications** を上書きします。)。

対応している認証方法は、「gssapi-keyex」、「gssapi-with-mic」、「publickey」、「keyboard-interactive」、および「password」です。 既定値は「gssapi-with-mic, publickey, keyboard-interactive, password」です。

AllowedPasswordAuthentications

このキーワードへの対応は終了しています。以前のバージョンで使用していた場合、手動で設定を移行する必要があります。次のキーワードを参照してください。 **AllowedAuthentications**、 **RequiredAuthentications**、および **AuthKbdInt.Required**。

AllowGroups

指定したグループのメンバーであるユーザのみにログインを許可するには、このキーワードを使用します。 正規表現に対応しています。 詳細については、「グループアクセスの構成 『88ページ 』」を参照してください。 このキーワードが構成されていない場合、すべてのグループのログインが許可されます。

AllowHosts

指定したクライアントホストのみにログインを許可するには、このキーワードを使用します。 正規表現に対応しています。 詳細については、「クライアントホストアクセスの構成 『88ページ 』」を参照してください。 このキーワードが構成されていない場合、すべてのクライアントホストが許可されます。

注意:

ホスト式をドメイン名 (IP アドレスではなく) を使用して構成している場合、この設定に加えて、**ResolveClientHostName** を「yes」に設定する必要があります。**ResolveClientHostName** が「yes」の時、解決された名前は完全修飾ドメイン名です。つまり、**RequireReverseMapping** が「yes」の時は、IP アドレスからの接続が正しく処理されるように、ホスト名に完全修飾ドメイン名を指定するか、または正規表現を使用する必要があります。

許可リストまたは拒否リストにアドレスを構成するには、IPv4 と IPv6 の両方のアドレスを指定する必要があります。拒否リストによってアクセスがブロックされるように構成する場合は、このことは特に重要です。 許可リストまたは拒否リストにローカルホストを構成するには、すべての外部インタフェースの IP アドレスと、ローカルループバックアドレス (127.0.0.1 および 0: 0: 0: 0: 0: 0: 0: 1) を記述します。

AllowSftpCommands

ユーザが実行可能な操作を制御します。 **sftp** および **scp** コマンドを Reflection for Secure IT クライアントから使用します。このキーワードには、カンマ区切りのリストとして、「all」、「none」、「browse」、「download」、「upload」、「delete」、「rename」の中から 1 つまたは複数の項目を指定できます。upload オプションを使用すると、ユーザは、サーバ上でファイルの変更、ファイルの作成、ディレクトリの作成、ファイル属性の変更が可能になります。download オプションでは、ユーザはファイルの内容を読み込むことができます。既定値は「all」です。

注意: この設定は、Reflection for Secure IT クライアントからの **sftp** と **scp** の両方の接続に影響します。**SessionRestricted** キーワードも、ファイル転送へのアクセスに影響します。**SessionRestricted** の既定値は「shell、exec、subsystem」です。Reflection for Secure IT クライアントの場合、**sftp** と **scp** の両方の転送で「subsystem」セッションタイプが必要です。OpenSSH タイプのクライアントの場合、**sftp** 転送では「subsystem」、**scp** 転送では「exec」セッションタイプが必要です。

AllowTCPForwarding

すべてのクライアントユーザにポート転送を許可または拒否するには、このキーワードを使用します。 許可される値は「yes」および「no」です。既定は「yes」です。このキーワードは、ローカル (クライアントからサーバ) とリモート (サーバからクライアントの転送) の両方を制御します。 **ForwardACL** を使用して、よりきめ細かく制御します。

AllowTCPForwardingForGroups

指定したグループのメンバーであるユーザのみにポート転送を許可するには、このキーワードを使用します。 正規表現に対応しています。

AllowTCPForwardingForUsers

指定したユーザのみにポート転送を許可するには、このキーワードを使用します。正規表現に対応しています。

AllowUsers

指定したユーザのみにログインを許可するには、このキーワードを使用します。正規表現に対応しています。 詳細については、「ユーザアクセスの構成『87ページ』」を参照してください。

AllowX11Forwarding

X11 転送を許可するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

AuditLog

監査ログを作成するかどうかを指定します。「sftp」を指定すると、ファイル転送動作の詳細レコードを含むカンマで区切られたログファイルが、 **AuditLog.Directory**で指定された場所に作成されます。監査ログファイルの最初の行には、次のようにログの内容が表示されます。「UserID」、「ClientIP」、「Action」、「ServerFilename」、「StartTime」、「EndTime」、「ServerFileModificationTime」、「ServerFileSize」、「BytesTransferred」、「Result」、「Reason」、「ServerFileHash」。
既定値は「none」です。

AuditLog.Directory

監査ログの出力場所です。日ごとの新しいログが sshd2-audit-YYYYMMDD.log の名前形式で作成されます。ここで、YYYYMMDD は日付です。AuditLog = sftp の場合、クライアントユーザによる最初のファイル転送時か、サーバの再起動時にこのファイルが作成されます。既定値は `/etc/ssh2/logs`です。

注意: sftp プロトコル接続について、ユーザが自分のホームディレクトリに限定されている場合 (**ChrootSftpUsers** または **ChrootSftpGroups**を使用する) 、監査ログディレクトリがユーザのホームディレクトリ (`AuditLog.Directory = /home/joe/logs` など) にないと転送が失敗します。

AuditLog.Sftp.WithHash

sftp ログエントリにファイルハッシュが含まれるかどうかを指定します。ハッシュ値を使用して、同一ファイルの転送を示す複数のレコードを特定することができます。変更されていないファイルが転送された場合、ログ中のハッシュ値は毎回同じになります。ファイルが変更されている場合は、ハッシュ値が異なります。 許可される値は「yes」および「no」です。既定は「yes」です。

AuthFailureErrorMessages

許可される値は「yes」および「no」です。既定は「no」です。 設定を「no」にすると、認証失敗に関する情報がクライアントに送信されません。これは SSH 規則に準拠します。この設定を有効にするには、 **AuthImmediateDisconnect**も有効にする必要があります。ここで **AuthFailureErrorMessages** および **AuthImmediateDisconnect** の両方の設定を「yes」にすると、失敗の理由に関する情報をクライアントが受信します。注意: 使用される認証方法にかかわらず、クライアントに送信されるレポートでエラーが報告されます。例えば、サーバホストでユーザアカウントが無効や不明であったり、ユーザが拒否ユーザリストに含まれていたりすると、情報がクライアントに送信されます。使用される認証方法固有のエラーに関する情報 (不正なパスワード、公開鍵がない、無効な証明書など) は提供されません。

警告: この設定を有効にすると、クライアントに有効なアカウント名に関する情報を提供することになり、セキュリティに対するリスクが高まります。

AuthImmediateDisconnect

許可される値は「yes」および「no」です。既定は「no」です。 この設定を「no」にすると、サーバは失敗したすべての認証試行に対して同じように応答します。これは SSH 規則に準拠します。この設定を「yes」にすると、ブロックされたアカウントを持つユーザは直ちに切断され、認証プロンプトが表示されないことがあります。Reflection for Secure IT サーバの設定 (**AllowUsers** または **DenyUsers**など) によりユーザのアクセスが拒否された場合、常に、また直ちに切断されます。オペレーティングシステムの構成によりユーザのアクセスが拒否された場合、切断のタイミングは **AccountManagement** の設定の影響を受けます。AccountManagement=pam という設定の場合、拒否されたユーザに対して、切断される前に PAM 認証のプロンプトが表示されます。PAM 認証が PAM アカウント管理より前に行われるためです。PAM 認証プロンプトを表示しないでユーザを切断するには、設定を AccountManagement=pam、password (既定) にします。ほとんどの場合、パスワードアカウント管理を有効にすると、PAM 認証が開始する前に接続を拒否するためのユーザアカウントに関する十分な情報がサーバに提供されます。

警告: この設定を有効にすると、クライアントに有効なアカウント名に関する情報を提供することになり、セキュリティに対するリスクが高まります。

AuthKbdInt.Required

キーボード対話型認証にどの認証方法を使用するかを指定します。ユーザの認証が成功するためには、指定した認証方法が成功する必要があります。許可される値は、「pam」、「password」、および「radius」です。既定値は「pam」で、認証およびパスワード管理に PAM モジュールを使用するよう指定されます。「password」を指定した場合、ユーザの入力が標準のログインパスワードとして処理されます。「radius」を指定した場合、1 台以上の RADIUS 認証サーバが認証用に使用されます。

AuthKbdInt.Retries

キーボード対話型認証で許可される最大試行回数を設定します。既定値は 3 です。

AuthKbdInt.Verbose

サーバが冗長なキーボード対話型プロンプトを使用するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

AuthorizationFile

公開鍵認証のユーザ鍵の構成に使用されるファイル名を指定します。公開鍵認証が成功するためには、クライアントユーザが認証のために提示した鍵が、このファイルで正しく識別される必要があります。ファイルの構文については、「ファイル」のセクションを参照してください。

絶対パスを指定した場合以外は、ファイルは `~/.ssh2` (または **UserConfigDirectory** に設定された場所) の相対パスを前提とします。次のマクロが認識されます。%U = ユーザのログイン名、%D = ユーザのホームディレクトリ、%IU = ユーザのユーザ ID、%IG = ユーザのグループ ID。既定のファイルは `%D/.ssh2/authorization` です。

AuthPublicKey.MaxSize

ユーザ認証で許可される公開鍵の最大サイズを設定します。既定値は 32768 で、これより大きい値は許可されません。許可される値の範囲は、512 ～ 32769 です。ゼロ (0) は既定と同じ意味になります。

AuthPublicKey.MinSize

ユーザ認証で許可される公開鍵の最小サイズを設定します。既定値は 512 で、これより小さい値は許可されません。ゼロ (0) は既定と同じ意味になります。

AuthPublicKey.Retries

公開鍵認証でサーバが受け入れる最大試行回数を指定します。この回数に達すると、以降、公開鍵を使用した認証の試みは拒否されます。ただし、接続は切断されません。このため、クライアントは、次に許可される方法を使用して認証を試みることができます。既定値は 100 です。

BannerMessageFile

バナーメッセージのテキストを含むファイルを識別します。サーバは、クライアントが認証する前に、クライアントにこのテキストを送信します。 注意: 一部のクライアントはバナーの表示に対応していません。バナーを構成する場合は、使用する Secure Shell クライアントがこの機能に対応していることを確認する必要があります。 既定値は `/etc/ssh2/ssh_banner_message`です。

ChrootSftpGroups

sftp プロトコル接続で、所属ユーザがホームディレクトリへのアクセスを制限されるグループを指定します。 ファイルまたはディレクトリ上で動作するすべての sftp プロトコル要求は、限定されたディレクトリまたはその子ディレクトリの範囲外でないことが確認されます。 正規表現に対応しています。 パターンの照合は、GID ではなく、グループ名で行われます。

注意: この設定は、Reflection for Secure IT クライアントからの **sftp** と **scp** の両方の接続に影響します。**SessionRestricted** キーワードも、ファイル転送へのアクセスに影響します。**SessionRestricted** の既定値は「shell、exec、subsystem」です。Reflection for Secure IT クライアントの場合、**sftp** と **scp** の両方の転送で「subsystem」セッションタイプが必要です。OpenSSH タイプのクライアントの場合、**sftp** 転送では「subsystem」、**scp** 転送では「exec」セッションタイプが必要です。

ここで **ChrootSftpUsers** または **ChrootSftpGroups** が有効になっている場合、接続されているユーザに対して、ホームディレクトリに追加されたサブディレクトリ (すべてのプラットフォーム上の`etc` および AIX 上の `dev`) も表示されます。これらのディレクトリは、移動したり削除したりすることができません。その `etc` ディレクトリには、2 つの必須ファイルが含まれます。その `rsit.conf` ファイルは、Reflection for Secure IT で必要となるファイルのインストール場所を特定するためのファイルです。その `localtime` ファイルは、記録などのプロセスが現在の時刻を取得するために必要となるファイルです。システムの `localtime` ファイルは、chroot したユーザがアクセスできない場所にあります。AIX 上で作業しているユーザは `/dev/null`も必要です。これは、syslog に正しく記録するために必要です。

ChrootSftpUsers

sftp プロトコル接続で、ホームディレクトリへのアクセスを制限されるユーザを指定します。 ファイルまたはディレクトリ上で動作するすべての sftp プロトコル要求は、限定されたディレクトリまたはその子ディレクトリの範囲外でないことが確認されます。 正規表現に対応しています。 パターンの照合は、UID ではなく、ユーザ名で行われます。

注意: この設定は、Reflection for Secure IT クライアントからの **sftp** と **scp** の両方の接続に影響します。**SessionRestricted** キーワードも、ファイル転送へのアクセスに影響します。**SessionRestricted** の既定値は「shell、exec、subsystem」です。Reflection for Secure IT クライアントの場合、**sftp** と **scp** の両方の転送で「subsystem」セッションタイプが必要です。OpenSSH タイプのクライアントの場合、**sftp** 転送では「subsystem」、**scp** 転送では「exec」セッションタイプが必要です。

ここで **ChrootSftpUsers** または **ChrootSftpGroups** が有効になっている場合、接続されているユーザに対して、ホームディレクトリに追加されたサブディレクトリ (すべてのプラットフォーム上のetc および AIX 上の dev) も表示されます。これらのディレクトリは、移動したり削除したりすることができません。その etc ディレクトリには、2 つの必須ファイルが含まれます。その rsit.conf ファイルは、Reflection for Secure IT で必要となるファイルのインストール場所を特定するためのファイルです。その localtime ファイルは、記録などのプロセスが現在の時刻を取得するために必要となるファイルです。システムの localtime ファイルは、chroot したユーザがアクセスできない場所にあります。AIX 上で作業しているユーザは /dev/nullも必要です。これは、syslog に正しく記録するために必要です。

Ciphers

サーバが対応する 1 つ以上の (カンマで区切った) 暗号化アルゴリズムを指定します。 特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。 許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。

この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。

あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと 128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。 既定値は「AnyStdCipher」です。

ClientAliveCountMax

クライアントアライブメカニズムによって、サーバはクライアントが非アクティブになった時を判断できます。 **ClientAliveCountMax** は、サーバが、クライアントからの応答を要求するために、暗号化されたチャネルを介して送信するクライアントアライブメッセージの最大数を設定します。クライアントからの応答がないままこの数字に達した場合、サーバはセッションを終了し、クライアントを切断します。メッセージの間隔は、 **ClientAliveInterval**を使用して指定します。既定値は 3 です。

注意: これらの設定は、SSH 接続に影響し、メッセージは SSH トンネルを介して送信されます。

ClientAliveInterval

クライアントアライブメッセージをクライアントに送信する間隔を秒単位で設定します。この間隔の間にクライアントから応答がない場合、サーバは、クライアントからの応答を要求するために、暗号化されたチャネルを介してメッセージを送信します。 **ClientAliveCountMax** を使用して、サーバがセッションを終了してクライアントを切断する前に、応答がなくとも送信するメッセージ数を指定します。既定値は 0 (無効) です。

Compat.RSA.HashScheme

このキーワードの使用は終了しています。7.2 SP1 より前のバージョンでは、MD5 ハッシュを使用してデジタル署名の検証を有効にするには、このキーワードを「yes」に設定する必要がありました。現在はサーバが必ず SHA-1 と MD5 の両方を使用して検証を行うため、いずれかのハッシュが一致すれば認証が許可されます。これは、以前のバージョンで Compat.RSA.HashScheme を「yes」に設定するのと同じです。Compat.RSA.HashScheme が構成ファイル内にあり「no」に設定されている場合、サーバはこの設定を無視するようになりました。

Compression

圧縮のレベルを指定します。0 ～ 9 の圧縮の値を指定することができます。値を大きくすると、圧縮量も大きくなります。大きな値を使用すると、ネットワークの帯域幅が小さくなりますが、CPU サイクルは大きくなります。レベル 6 は「yes」に相当します。レベル 0 は「no」に相当します。 既定値は「yes」(6) です。

DenyGroups

指定したユーザグループのログインを拒否するには、このキーワードを使用します。正規表現に対応しています。 詳細については、「グループアクセスの構成 『88ページ 』」を参照してください。 このキーワードが構成されていない場合、すべてのグループのログインが許可されます。

DenyHosts

指定したクライアントホストのログインを拒否するには、このキーワードを使用します。正規表現に対応しています。 詳細については、「クライアントホストアクセスの構成 『88ページ 』」を参照してください。 このキーワードが使用されていない場合、すべてのクライアントホストが許可されます。

注意:

ホスト式をドメイン名 (IP アドレスではなく) を使用して構成している場合、この設定に加えて、**ResolveClientHostName** を「yes」に設定する必要があります。また、ドメイン名が解決できなかったホストからのアクセスを防ぐには、**RequireReverseMapping** を「yes」に設定する必要があります。**ResolveClientHostName** が「yes」の時、解決された名前は完全修飾ドメイン名です。つまり、**RequireReverseMapping** が「yes」の時は、IP アドレスからの接続が正しく処理されるように、ホスト名に完全修飾ドメイン名を指定するか、または正規表現を使用する必要があります。

許可リストまたは拒否リストにアドレスを構成するには、IPv4 と IPv6 の両方のアドレスを指定する必要があります。拒否リストによってアクセスがブロックされるように構成する場合は、このことは特に重要です。 許可リストまたは拒否リストにローカルホストを構成するには、すべての外部インタフェースの IP アドレスと、ローカルループバックアドレス (127.0.0.1 および 0: 0: 0: 0: 0: 0: 0: 1) を記述します。

DenyTCPForwardingForGroups

指定したユーザグループのポート転送を拒否するには、このキーワードを使用します。 正規表現に対応しています。 詳細については、「グループアクセスの構成 『88ページ 』」を参照してください。

DenyTCPForwardingForUsers

指定したユーザに対してポート転送を拒否するには、このキーワードを使用します。 正規表現に対応しています。 詳細については、「ユーザアクセスの構成 『87ページ 』」を参照してください。

DenyUsers

指定したユーザのログインを拒否するには、このキーワードを使用します。 正規表現に対応しています。 詳細については、「ユーザアクセスの構成 『87ページ 』」を参照してください。 このキーワードが構成されていない場合、すべてのユーザのログインが許可されます。

FipsMode

すべての接続で、FIPS 140-2 仕様に準拠したセキュリティプロトコルおよびアルゴリズムを使用するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

ForceSftpFilePermissions

サーバにアップロードするすべてのファイルに指定したファイル権限を設定します (**sftp** または **scp** を使用します)。他のすべての権限設定の動作を上書きします。3 桁の許可モード値を使用します。例えば、 **ForceSftpFilePermissions** を 600 に設定すると、アップロードするすべてのファイルが 600 に設定されます (`-rw-------`)。
また、既存ファイルに対する権限を変更しようとすると、クライアントユーザが要求する権限値に関係なく、そのファイルも 600 に設定されます。この設定は、ディレクトリへの権限には影響しません。

ここで **ForceSftpFilePermissions** が構成されている場合は、次のようになります。

ファイルを新規作成するか既存ファイルに上書きするかにかかわらず、アップロードするファイルはすべて指定値に設定されます。

システムの UMASK 設定は無視されます。

すべての **chmod** コマンド (**sftp** ユーザが実行) では、ユーザが指定する値は無視されます。また、ファイルの権限の値は **ForceSftpFilePermissions**で設定された値に変更されます。

その **-p** オプションを **sftp** または **scp** コマンドラインで使用すると無視されます。

ForwardACL

クライアントのポート転送へのアクセスを詳細レベルで制御するには、このキーワードを使用します。 正規表現に対応しています。 構文は次のとおりです。

`ForwardACL allow|deny local|remote` *`user_ex forward_ex`* `[`*`origin_ex`*`]`

`user_ex` は、どのユーザがポート転送へのアクセスを許可または拒否されるかを決定する正規表現です。 詳細については、「ユーザアクセスの構成 『87ページ 』」を参照してください。"

`forward_ex` は *`host%port`*の形式の正規表現です。その意味はローカル転送とリモート転送のどちらに制限を構成しているかによって異なります。ローカル転送の制御を構成している場合は、対象のホストとポートを指定します。リモート転送の制御を構成している場合、ホストはサーバコンピュータであり、ポートは、そのサーバがクライアントに転送するポートです。

`origin_ex` は、IP アドレスを識別する正規表現です。その意味はローカル転送とリモート転送のどちらに制限を構成しているかによって異なります。ローカル転送の制御を構成している場合、転送要求を行うクライアントマシンをを指定します。リモート転送の制御を構成している場合、サーバ上の転送されるポートに接続するコンピュータを指定します。

GatewayPorts

リモートホストがクライアントに転送されるポートに接続できるかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

HostCertificateFile

サーバ認証に X.509 証明書を使用するよう指定します。関連する秘密鍵を、**HostKeyFile**を使用して指定します。

HostKeyFile

サーバの認証に使用する秘密鍵のファイル名と格納場所を指定します。
既定値は `/etc/ssh2/hostkey`です。

HostSpecificConfig

ホスト固有のサブ構成ファイルを指定します。構文は次のとおりです。

```
HostSpecificConfig host_expression subconfig_file
```

ホスト式がクライアントホストに一致する場合、サーバは指定されたサブ構成ファイルを使用します。

ホスト式をドメイン名 (IP アドレスではなく) を使用して構成している場合、この設定に加えて、**ResolveClientHostName** を「yes」に設定する必要があります。

HPNDisabled

パフォーマンスの向上のために Reflection for Secure IT で HPN ダイナミック TCP ウィンドウ機能を使用するかどうかを指定します。HPNDisabled = 「no」 (既定) と設定すると、最適なパフォーマンスが得られるように、Reflection for Secure IT によって TCP ウィンドウと TCP 受信バッファが調節されます。**HPNDisabled** を「yes」にすると、受信バッファは 64KB に設定されます。

IdleTimeout

接続がここで指定した時間非アクティブであり続けると、サーバは接続を終了します。時間を秒単位で設定するには、数字の後に s を使用するか、何も使用しません。また、時間を分 (m)、時 (h)、日 (d)、または週 (w) 単位で指定することもできます。制限しない場合は、ゼロ (0) を使用します。 既定は 0 です。

IgnoreRlogin

このキーワードは AIX システムにのみ適用されます。「rlogin」属性 `/etc/security/user` を無視するか適用するかを指定します。許可される値は「yes」および「no」です。既定値は「no」で、サーバにより現在の「rlogin」値が適用されます。

注意:

「login」属性 `/etc/security/user` は、Secure Shell クライアントによるリモートログインには影響しません。次の値に関係なく、常に **IgnoreRlogin**です。

AIX システムでは、 **IgnoreRlogin** は **AccountManagement** が「none」であると無視されます。

KeepAlive

システムが接続の相手側に TCP キープアライブメッセージを送信するかどうかを指定します。サーバは、メッセージの送信間隔にシステム全体に設定された値を使用します。 許可される値は「yes」および「no」です。既定は「yes」です。 注意: **ClientAliveCountMax** および **ClientAliveInterval** は、SSH 接続に影響し、メッセージは SSH トンネルを介して送信されます。その **KeepAlive** 設定は TCP 接続に影響し、TCP メッセージは安全なトンネルで送信されないため、スプーフィングに対してより脆弱です。

KEXs

サーバが対応している鍵交換アルゴリズムを指定します。 対応している値は、「diffie-hellman-group-exchange-sha256」、「diffie-hellman-group-exchange-sha1」、「diffie-hellman-group14-sha1」、「diffie-hellman-group1-sha1」です。カンマ区切りのリストとして、複数のアルゴリズムを指定することができます。既定値は「diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1,diffie-hellman-group1-sha1」です。

LibGssKrb5

GSSAPI (Kerberos 5) 認証を使用する場合は、この設定を使用します。この設定は、libgssapi_krb5.so という名前の Kerberos ライブラリへの完全修飾パスを指定します。

LibKrb5

GSSAPI (Kerberos 5) 認証を使用する場合は、この設定を使用します。この設定は、`libkrb5.so` という名前の Kerberos ライブラリへの完全修飾パスを指定します。

注意: サーバには、次のライブラリが必要です。 `libkrb5.so` (HP-UX PARISC では `.sl`)。この名前のライブラリが存在しない場合、実際のライブラリを指す `libkrb5.so` という名前のシンボリックリンクを作成する必要があります。

LibWrap

このキーワードは TCP ラッパのダイナミックな対応を提供します。TCP ラッパの対応を有効にするには、libwrap 共有ライブラリへの完全修飾パス (`LibWrap=/usr/lib/libwrap.so` など) を指定します。libwrap ファイルは、スタティックなライブラリではなく、共有ライブラリでなければなりません。既定では、このキーワードは空で、TCP ラッパ機能は無効になっています。

注意: このキーワードを使用する前に、指定したファイルが有効な libwrap ライブラリであることを確認する必要があります。このことは、許可したユーザのみが接続できるようにする上で重要です。指定したファイルが存在しない場合、 **sshd** サーバは起動しません。ただし、このファイルが存在する場合、 **sshd** は起動しますが、ファイルが有効なライブラリであるかどうかは確認しません。接続のたびに、 **sshd** プロセスにより指定のファイルのロードが試行され、ファイルが有効なライブラリでなければ、サーバによりエラーメッセージが記録され、ユーザの接続が許可されます。

ListenAddress

インタフェースのアドレスを指定します。 **sshd** サーバのソケットをバインドさせます。IPv4 または IPv6 形式で指定するか、あるいは「any」 (既定) を使用することができます。値「any」を指定すると、使用可能な任意の IPv4 または IPv6 アドレスに対してリッスンするようにサーバが構成されます (「`[::],0.0.0.0`」と同様)。IPv4 アドレスのみを指定する場合、クライアントは IPv4 アドレスを使用して接続する必要があります。IPv6 形式のみを指定する場合、大部分のオペレーティングシステムは IPv4 クライアントの接続にも対応していますが、これは、Secure Shell サーバではなく、オペレーティングシステムによって制御されています。オプションで、コロンまたはスペースに続けてポート番号を追加することにより、アドレスにポートを組み込むことができます。このポート値は、 **Port** キーワードの設定より優先されます。IPv6 アドレスを指定している場合、アドレスを角括弧で囲む必要があります。たとえば、以下のようになります。

IPv4 の構文: `ListenAddress=209.85.171.99:6666`

IPv6 の構文: `ListenAddress=[::D155:AB63]:6666`

ListenAddress は、 **AddressFamily** の設定の影響を受けます。設定を **AddressFamily**=inet にした場合、**ListenAddress** の値「any」は「`0.0.0.0`」と同じです。設定を **AddressFamily**=inet6 にした場合、ListenAddress の値「any」は「`[::]`」と同じです。**AddressFamily** の設定を「inet」または「inet6」にし、**ListenAddress** に別のファミリのアドレスを指定した場合、構成ファイルエラーにより、**sshd** の起動に失敗します。**ListenAddress** に IP アドレスではなくホスト名を指定した場合、**AddressFamily** の制約により、ホスト名を該当するファミリのアドレスと関連付ける必要があり、サーバはそのアドレスにバインドされます。

注意: このキーワードで設定される値は累積されます。このキーワードを 1 つまたは複数の構成ファイルで複数回構成することで、複数の値を設定できます。

LogCertificateSubject

認証に使用した証明書のシリアル番号と Subject (サブジェクト)をシステムログに記録するかどうかを指定します。正常終了と失敗の両方のメッセージが記録されます。許可される値は「yes」および「no」です。既定は「yes」です。

LoginGraceTime

クライアント認証に許可される秒数を設定します。クライアントが指定された秒数内にユーザを認証できないと、サーバは接続を切断して終了します。 制限しない場合は、ゼロ (0) を使用します。 既定値は 120 です。

LogLevel

syslog にログ記録される **sshd** メッセージに使用される冗長レベルを設定します。使用可能な値は、「fatal」、「error」、「quiet」、「info」、「verbose」、「debug1」 (「debug」は 1 に相当)、「debug2」 (2 に相当)、「debug3」 (3 に相当)、および「trace」 (「debug99」は 99 に相当) です。これらの値と関連する syslog レベルは、fatal は CRIT、error および quiet は ERROR、info および verbose は INFO、debug1、debug2、debug3、および trace は DEBUG です。 既定値は「error」です。

注意: 記録の設定を「trace」にすると、セキュリティのリスクが高まります。このレベルでは、暗号化されていないプロトコル情報が書き出されることがあるため、情報の漏洩が懸念されます。また、大量に書き込まれる情報によりディスクスペースが急速に占有され、ホストや Reflection for Secure IT の応答が停止する可能性があります。

LogPublicKeyFingerPrint

認証に使用した公開鍵の指紋をシステムログに記録するかどうかを指定します。正常終了と失敗の両方のメッセージが記録されます。 許可される値は「yes」および「no」です。既定は「yes」です。

MACs

サーバがデータの整合性検証に許可する MAC (ハッシュされたメッセージ認証コード) を優先順に指定します。 許可される値は、「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」、および「hmac-ripemd160」です。これらすべてに対応するには「AnyMac」を使用します。「AnyStdMac」を使用して「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」を指定します。hmac-sha256 を指定すると hmac-sha2-256 も有効になります。hmac-sha512 を指定すると hmac-sha2-512 も有効になります。複数の MAC をカンマ区切り一覧として指定することもできます。その他には、「none」、「any」 (AnyMac + 「none」と同等)、および「AnyStd」 (「AnyStdMac」 + 「none」と同等) オプションがあります。MAC について「none」で合意すると、メッセージ認証コードは使用されません。この場合データの整合性は保護されないため、「none」を含むオプションはお勧めしません。 既定値は「AnyStdMac」です。

MaxConnections

許可される最大クライアント接続数を設定します。 制限しない場合は、ゼロ (0) を使用します。 既定値は 50 です。

MaxSessions

TCP のシングル接続に対応する多重セッションの最大数を指定します (Reflection for Secure IT クライアントでキーワード **ConnectionReuse** を使用すると、多重化を有効にできます)。値の範囲は、0 ～ 10 です。既定値は 10 です。この値を 1 に設定すると、接続を再使用できなくなります。この値を 0 に設定すると、すべての接続が無効になります。

MaxStartups

認証されなかった同時接続試行の最大許容数を指定します。この制限値に達すると、認証に成功するか、または接続試行用の **LoginGraceTime** 制限に達するまで、その後の接続は切断されます。既定値は 10 です。

PamServiceName

認証およびセッションに使用される PAM (Pluggable Authentication Modules) サービスの名前を指定します。既定値は「ssh」です。

PamServiceNameForInternalProcesses

内部プロセスに使用するオプションの PAM サービスの名前を指定します。指定したサービスを使用して、追加のアカウントおよびセッション管理を行うことができます。たとえば、以下のようになります。

```
PamServiceNameForInternalProcesses ssh-shell
```

この場合、すべてのユーザが **PamServiceName** で指定されたサービス (既定値は「ssh」) を利用します。shell および exec ユーザは、「ssh-shell」サービスも利用します。

注意: 指定された PAM サービスは、常に PAM アカウントおよびセッション管理に対応し、特定のプラットフォーム (Linux および AIX が該当、Solaris は該当しない) では認証管理にも対応します。プラットフォームによって認証管理が使用できる場合と使用できない場合があるため、システムへのアクセスがアカウントおよびセッション管理を使用して構成できるように、必ず設定を pam_permit.so にする必要があります。

PamServiceNameForSubsystems

サブシステムで使用するオプションの PAM サービスの名前を指定します。指定したサービスを使用して、追加のアカウントおよびセッション管理を行うことができます。構文は次のとおりです。

```
PamServiceNameForSubsystems subsystem PAMservicename
```

例えば、以下を使用して、SFTP 接続用の追加のアカウントおよびセッション管理を行うことができます。

```
PAMServiceNameforSubsystems sftp ssh-sftp
```

この場合、すべてのユーザが PamServiceName で指定されたサービス (既定値は「ssh」) を利用します。SFTP ユーザは、「ssh-sftp」サービスも利用します。

注意: 指定された PAM サービスは、常に PAM アカウントおよびセッション管理に対応し、特定のプラットフォーム (Linux および AIX が該当、Solaris は該当しない) では認証管理にも対応します。プラットフォームによって認証管理が使用できる場合と使用できない場合があるため、システムへのアクセスがアカウントおよびセッション管理を使用して構成できるように、必ず設定を pam_permit.so にする必要があります。

PasswordGuesses

ユーザが許可されるパスワード認証の最大試行回数を設定します。既定値は 3 です。

PermitEmptyPasswords

パスワードが空 (NULL) のユーザによるパスワード認証をサーバが許可するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

PermitRootLogin

クライアントのユーザが root 権限でログインできるかどうかを指定します。指定可能な値は「yes」、「no」、および「without-password」です。「without-password」を指定すると、「公開鍵」または「GSSAPI」の認証方法によってユーザ認証を行っている場合のみ、ユーザが root 権限でログインすることができます。既定値は「yes」で、すべての認証方法に対して root ログインを許可します。

PidFile

次のプロセス ID を含むファイルを指定します (**sshd** デーモン)。完全修飾パスを使用します。ファイル名に文字列 %s が含まれる場合、その文字列はサーバのポート番号に置き換えられます。

PkidAddress

PKI Services Manager への接続に使用するポートを指定します。*host*: *port* という形式を使用します。既定は `localhost:18081` です。ホストを指定してポートを省略した場合、既定の PKI Services Manager ポート (18081) が使用されます。

PkidPublicKey

Reflection PKI Services Manager の ID を確認するために使用される公開鍵の名前と場所を指定します。既定は `/opt/attachmate/pkid/config/pki_key.pub` です。

Port

サーバがリッスンするポートを指定します。既定は 22 で、これは Secure Shell 接続の標準ポートです。

PrintLastLog

ユーザが対話型でログインする際、Reflection for Secure IT サーバで最終ログイン日時が表示されるかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

PrintMotd

ユーザが端末セッションにログインした時に、サーバでファイル `/etc/motd` の message-of-the-day テキストを表示するかどうかを指定します(この設定は、`/etc/issue`の表示より優先されることはありません)。 許可される値は「yes」および「no」です。既定は「yes」です。

ProtocolVersionString

初期接続プロトコル中にサーバからクライアントに送信する文字列のソフトウェアバージョン部分を指定します(文字列の最初の部分は常に「SSH-2.0-」となり、これはサーバが対応している SSH のバージョンを示します。これはプロトコル RFC で必要となる部分で、編集はできません)。文字列にスペースが含まれる場合は二重引用符を使用します。ここで **ProtocolVersionString** が空の文字列 (既定) の場合、文字列のソフトウェアバージョン部分は自動的に生成され、サーバのバージョンとビルド番号が書き込まれます。この番号は、サーバのソフトウェアをアップグレードすると自動的に更新されます。

注意: 多くのクライアントで、サーバの種類の識別と互換機能の有効化のためにプロトコル文字列を使用しています。既定値を変更すると、公開鍵の認証に失敗することがあり、またサーバごとに異なる機能に影響を及ぼす可能性があります。

QuietMode

このキーワードはお勧めしません。 **LogLevel**を使用します。

RadiusFile

RADIUS 認証を構成するために使用するファイルの名前を指定します。絶対パスを指定した場合以外は、ファイルの相対パスは `/etc/ssh2` であることを前提とします。ファイルの構文については、「ファイル」セクションの `/etc/ssh2/radius_config` を参照してください。既定値はなく、このキーワードは値をなしにすることもできます。

RekeyIntervalSeconds

サーバが新しい鍵交換を開始するまでの間隔 (秒単位) を指定します。この値が低すぎると、クライアントとサーバ間の通信ができなくなることがあります。この問題を回避するには、間隔を 200 秒未満に設定しないことをお勧めします。サーバが開始する再入力要求をオフにするには 0 (ゼロ) を使用します。0 を使用しても、クライアントは再入力を要求できます。 既定値は 3600 です。

RequiredAuthentications

1 つ以上のクライアント認証方法を要求するには、このキーワードを使用します。ユーザが認証済みと見なされるためには、指定したすべての認証方法が成功する必要があります。 対応している認証方法は、「gssapi-keyex」、「gssapi-with-mic」、「publickey」、「keyboard-interactive」、および「password」です。

注意: **RequiredAuthentications** は **AllowedAuthentications** を上書きします。

RequireReverseMapping

クライアントホストからの接続が許可されるかどうかを確認する時に、DNS ルックアップが成功しなければならないかどうかを指定します。この機能を有効にするためには、さらに **ResolveClientHostName** を「yes」に設定する必要があります。 許可される値は「yes」および「no」です。既定は「no」です。

ResolveClientHostname

サーバがクライアントの IP アドレスをドメイン名に解決しようとするかどうかを指定します。このキーワードを「yes」に設定すると、接続時間が遅くなることがありますが、いずれかのキーワードでホスト名の一致基準を IP アドレスではなくドメイン名に構成している場合、このキーワードは必須です(次を参照してください。**AllowHosts**、 **DenyHosts**、 **UserSpecificConfig**、および **HostSpecificConfig**)。また、このキーワードを「yes」に設定すると、IP アドレスではなく DNS 名がログに表示されます。 許可される値は「yes」および「no」です。既定は「yes」です。

注意: **ResolveClientHostname** が「yes」の時、解決された名前は常に完全修飾ドメイン名です。つまり、ホスト名を指定するキーワードでは、ホスト名が正しく処理されるように完全修飾ドメイン名を使用するか、または正規表現を使用する必要があります。

SessionRestricted

サーバが許可するセッションの種類を指定します。設定できる値は、「shell」 (端末シェルセッションを許可)、「exec」 (クライアントがサーバ上でコマンドを実行することを許可)、および「subsystem」 (Reflection for Secure IT クライアントからの **sftp** および **scp** 転送に対応するために必要) です。既定値は、「shell, exec, subsystem」です。

注意: OpenSSH タイプのクライアントの場合、 **sftp** 転送では「subsystem」、 **scp** 転送では「exec」セッションタイプが必要です。

SettableEnvironmentVars

クライアントで構成可能な環境変数を指定します。この値により、クライアントおよびユーザ固有の環境ファイル (`~/.ssh2/environment`) でのクライアント **SetRemoteEnv** キーワードの範囲が制限されます(注意: この設定は、`/etc/environment` や `/etc/ssh2/environment` root でのみ制御可能なサーバのファイルで構成される変数には影響しません)。引数は大文字で指定する必要があります。このキーワードは、既定の構成ファイルで有効になっており、次の値に設定されています。「'LANG, LC_ALL, LC_COLLATE, LC_CTYPE,LC_MONETARY, LC_NUMERIC, LC_TIME, PATH, TERM, TZ, UMASK」

SftpLogCategory

次で指定した機能に送信する sftp サーバメッセージの種類を決定します (**SftpSysLogFacility**)。カンマ区切りのリストを使用します。既定値は「loginlogout,directorylistings,downloads,modifications,uploads」です。これですべての種類の記録が構成されます。これらのオプションか、「all」または「none」を指定することができます。

SftpSysLogFacility

sftp-server サブシステムからのメッセージをログ記録するのに使用されるファシリティコードを指定します。既定ではこの値は空です。この値が空で **LogLevel** が空でない場合、AUTH 機能によって記録が行われます。ここで **SftpSysLogFacility** と **LogLevel** が両方とも空の場合、サーバは syslog への記録を行いません。この値が「none」の場合、Reflection for Secure IT は (**LogLevel** の設定に関係なく) syslog への記録を無効にします。その他の有効な値はプラットフォームに依存します。syslog(3) を参照してください。有効な値はプラットフォームに依存します。syslog(3) を参照してください。この設定を「auth」にすると、ログメッセージは次の既定と同じ機能に記録されます (**sshd**)。

SftpVersion

サーバが対応するSFTP プロトコルの最大バージョンを指定します。有効な値は 3 および 4 (既定値) です。クライアントが、この設定で指定するバージョンより古いバージョンにしか対応していない場合、クライアントが指定するバージョンが使用されます。

注意: このキーワードは、 **Subsystem-sftp** で既定の内部の sftp-server (Subsystem-sftp internal://sftp-server) の使用が構成されている場合にのみ、接続に影響します。外部の sftp-server を構成した場合は、-v 3 または -v 4 を使用して SFTP バージョンを指定します。例えば、`subsystem-sftp /usr/libexec/sftp-server -v 3` のように指定します。

SmartFileTransfer

データの転送前にサーバでファイルの同一性を確認するかどうかを指定します。このキーワードを「yes」 (既定) にすると、スマートファイルコピー (同じファイルは転送しないようにする) とチェックポイント再開 (中断していたファイルの転送を中断の時点から再開する) が有効になります。このキーワードを「no」にすると、Reflection for Secure IT は常にすべてのファイルのすべての内容を転送します。
注意: クライアントで **SmartFileCopy**を使用することにより、スマートファイルコピーを無効にできます。クライアントで **CheckpointResume**を使用することにより、チェックポイント再開を無効にできます。

StrictModes

公開鍵認証のためのディレクトリのパーミッションを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。 設定を「yes」にする場合、ユーザのディレクトリ (`~/.ssh2`) とすべての親ディレクトリの書き込みと実行がそのユーザによってのみ可能になるようにする必要があります (モード 744 が設定可能)。ユーザのディレクトリの推奨のパーミッションは 700 です。これらの条件が満たされない場合、公開鍵認証は失敗します。設定を「no」にする場合、このファイルパーミッションが必須でないため、ファイルや情報の機密保持が保証されない可能性があります。

注意: 各ユーザの認証ファイル (`~/.ssh2/authorization`) には、現在の **StrictModes** の設定に関係なく、追加のファイルパーミッション要件が必要です。このファイルは、グループおよびグループ外の書き込みアクセスが防止されるよう構成する必要があります (600 を推奨、644 も設定可能)。認証ファイルの制限が十分でない場合、公開鍵の認証は常に失敗します。

Subsystem

クライアントにエクスポートするサブシステムを指定します。引数は、クライアントがサブシステムを要求した時に実行されるコマンドを指定します。キーワードに続く区切り文字には、ダッシュ、等号、またはスペースを使用できます。

ここで **sftp** および **scp** 転送に対応するには、sftp-server サブシステムを指定する必要があります。以下の既定の構成は、sftp サービスを子プロセスで内部的に実行します。

```
Subsystem-sftp internal://sftp-server
```

SyslogFacility

サーバからのメッセージを記録するのに使用されるファシリティコードを指定します。既定値は「AUTH」です。この値が「none」の場合、Reflection for Secure IT は syslog への記録を無効にします。その他の有効な値はプラットフォームに依存します。syslog(3) を参照してください。

注意: この値を「none」に設定すると、接続の試行やユーザのログインの監査ログが作成されないため、この設定は推奨できません。サービス妨害攻撃が発生した場合、過剰に接続を行っている一連の IP アドレスを特定するのに監査ログが役立ちます。また、監査ログは、ユーザがシステムにアクセスしていないと虚偽の主張をした場合の重要な証拠となります (否認防止)。

注意: このログへの書き込みのために指定するデバッグレベルは、セキュリティの面で悪影響を及ぼすことがあります。詳細は、**LogLevel** を参照してください。

TrustAnchor

このキーワードへの対応は終了しています。トラストアンカを構成するには、Reflection PKI Services Manager を使用してください。

UseLogin

対話型ログインセッションに login(1) を使用するかどうかを指定します。許可される値は「yes」および「no」です。既定は「no」です。

注意:

リモートコマンド実行には login(1) は使用されません。

この設定を有効にすると次が無効になります。 **X11Forwarding** 。login(1) では xauth(1) cokkie の処理方法を認識していないためです。

login(1) を使用すると、権限の分離が無効になります。既定では、 **sshd** により、認証されたユーザの権限を持つ新しいプロセスが作成されます。これは、権限のないプロセス内の変造を抑制することにより、権限の昇格を防止するために行われます。ここで **UseLogin** を有効にすると、この機能が無効になります。

UsePAM

この設定は、サーバで PAM を使用するよう構成するための代替方法です。許可される値は「yes」および「no」です。ここで **UsePam** が構成されていない場合、サーバでは現在の値に **AuthKbdInt.Required**、 **AccountManagement**、および **UsePamSessions**を使用します。このキーワードの設定を「yes」にした場合は、`AuthKbdInt.Required=pam`、`AccountManagement=pam`、および `UsePamSessions=yes` と設定したのと同じです。このキーワードの設定を「no」にした場合は、`AuthKbdInt.Required=password`、`AccountManagement=password`、および `UsePamSessions=no` と設定したのと同じです。

注意: ここで **UsePAM**を変更する場合、関連するキーワードを構成ファイルの **UsePAM** の後に設定しないようにしてください。ここで **AuthKbdInt.Required**、**AccountManagement**、または **UsePamSessions** を **UsePAM**の後で競合する値に設定すると、サーバで最後に読み込まれた値が使用されるため、 **UsePAM** によって構成された値よりもこの値が優先されます。

UsePAMAcctMgmt

このキーワードはお勧めしません。このキーワードの設定を「yes」にした場合は、`AccountManagement=pam` と設定したのと同じです。

UsePamSessions

セッション管理に PAM を使用するかどうかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

UserConfigDirectory

ユーザ固有の情報に使用されるディレクトリを指定します。このディレクトリには、(鍵の認証に必要な) 認証ファイルのほか、「ファイル」セクションに列記されているユーザ固有ファイルが格納されます。次のマクロが認識されます。%U = ユーザのログイン名、%D = ユーザのホームディレクトリ、%IU = ユーザのユーザ ID、%IG = ユーザのグループ ID。既定値は「`%D/.ssh2`」です。

UserSpecificConfig

ユーザ固有の構成ファイルを指定します。構文は次のとおりです。

```
UserSpecificConfig user_expression subconfig_file
```

ユーザ式が接続を試みているユーザに一致する場合、サーバは指定されたサブ構成ファイルを使用します。

注意: この式のホスト部分の一致基準をホストのドメイン名 (IP アドレスではなく) に構成している場合、この設定に加えて、 **ResolveClientHostName** を「yes」に設定する必要があります。

VerboseMode

このキーワードはお勧めしません。 **LogLevel**を使用します。

X11DisplayOffset

サーバによる X11 転送で使用できる最初のディスプレイ番号を設定します。既定値は 10 です。

X11UseLocalHost

サーバが、X11 転送をループバックアドレスとワイルドカードアドレスのどちらにバインドすべきかを指定します。 許可される値は「yes」および「no」です。既定は「yes」です。

XAuthPath

xauth(1) プログラムの場所を指定します。既定値 (例えば `/usr/X11R6/bin/xauth`) はシステムによって異なります。

付 録 E

ファイルおよびディレクトリの権限

認証をセキュアに実装し、不正変更、情報の漏洩、スプーフィングを防止するために、クライアントとサーバで使用されるファイルおよびディレクトリにパーミッションと所有者を正しく構成します。この基準に満たない場合は、Secure Shell 接続と公開鍵認証が失敗します。

注意:

- **StrictModes** の設定は、サーバもクライアントも既定値は有効 (=yes) で、十分な安全レベルを確保するのに役立ちます。
- ファイルの所有者は、`root` か、関連ファイルが存在するホームディレクトリの所有者（`owner`）でなければなりません。
- パーミッションの要件を満たすには、パーミッションを以下の表に示す制限レベル以上（括弧内の 8 進数の値以下）にする必要があります。
- 括弧内に示すファイルとディレクトリは既定値です。

クライアント側のファイルとディレクトリ

ファイル/ディレクトリ	最大限のセキュリティ	StrictModes = no の場合	StrictModes = yes の場合
Secure Shell ディレクトリ (`~/.ssh2/`)	700	条件なし	ユーザ専用の書き込み権限 (755)
ユーザのホームディレクトリとすべての親ディレクトリ	744 755	条件なし	ユーザ専用の書き込み権限 (755)
ユーザの秘密鍵	600	ユーザ専用の読み/書き権限 (600)	ユーザ専用の読み/書き権限 (600)
ユーザの公開鍵	600	条件なし	条件なし
ユーザの ID ファイル (`~/.ssh2/identification`)	600	条件なし	ユーザ専用の書き込み権限 (644)
ユーザのホスト鍵のディレクトリ (`~/.ssh2/hostkeys`)	700	条件なし	条件なし
ホストの公開鍵ファイル	600	条件なし	条件なし
ユーザの構成ファイル (`~/.ssh2/ssh2_config`)	600	条件なし	ユーザ専用の書き込み権限 (644)
クライアント PKI Services Manager の公開鍵 (**PkidPublicKey** を使用して指定)	600	条件なし	条件なし
グローバル構成ディレクトリ (`/etc/ssh2/`)	755	条件なし	条件なし
グローバルホスト鍵ディレクトリ (`/etc/ssh2/hostkeys`)	755	条件なし	条件なし

ファイル/ ディレクトリ	最大限の セキュリティ	StrictModes = no の場合	StrictModes = yes の場合
グローバルホストの公開鍵ファイル	644	条件なし	条件なし
グローバルユーザの構成ファイル (`/etc/ssh2/ssh2_config`)	644	条件なし	条件なし
サーバのログディレクトリ ()	711	711	711
サーバの監査ログファイル	600		

サーバ側のファイルとディレクトリ (ユーザ固有)

ファイル/ ディレクトリ	最大限の セキュリティ	StrictModes = no の場合	StrictModes = yes の場合
Secure Shell ディレクトリ (`~/.ssh2/`)	700	条件なし	ユーザ専用の書き込み権限 (755)
ユーザのホームディレクトリとすべての親ディレクトリ	744 755	条件なし	ユーザ専用の書き込み権限 (755)
サーバ上のユーザの認証ファイル (`~/.ssh2/authorization`)	600	ユーザ専用の書き込み権限 (644)	ユーザ専用の書き込み権限 (644)
サーバ上のユーザの Secure Shell 環境ファイル (`~/.ssh2/environment`)	600	条件なし	条件なし
ユーザのログイン動作ファイル (`~/.hushlogin`)	600	条件なし	条件なし

サーバ側のファイルとディレクトリ (グローバル)

ファイル/ ディレクトリ	最大限の セキュリティ	StrictModes = no の場合	StrictModes = yes の場合
サーバの構成ディレクトリ (`/etc/ssh2`)	644	条件なし	条件なし
サーバの秘密鍵ファイル (`/etc/ssh2/hostkey`)	600	root 専用の読み/書き権限 [600]	root 専用の読み/書き権限 [600]
サーバの公開鍵ファイル (`/etc/ssh2/hostkey.pub`	600	条件なし	条件なし
サーバの RADIUS 認証構成ファイル (`/etc/ssh2/radius_config`)	600	条件なし	条件なし
サブ構成ファイルのディレクトリ (`/etc/ssh2/subconfig`)	700	条件なし	条件なし
サブ構成ファイル	600	条件なし	条件なし

ファイル/ディレクトリ	最大限のセキュリティ	StrictModes = no の場合	StrictModes = yes の場合
グローバル Secure Shell 環境ファイル (`/etc/ssh2/environment`)	600	条件なし	条件なし
クライアント PKI Services Manager の公開鍵 (**PkidPublicKey** を使用して指定)	600	条件なし	条件なし
サーバのログディレクトリ (`/etc/ssh2/logs`)	711	条件なし	条件なし
サーバの監査ログファイル (`/etc/ssh2/logs/sshd2-audit-*`)	600	条件なし	条件なし

付 録 F

ssh コマンドラインオプション

ssh の構文は次のとおりです。

ssh [**-4**] [**-6**] [**-a**] [**-c** *cipher*] [**-C**] [**-d** *debug_level*] [**-e** *character*]
[**-f**] [**-fo**] [**-F** *file*] [**-g**] [**-h**] [**-i** *file*] [**-l** *username*]
[**-L** [[*protocol/*]*listening_port*:*host*:*hostport*] [**-m** *mac_algorithm*] [**-n**]
[**-o** *option*] [**-p** *port*] [**-q**] [**-R** [*protocol/*]*listening_port*:*host*:*hostport*]
[**-s** *subsystem*] [**-S**] [**-t**] [**-v**] [**-V**] [**-W**] [**-x**] [**-X**] [**-Y**]
[[*username@*]*host*[*#port*]] [*remote_command* [*arguments*] ...]

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

注意: コマンドラインで指定したすべてのオプション (ユーザ名、ホスト名、およびその他の機密情報) は、プロセス状態 (**ps**) 一覧に表示されます。機密のオプションやスイッチを指定する場合は、他のユーザが容易にその情報を参照できないように注意する必要があります。安全な方法として、これらのオプションを構成ファイル内で設定して、推奨のファイルのパーミッション (構成ファイル = 600、ファイルの格納ディレクトリ = 700) で構成ファイルを保護する方法が考えられます。

-4

接続に IPv4 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-6

接続に IPv6 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-a

認証エージェントの転送を無効にします。 認証エージェントの転送を有効にするには、 **ForwardAgent** キーワードを使用します。このキーワードの既定値は「yes」です。次を使用して (**-a**)、構成ファイルの設定を上書きできます。

-c *暗号*

対応する 1 つ以上の (カンマで区切った) 暗号化アルゴリズムを指定します。 特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。 許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。

この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。

あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと 128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。

また、構成ファイルで次を使用して、暗号化アルゴリズムを構成できます (**Ciphers** キーワード)。既定値は「AnyStdCipher」です。

-C

圧縮を無効にします。 圧縮は、モデム回線などの低速接続には向いていますが、高速ネットワークでは応答速度を低下させます。また、圧縮はパケットをより不規則にするため、悪意のある人物がパケットを解読することが難しくなります。 構成ファイルで次を使用して圧縮を有効にすることができます (**Compression** キーワード (構成ファイル内))。次を使用して (**-C**)、構成ファイルの設定を上書きできます。

-d *debug_level*

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。 1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。

注意: 記録の設定を 99 にすると、セキュリティのリスクが高まります。このレベルでは、暗号化されていないプロトコル情報が書き出されることがあるため、情報の漏洩が懸念されます。また、大量に書き込まれる情報によりディスクスペースが急速に占有され、ホストや Reflection for Secure IT の応答が停止する可能性があります。

-e *character*

端末セッションのエスケープ文字を設定します。既定の文字はチルダ (~) です。エスケープ文字を「none」に設定すると、使用できるエスケープ文字はなくなり、チルダは他の文字と同様に機能します。 詳細については、以下の「エスケープシーケンス」を参照してください。また、構成ファイルで次を使用してエスケープ文字を設定することもできます (**EscapeChar** キーワード)。

-f

ポート転送が設定されていて Scure Shell セッションをバックグラウンドで実行したい場合に、このオプションを使用します。少なくとも 1 つのポート転送ルールが構成されている場合、このオプションにより、認証の完了後に Secure Shell セッションがバックグラウンドに送信されます。プロセスを手動で終了するまで、このセッションはバックグラウンドに留まり、無限に転送要求を受け付けます。(これは、構成ファイル内で `GoBackground=yes` と設定するのと同じです。)

-fo

このオプションは **-f**と同様に機能しますが、この場合、バックグラウンドセッションは転送された 1 つだけの接続の開始を待機し、転送された接続が閉じられるとすぐに終了します。(これは、構成ファイル内で `GoBackground=oneshot` と設定するのと同じです。)

-F *file*

追加の構成ファイルを指定します。既定のユーザ固有のファイル (`~/.ssh2/ssh2_config`) およびシステム全体にわたるファイル (`/etc/ssh2/ssh2_config`) に加えて、このファイルからも設定が読み取られます。このファイルの設定は、ユーザ固有のファイルおよびシステム全体にわたるファイルの設定を上書きします。

-g

ゲートウェイポートを有効にします。リモートホストをローカル転送ポートに接続できます。 また、構成ファイルで次を使用して、この構成を行うことができます (**GatewayPorts** キーワード)。

注意: クライアントではリモートホスト接続の認証が行われないため、このオプションの使用には細心の注意が必要です (インターネット接続用のネットワークアダプタで使用しないでください)。この接続が転送されるアプリケーションで独自の認証を行わない場合、すべてのリモートホスト接続においてそのアプリケーションへの無制限のアクセスが許可されます。

-h

コマンドオプションの簡単な概要を表示します。

-i *file*

公開鍵認証に使用する代替 ID ファイルを指定します。ファイルの場所は、完全修飾パスまたは相対パスを指定しないかぎり現在の作業ディレクトリと見なされます。既定の ID ファイルは `~/.ssh2/identification` です。 また、構成ファイルで次を使用して ID ファイルを指定することもできます (**IdentificationFile** キーワード)。

-l *username*

リモートコンピュータでのログインに使用する名前を指定します。 また、構成ファイルで次を使用してユーザ名を指定することもできます (**username** キーワード)。(注意: ホスト仕様の一部としてオプションの [user@] を使用する場合、 **-l** が指定したユーザ名より優先されます。)

-L [*protocol*/][*listening_host*:]*listening_port*: *host*: *hostport*

指定されたローカルポートからのデータを、安全なトンネルを介して、指定された宛先ホストおよびポートにリダイレクトします。 Secure Shell 接続が確立されている場合、Secure Shell クライアントは指定のローカルポート (*listening_port*) を使用して、Secure Shell クライアントホストのソケットを開きます(複数のインタフェースを使用しているクライアントホストでは、*listening_host* を使用してインタフェースを指定してください)。(転送するデータが属する) アプリケーションクライアントは、転送されるソケットにデータを送信するように構成します (宛先のホストおよびポートに直接送信しません)。そのクライアントが接続を確立すると、転送されるポートに送信されるデータはすべて、安全なトンネルを介して Secure Shell サーバにリダイレクトされます。このサーバではデータが復号化され、宛先のソケット (host,hostport) に送信されます。ゲートウェイポートオプションが有効でないかぎり、転送されるローカルポートは、Secure Shell クライアントと同じコンピュータで実行されているクライアントのみが使用できます。オプションの *protocol* は、**tcp** または **ftp** にすることができます。 複数のクライアントアプリケーションが転送されるポートを使用できますが、転送は **ssh** が実行されている場合のみ有効です。

注意: 最終宛先ホストおよびポートが Secure Shell サーバホスト上にない場合、Secure Shell ホストとアプリケーションサーバホスト間でデータは平文で送信されます。

また、構成ファイルで次を使用して、ローカル転送を構成することができます (**LocalForward** キーワード)。

-m *mac_algorithm*

クライアントが対応する MAC (メッセージ認証コード) を優先順に指定します。 許可される値は、「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」、および「hmac-ripemd160」です。これらすべてに対応するには「AnyMac」を使用します。「AnyStdMac」を使用して「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」を指定します。hmac-sha256 を指定すると hmac-sha2-256 も有効になります。hmac-sha512 を指定すると hmac-sha2-512 も有効になります。複数の MAC をカンマ区切り一覧として指定することもできます。その他には、「none」、「any」 (AnyMac + 「none」と同等)、および「AnyStd」 (「AnyStdMac」 + 「none」と同等) オプションがあります。MAC について「none」で合意すると、メッセージ認証コードは使用されません。この場合データの整合性は保護されないため、「none」を含むオプションはお勧めしません。

また、構成ファイルで次を使用して、MAC を構成できます (**MACs** キーワード)。既定値は「AnyStdMac」です。

-N

stdin を `/dev/null` からリダイレクトするため、stdin からの読み取りを防止します。 また、構成ファイルで次を使用して、この構成を行うことができます (**DontReadStdin** キーワード)。

-o *option*

構成ファイルのキーワードを使用して構成できる任意のオプションを設定します。キーワードの一覧とその意味については、ssh2_config(5) を参照してください。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
-o key=value1,value2
-o key="value1, value2"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

-p *port*

サーバ上の接続先ポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。 また、構成ファイルで **Port** キーワードを使用してポートを構成することもできます。

-q

クワイエットモードを有効にします。このモードでは、バナーを含むすべての警告および診断メッセージが表示されません。 また、構成ファイルで次を使用して、この構成を行うことができます (**QuietMode** キーワード)。

-R [*protocol*/][*listening_host*:]*listening_port*: *host*: *hostport*

(Secure Shell サーバを実行するコンピュータ上の) 指定されたリモートポートからのデータを、安全なトンネルを介して、指定された宛先ホストおよびポートにリダイレクトします。 Secure Shell 接続が確立されている場合、Secure Shell サーバは指定のリモートポート (*listening_port*) を使用して、そのホスト (Secure Shell サーバが動作しているコンピュータ) のソケットを開きます(複数のインタフェースを使用しているサーバホストでは、*listening_host* を使用してインタフェースを指定してください)。(転送するデータが属する) クライアントアプリケーションは、転送されるソケットにデータを送信するように構成します (宛先のホストおよびポートに直接送信しません)。そのクライアントが接続を確立すると、転送されるポートに送信されるデータはすべて、安全なトンネルを介して Secure Shell クライアントにリダイレクトされます。ここではデータが復号化され、宛先のソケット (host,hostport) に送信されます。オプションの *protocol* は、**tcp** または **ftp** にすることができます。

また、構成ファイルで次を使用して、リモート転送を構成することができます (**RemoteForward** キーワード)。

-s *subsystem*

リモートシステム上の指定されたサブシステムを起動します。サブシステムは Secure Shell プロトコルの機能で、Secure Shell を (**sftp** などの) 他のアプリケーションで安全なトランスポートとして使用できるようにします。サブシステムは、Secure Shell サーバで定義する必要があります。

-S

サーバのセッションチャネルを要求せずに接続します。 セッションチャネル (および tty) が必要ない場合、またはサーバがセッションチャネルを提供しない場合は、ポート転送要求とともに使用できます。

-t

コマンドが指定されている場合も TTY を強制的に割り当てます。 また、構成ファイルで次を使用して、この構成を行うことができます (**ForcePTTYAllocation** キーワード)。

-v

デバッグレベルを冗長モードに設定します。これは、「-d 2」を使用することと同じです。 また、構成ファイルで **VerboseMode** キーワードを使用してこの構成を行うこともできます。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-W *password_file*

接続用に使用するパスワードが記述されたファイルを指定します。パスワードファイルのパーミッションを 600 に設定します。グループまたはそれ以外に読み取りまたは書き込み権限が付与されている場合、ファイルは受け付けられません。また、root 以外のユーザの場合、ID (userid) が変更されている場合はファイルが受け付けられません。このオプションはパスワード認証にのみ適用されます。**AllowedAuthentications** が、パスワード認証の前にキーボード入力を行うよう構成されている場合 (既定)、有効なパスワードファイルが存在する場合でも、パスワードのプロンプトが表示されます。表示されないようにするには、パスワード認証のみに対応するように、またはキーボード入力の前にパスワードの認証が行われるように、許可認証リストを変更します。

注意: 秘密鍵はパスワードと違って暗号化された接続を介して送信されないため、ユーザとの対話が不要な認証を構成するには、パスフレーズなしの公開鍵を使用する方法が安全です。

-X

X11 接続の転送を有効にし、X11 クライアントを信頼されないものとして扱います。ゲストのリモート X11 クライアントは、信頼される X11 クライアントに属するデータを不正に変更できません。 また、構成ファイルで次を使用して、この構成を行うことができます (**ForwardX11** キーワード)。

-x

X11 接続の転送を無効にします。 また、構成ファイルで次を使用して、この構成を行うことができます (**ForwardX11** キーワード)。

-Y

X11 接続の転送を有効にし、X11 クライアントを信頼関係があるクライアントとして扱います。

付 録 G

ssh エスケープシーケンス

クライアントの端末セッションを管理するには、エスケープシーケンスを使用します。 改行文字の後にあるエスケープシーケンスのみが認識されます。 ログインしたばかりの場合は、最初のエスケープシーケンスを入力する前に [Enter] キーを押します。 コマンドラインで **-e** を使用するか、または構成ファイルで **EscapeChar** を使用すると、代替エスケープ文字を構成できます。

次のエスケープシーケンスを使用できます。 既定のエスケープ文字であるチルダ (~) と合わせて表示されています。

~. 接続を終了します。

~^Z ssh を中断します。

~# アクティブな転送接続を一覧表示します。 注意: 転送接続は、ポートが実際にデータを送信している場合のみ一覧表示されます。

~- セッションの間、エスケープ文字の使用を無効にします。

~? 使用できるエスケープシーケンスの一覧を表示します。

~~ エスケープ文字をホストに送信します。 (1 つのエスケープ文字を送信するには、エスケープ文字を 2 回入力します)。

~C コマンドモードを実行します。これは、ポート転送の要求に使用できます。オプションは以下のとおりです。

```
-L[bind_address:]port:host:hostport    Request local forward
-R[bind_address:]port:host:hostport    Request remote forward
-KL[bind_address:]port                 Cancel local forward
-KR[bind_address:]port                 Cancel remote forward
```

~V stderr にバージョン情報を送信します。

~s stderr に接続情報を送信します。

~r 新しい暗号化鍵および整合性鍵を確立するために、すぐに鍵交換を開始します。

~l 回線モードに入ります。 キーストロークはバッファに保存され、[Enter] キーを押した時に出力されます。

~B リモートシステムに BREAK を送信します。

付 録 H

ssh Exit 値

Exit 値は、トラブルシューティングを支援する目的で用意されています。スクリプトでは、エラー処理にゼロかゼロ以外のみ使用することをお勧めします。ゼロ以外の特定の値を探しても、オペレーティングシステムおよびサーバによってさまざまであるため、信頼できません。

エラーコードは、値 0 ～ 255 に制限されています。エラー 65 ～ 79 は切断状態で、RFC4253 に定義されているように、64 にホストから返されたエラー値を足して計算されます。エラー値 128 ～ 254 はシステムシグナルで、128 にシグナル値を足して計算されます。エラー値 255 は、**ssh** が **scp** などの別のプロセスによって実行された後に失敗すると返されます。

0 成功です。

1 一般的なエラーです。

2 リモートホスト接続が失敗しました。

65 このクライアントアドレスに対し、ホストによってアクセスが拒否されました。

66 プロトコルエラーです。

67 鍵の交換ができませんでした。

68 認証ができませんでした。

69 MAC エラーです。

70 圧縮エラーです。

71 使用可能なサービスがありません。

72 このプロトコルバージョンには対応していません。

73 ホスト鍵の信頼性を確認できません。

74 接続が失われました。

75 アプリケーションによって切断されました。

76 接続が多すぎます。

77 ユーザによって取り消されました。

78 試していない認証方式はもうありません。

79 不明なユーザ名です。

付 録 I

ssh-keygen コマンドラインオプション

ssh-keygen の構文は次のとおりです。

ssh-keygen [**-7** *file*] [**-b** *bits*] [**-c** *comment*] [**-D** *private_key*] [**-e** *private_key*] [**-f**] [**-F** *key*] [**-h**] [**-H** *key*] [**-i** *key*] [**-k** *file*] [**-N** *new_passphrase*] [**-o** *key_name*] [**-O** *key_file*] [**-p** *passphrase*] [**-P**] [**-q**] [**-t** *key_type*] [**-V**] [**-X** *cert*] [*key_name1 key_name2* ...]

公開鍵認証用の RSA 鍵および DSA 鍵を作成する場合、既存の鍵のプロパティを編集する場合、およびほかの Secure Shell 実装との互換性のために鍵ファイルの形式を変換する場合は、**ssh-keygen** を使用します。

オプションを何も指定しないと、**ssh-keygen** によって 2048 ビット RSA 鍵のペアが生成され、秘密鍵を保護するためのパスフレーズの入力が求められます。 コマンドラインでファイル名を指定しないと、鍵は ~/.ssh2/ に作成され、鍵の種類とサイズ、およびホスト名を識別する既定の名前 (例えば **/home/joe/.ssh2/id_rsa_2048_myhost_a**) が付けられます。 ファイル名を指定すると、完全修飾パス名を含めないかぎり、鍵は現在の作業ディレクトリに保存されます。 作成した秘密鍵ごとに **ssh-keygen** によって公開鍵も生成されます。 公開鍵には秘密鍵と同じベース名が付けられ、さらに .pub 拡張子が追加されます (例えば **id_rsa_2048_myhost_a.pub**)。

コマンドラインオプション

オプションは、1 文字の形式 (**-b**など) と、同様の意味を持つ記述語 (**--bits**)。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、次を使用します (**-h** コマンドラインオプション)。

-7 *file*

指定された PKCS#7 ファイルから証明書および CRL を抽出します。

-b *bits*

鍵のサイズを指定します。鍵のサイズを大きくすると、ある程度までセキュリティは向上します。鍵のサイズを大きくすると最初の接続が遅くなりますが、正常に接続した後は、鍵のサイズはデータストリームの暗号化や解読の速度に影響しません。使用する鍵の長さは、多くの要素に依存します。その要素には、鍵の種類、鍵の有効期間、保護するデータの値、潜在的な攻撃者にとって利用可能なリソース、この非対称鍵とともに使用する対称鍵のサイズなどがあります。ニーズに合った最適な選択をするには、セキュリティ管理者にお問い合わせください。 既定では、RSA 鍵は 2048 ビットで、DSA 鍵は 1024 ビットです。許可される最小値は 512 です。許可される最大値は 32768 です。

-c *comment*

鍵ファイル内のコメントフィールドの情報を指定します。文字列にスペースが含まれる場合は引用符を使用します。コメントを指定しないと、鍵の種類、作成者、日付、および時間が含まれる既定のコメントが作成されます。注意: コメントは、パスフレーズで保護された鍵がクライアント認証に使用される時に表示されます。コメントには、パスフレーズなどの機密情報を保存しないでください。

-D *private_key*

指定された秘密鍵を基に、公開鍵の新しいコピーを生成します。

-e *private_key*

指定された秘密鍵のパスフレーズを変更します。このオプションのみを使用した場合、指定した秘密鍵の古いパスフレーズと新しいパスフレーズの入力を求められます。対話セッションを開かずにパスフレーズを編集するには、このオプションを次と組み合わせて使用します (**-p** および**-N**)。NULL パスフレーズに変更するには、このオプションを次と組み合わせて使用します (**-P**)。

-f

FIPS モードを有効にします。このモードでは、FIPS 承認の鍵の強さを使用した鍵の作成が強制されます。

-F *key*

指定された鍵の指紋を Bubble Babble 形式で表示します。

-h

コマンドオプションの簡単な概要を表示します。

-H *key*

指定された Reflection 公開鍵を基に OpenSSH 形式で公開鍵を生成します。変換された鍵は、同じベースファイル名に .ssh 拡張子が追加されて作成されます。作成された鍵を使用して、OpenSSH サーバで公開鍵クライアント認証を構成できます。

-i *key*

指定された鍵に関する情報を表示します。

-k *file*

指定された PKCS #12 ファイルから証明書および秘密鍵を抽出します。

-N *new_passphrase*

パスフレーズを指定された新しいパスフレーズに変更します。このオプションは、次と組み合わせて使用します (**-e**)。

-o *key_name*

生成される秘密鍵のファイル名を指定します(公開鍵も作成され、常に秘密鍵と同じ名前となり、.pub というファイル拡張子が付きます)。 注意: 次の末尾に 1 つ以上の鍵ファイル名を指定して、鍵ファイルに名前を付けることもできます (**ssh-keygen** コマンド)。

-o *key*

指定された OpenSSH 公開鍵または秘密鍵を基に、Reflection 形式で公開鍵または秘密鍵を作成します。変換された鍵は、同じベースファイル名に .ssh2 拡張子が追加されて作成されます。

-p *passphrase*

パスフレーズを指定します。フレーズにスペースが含まれる場合は引用符を使用します。このオプションによって、新しい鍵を生成した時に初期パスフレーズが作成されます。既存の鍵を管理する場合は、このオプションを使用してその鍵を保護するパスフレーズを指定します。パスフレーズが必須の場合に **-p**を使用しないと、パスフレーズの入力を求められます。パスワードの長さと複雑さの程度については、会社のセキュリティポリシに従ってください。

-P

パスフレーズなしで鍵を作成します。サーバ認証用の鍵を作成する場合に、このオプションを使用できます。クライアント鍵にはパスフレーズを使用することを強くお勧めします。パスフレーズなしの鍵の使用は、無人での認証 (ファイル転送スクリプトなど) が必要なアカウントだけに制限する必要があります。パスフレーズなしの秘密鍵ファイルは、オペレーティングシステムのファイルアクセス制御 (鍵ファイル = 400、鍵が格納されたディレクトリ = 700) を使用して保護する必要があります。

-q

鍵生成進捗インジケータを非表示にします。

-t *key_type*

鍵の生成に使用する鍵のアルゴリズムを指定します。 設定できる値は「rsa」および「dsa」です。既定値は「rsa」です。

-V

次を表示します (**ssh-keygen** バージョン情報)。

-X *cert*

指定された X.509 証明書ファイルから公開鍵を抽出します。

[*key_name1 key_name2*...]

生成される秘密鍵に使用されるファイル名を指定します(鍵が複数の場合は複数のファイル名)。公開鍵は、同じ名前に「.pub」ファイル拡張子が追加されて作成されます。

付 録 J

scp コマンドラインオプション

scp 構文は次のとおりです。

scp [**-4**] [**-6**] [**-a** [*arg*]] [**-b** *buffer_size*] [**-B**] [**-c** *cipher*] [**-d**]
[**-D** *debug_level*] [**-F** *file*] [**-h**] [**-i** *file*] [**-N** *max_requests*] [**-o** *option*]
[**--overwrite**] [**-p**] [**-P** *port*] [**-q**] [**-Q**] [**-Q**] [**-r**] [**-u**] [**-v**] [**-V**]
[**-W**] [[*user*@]host[#port]:]file_or_dir ...[[user@]host[#port]:]file_or_dir

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

注意: コマンドラインで指定したすべてのオプション (ユーザ名、ホスト名、およびその他の機密情報) は、プロセス状態 (**ps**) 一覧に表示されます。機密のオプションやスイッチを指定する場合は、他のユーザが容易にその情報を参照できないように注意する必要があります。安全な方法として、これらのオプションを構成ファイル内で設定して、推奨のファイルのパーミッション (構成ファイル = 600、ファイルの格納ディレクトリ = 700) で構成ファイルを保護する方法が考えられます。

-4

接続に IPv4 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-6

接続に IPv6 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-a [*newline_type*]

ASCII モードでファイルを転送します。改行変換を処理するには、オプションの引数を使用します。「unix」または「dos」を指定できます。既定では、*newline_type* に指定した値によって宛先の改行規則が設定されますが、ソースまたは宛先の規則を指定することもできます。指定するには、引数の接頭語として「src: 」または「dest: 」を付加します。たとえば、以下のようになります。

scp -a src: unix –a dest: dos unixhost: src_file winhost: dest_file

既定値は、「dest: unix」、「src: unix」です。宛先の種類とソースの種類が同じ場合、変換は行われません。それ以外の場合、「src」および「dest」の改行の種類に指定した値に基づいて変換が行われます。

ソースまたは宛先の規則なしで **-a** が使用されている場合、クライアントはソースまたは宛先の行末規則を、接続が確立されているサーバから取得しようとします。サーバがこの機能に対応していない場合、DOS の行末規則が適用されます。

-b *buffer_size*

データ転送に使用されるバッファサイズを指定します。既定は 32768 バイトです。許可される最小値は 1024 です。許可される最大値は 4194304 バイトです。ほとんどの場合、既定値でほぼ最適な転送速度が得られます。一部のシステムでは、バッファサイズを適度に増やすとパフォーマンスが改善されることがあります。注意: バッファサイズを非常に大きくしても、パフォーマンスが改善されることはまずありません。逆に、転送スピードが低下したり、バッファサイズに対応していないサーバとの転送が失敗したり、クライアントまたはサーバのメモリ制限を超えたために致命的なエラーが発生する、といった問題を引き起こすことがあります。

-B

バッチモードで **scp** を実行します。ユーザ入力のクエリーはすべての無効になります。これは、スクリプトおよびバッチジョブに便利です。このオプションは、ユーザとの対話が必要な認証方法には対応していません。バッチモードでは、**--overwrite** の設定を「no」にしなければ、**scp** によって常に既存の宛先ファイルが上書きされます。

-c *cipher*

対応する 1 つ以上の (カンマで区切った) 暗号化アルゴリズムを指定します。特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。 許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。

この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。

あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと 128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。

暗号を指定しない場合、Secure Shell 構成ファイル ssh2_config(5) 内の **Ciphers** キーワードによって暗号が決定されます。既定値は「AnyStdCipher」です。

-d

宛先が既に存在するディレクトリであることを要求します。例えば、以下のコマンドを実行すると、`destination` というディレクトリが存在すれば、このディレクトリに `source_file` がコピーされます。このディレクトリが存在しない場合、`source_file` は `demo` ディレクトリにコピーされ、`destination` というファイル名が割り当てられます。

```
scp source_file joe@myhost:~/demo/destination
```

以下のコマンドに **-d** フラグを追加すると、`source_file` が `destination` ディレクトリにコピーされますが、このディレクトリが存在しない場合はエラーになります。

```
scp -d source_file joe@myhost:~/demo/destination
```

-D *debug_level*

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。 1、2、3、または 99 を使用します (値 4 〜 98 もエラーにはなりませんが、3 と同じものとして扱われます)。

-F *file*

追加の構成ファイルを指定します。既定のユーザ固有のファイル (`~/.ssh2/ssh2_config`) およびシステム全体にわたるファイル (`/etc/ssh2/ssh2_config`) に加えて、このファイルからも設定が読み取られます。このファイルの設定は、ユーザ固有のファイルおよびシステム全体にわたるファイルの設定を上書きします。

-h

コマンドオプションの簡単な概要を表示します。

-i *file*

公開鍵認証に使用する代替 ID ファイルを指定します。ファイルの場所は、完全修飾パスまたは相対パスを指定しないかぎり現在の作業ディレクトリと見なされます。既定の ID ファイルは `~/.ssh2/identification` です。

-N *max_requests*

同時要求の最大数を指定します。この値を増やすと、ファイルの転送スピードがわずかに改善されることがありますが、メモリの使用量も増えます。既定は 256 です。

-o *option*

構成ファイルのキーワードを使用して構成できる任意のオプションを設定します。キーワードの一覧とその意味については、ssh2_config(5) を参照してください。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
-o key=value1,value2
-o key="value1, value2"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

--overwrite [yes|no|ask]

既存の宛先ファイルを上書きするかどうかを指定します。許可される値は「yes」、「no」、および「ask」です。既定は「yes」です。注意: コピー元ファイルと同一のファイルが既にコピー先に存在する場合は、Smart Copy 機能により、実際のファイル転送処理は行われません。

-p

元のファイルの変更時間およびファイル属性を維持します。

-P *port*

サーバ上の接続先ポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。 また、構成ファイルで **Port** キーワードを使用してポートを構成することもできます。

-q

クワイエットモードで実行します。致命的エラーのみが表示されます。

-Q

進捗インジケータを非表示にします。

-r

すべてのサブディレクトリを含めて再帰的にコピーします。

-u

宛先の場所へのコピーが完了したら、ソースファイルを削除します。

-v

デバッグレベルを冗長モードに設定します。これは、デバッグレベルを 2 に設定することと同じです。 また、構成ファイルで **VerboseMode** キーワードを使用してこの構成を行うこともできます。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-W *password_file*

接続用に使用するパスワードが記述されたファイルを指定します。パスワードファイルのパーミッションを 600 に設定します。グループまたはそれ以外に読み取りまたは書き込み権限が付与されている場合、ファイルは受け付けられません。また、root 以外のユーザの場合、ID (userid) が変更されている場合はファイルが受け付けられません。このオプションはパスワード認証にのみ適用されます。**AllowedAuthentications** が、パスワード認証の前にキーボード入力を行うよう構成されている場合 (既定)、有効なパスワードファイルが存在する場合でも、パスワードのプロンプトが表示されます。表示されないようにするには、パスワード認証のみに対応するように、またはキーボード入力の前にパスワードの認証が行われるように、許可認証リストを変更します。

注意: 秘密鍵はパスワードと違って暗号化された接続を介して送信されないため、ユーザとの対話が不要な認証を構成するには、パスフレーズなしの公開鍵を使用する方法が安全です。

付 録 K

sftp コマンドラインオプション

sftp 構文は次のとおりです。

sftp [**-4**] [**-6**] [**-b** *buffer_size*] [**-B** *batch_file*] [**-c** *cipher*] [**-D** *debug_level*] [**-h**] [**-m** *mac_algorithm*] [**-N** *max_requests*] [**-o** *option*] [**-P** *port*] [**-v**] [**-V**] [**-W**] [[*user*@]*host*[#*port*]]

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

注意: コマンドラインで指定したすべてのオプション (ユーザ名、ホスト名、およびその他の機密情報) は、プロセス状態 (**ps**) 一覧に表示されます。機密のオプションやスイッチを指定する場合は、他のユーザが容易にその情報を参照できないように注意する必要があります。安全な方法として、これらのオプションを構成ファイル内で設定して、推奨のファイルのパーミッション (構成ファイル = 600、ファイルの格納ディレクトリ = 700) で構成ファイルを保護する方法が考えられます。

-4

接続に IPv4 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-6

接続に IPv6 のアドレスのみを使用します。IP アドレスの要件を構成するために、**AddressFamily** キーワードを使用することもできます。

-b *buffer_size*

データ転送に使用されるバッファサイズを指定します。既定は 32768 バイトです。許可される最小値は 1024 です。許可される最大値は 4194304 バイトです。ほとんどの場合、既定値でほぼ最適な転送速度が得られます。一部のシステムでは、バッファサイズを適度に増やすとパフォーマンスが改善されることがあります。注意: バッファサイズを非常に大きくしても、パフォーマンスが改善されることはまずありません。逆に、転送スピードが低下したり、バッファサイズに対応していないサーバとの転送が失敗したり、クライアントまたはサーバのメモリ制限を超えたために致命的なエラーが発生する、といった問題を引き起こすことがあります。

-B *batch_file*

バッチ処理 **sftp** コマンドで使用するファイルを指定します。ログインが正常に行われると、**sftp** によって指定のファイル内のコマンドが順に実行され、**bye**、**exit**、または **quit** コマンドが検出されると接続が終了します。このモードは、ユーザとの対話を必要とする認証方法に対応していません。バッチファイルでは、以下に示した任意の対話型コマンドを使用できます。バッチファイル内のコマンドの実行に失敗しても、**sftp** は残りのコマンドの実行を続行し、最初に失敗したコマンドのエラーコードを返します。ただし、接頭語に「-」(ダッシュ) が付いたコマンドは、失敗しても常に 0 を返します。

-c *cipher*

対応する 1 つ以上の (カンマで区切った) 暗号化アルゴリズムを指定します。 特定のセッション用に使用される暗号は、クライアントの優先順位が最高の暗号であり、サーバもこの一覧に対応しています。 許可された値は、「aes128-ctr」、「aes128-cbc」、「aes192-ctr」、「aes192-cbc」、「aes256-ctr」、「aes256-cbc」、「blowfish-cbc」、「arcfour'」、「arcfour128」、「arcfour256」、「cast128-cbc」、および 「3des-cbc」です。

この値を「none」に設定することもできます。暗号について「none」で合意すると、データは暗号化されません。この方法は機密保護の機能を持たないのでお勧めしません。

あらかじめ次の値が用意されています。 「aes」(-ctr/-cbcモードと 128/192/256 bit の全組み合わせ)、「blowfish」(「blowfish-cbc」と同等)、「cast」(「cast128-cbc」と同等)、「3des」(「3des-cbc」と同等)、「Any」または「AnyStd」(使用可能なすべての暗号化 + 「none」)、および「AnyCipher」または「AnyStdCipher」(使用可能なすべての暗号化)。 暗号を指定しない場合、Secure Shell 構成ファイル ssh2_config(5) 内の **Ciphers** キーワードによって暗号が決定されます。既定値は「AnyStdCipher」です。

-D *debug_level*

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。

-h

コマンドオプションの簡単な概要を表示します。

-m *mac_algorithm*

この接続で使用可能な MAC (メッセージ認証コード) を指定します。許可される値は、「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」、および「hmac-ripemd160」です。これらすべてに対応するには「AnyMac」を使用します。「AnyStdMac」を使用して「hmac-sha256」、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-sha512」を指定します。hmac-sha256 を指定すると hmac-sha2-256 も有効になります。hmac-sha512 を指定すると hmac-sha2-512 も有効になります。複数の MAC をカンマ区切り一覧として指定することもできます。その他には、「none」、「any」 (AnyMac + 「none」と同等)、および「AnyStd」(「AnyStdMac」 + 「none」と同等) オプションがあります。MAC について「none」で合意すると、メッセージ認証コードは使用されません。この場合データの整合性は保護されないため、「none」を含むオプションはお勧めしません。

-N *max_requests*

同時要求の最大数を指定します。この値を増やすと、ファイルの転送スピードがわずかに改善されることがありますが、メモリの使用量も増えます。既定は 256 です。

-o *option*

構成ファイルのキーワードを使用して構成できる任意のオプションを設定します。キーワードの一覧とその意味については、ssh2_config(5) を参照してください。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
-o key=value1,value2
-o key="value1, value2"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

--overwrite [yes|no|ask]

既存の宛先ファイルを上書きするかどうかを指定します。許可される値は「yes」、「no」、および「ask」です。既定は「yes」です。注意: コピー元ファイルと同一のファイルが既にコピー先に存在する場合は、Smart Copy 機能により、実際のファイル転送処理は行われません。

-P *port*

サーバ上の接続先ポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。 また、構成ファイルで **Port** キーワードを使用してポートを構成することもできます。

-v

デバッグレベルを冗長モードに設定します。これは、デバッグレベルを 2 に設定することと同じです。 また、構成ファイルで **VerboseMode** キーワードを使用してこの構成を行うこともできます。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-W *password_file*

接続用に使用するパスワードが記述されたファイルを指定します。パスワードファイルのパーミッションを 600 に設定します。グループまたはそれ以外に読み取りまたは書き込み権限が付与されている場合、ファイルは受け付けられません。また、root 以外のユーザの場合、ID (userid) が変更されている場合はファイルが受け付けられません。このオプションはパスワード認証にのみ適用されます。**AllowedAuthentications** が、パスワード認証の前にキーボード入力を行うよう構成されている場合 (既定)、有効なパスワードファイルが存在する場合でも、パスワードのプロンプトが表示されます。表示されないようにするには、パスワード認証のみに対応するように、またはキーボード入力の前にパスワードの認証が行われるように、許可認証リストを変更します。

注意: 秘密鍵はパスワードと違って暗号化された接続を介して送信されないため、ユーザとの対話が不要な認証を構成するには、パスフレーズなしの公開鍵を使用する方法が安全です。

付 録　L

対応している sftp コマンド

以下のコマンドを対話型 **sftp** セッションおよび **sftp** バッチファイルで使用できます。

ascii [**-s**] [*remote_newline*] [*local_newline*]

現在のファイル転送モードを ASCII に設定します。ASCII モードは、行末文字を変換するのに便利です。既定の改行処理より優先させるには、*remote_newline* と *local_newline* を使用します。 *remote_newline* で使用できる値は、「DOS」(\r\n) と「Unix」 (\n) です。リモートの行末規則に明確な値が規定されていない場合、リモートホストは規則を規定するよう求められます。リモートホストがこの機能に対応していない場合、DOS の行末規則が適用されます。*local_newline* に指定できる値は、「Unix」 (\n) のみです。現在の転送モードを表示するには **-s** を使用します。

auto

転送モードを自動に設定します。自動モードでは、転送方式はファイル拡張子によって決まります。指定されたファイル拡張子を持つファイルには ASCII 転送が使用され、それ以外のファイルにはバイナリ転送が使用されます。ASCII ファイルの種類の既定の一覧は「`txt, htm*, pl, php*`」です。指定の **sftp** セッション用にこの一覧を変更するには、**setext** コマンドを使用します。既定のファイル拡張子一覧を変更するには、クライアントキーワード **FileCopyAsciiExtensions** を使用します。

binary

転送モードをバイナリに設定します。このモードでは、変更は一切行わずにファイルを転送します。バイナリは既定の転送モードです。このコマンドは、バッチスクリプト内部で ASCII モードをオフにするのに便利です。

bye

quit と同様に機能します。

cd *directory*

リモートディレクトリを *directory* に変更します。

chgrp *group file*

file によって指定されたファイルまたはディレクトリのグループ所有者を *group* に設定します。グループは、数値のグループ ID (GID) として指定する必要があります。

chmod[**-R**]*mode file*

file によって指定されたファイルまたはディレクトリのファイル権限を設定します。モードは、数値形式 (例: 664) で指定する必要があります。ファイルとディレクトリを再帰的に変更するには **-R** を使用します。

chown *owner file*

file によって指定されたファイルまたはディレクトリの所有者を *owner* に設定します。所有者は、数値のユーザ ID (UID) として指定する必要があります。

close

sftp なしで、リモートサーバとの接続を終了します。

debug *debug_level* |**disable** | **no**

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。「disable」または「no」を使用してデバッグを無効にします。

dir

ls と同様に機能します。

exit

quit と同様に機能します。

get [**--preserve**] | [**-p**] *remote-file* [*remote-file ...*]

指定された 1 つまたは複数のファイルを現在のローカル作業ディレクトリにコピーします(別の場所にコピーするには、**lcd** を使用してローカルの作業ディレクトリを変更します)。同じ名前のファイルがすでに存在する場合は、既存のファイルが上書きされます。ワイルドカードに対応していますが、名前の置換はファイル名のみに適用され、ディレクトリには適用されません。ファイル属性とタイムスタンプを予約するには、 **--preserve** または **-p** を使用します。

getext

自動モードが有効になっている場合に ASCII ファイル転送を使用するファイル拡張子の現在の一覧を表示します。自動モードを有効にするには **auto** を使用します。現在のセッション用にこのリストを変更するには **setext** を使用します。既定のリストを変更するには、クライアントキーワード **FileCopyAsciiExtensions** を使用します。

help | **?** [*command*]

sftp ヘルプを表示します。指定したコマンドに関するヘルプを表示するには *command* を使用します。

lcd *directory*

ローカルディレクトリを *directory* に設定します。

lls [**-1** | **-a** | **-f** | **-l** | **-n** | **-r** | **-S** | **-t** | **--**] [*file*]

ローカルディレクトリの一覧を表示します。オプションは以下のとおりです。

-1（1 つの列）
-a（非表示ファイルの表示）
-f（ソートしない）
-l（ロングリスト形式）
-n（数字のユーザおよびグループ ID 付きのロングリスト形式）
-r（逆順）
-S（サイズによるソート）
-t（ファイルのアクセス時間によるソート）
--（後続のハイフンを通常文字として扱う）

lmkdir *directory*

指定のローカルディレクトリを作成します。

ln

symlink と同様に機能します。

lpwd

ローカル作業ディレクトリを表示します。

ls [**-1** | **-a** | **-f** | **-l** | **-n** | **-r** | **-S** | **-t** | **--**] [*file*]

リモートディレクトリの一覧を表示します。このオプションは、上記の **lls** と同様に機能します。

mget

get と同様に機能します。

mkdir *directory*

指定のリモートディレクトリを作成します。

mput

put と同様に機能します。

open [**-l** | [*user*@]*host*]

指定のホストへの接続を開きます。ローカルホストに接続するには、**-l** を使用します。この場合、ローカルとリモートの両方のコマンドがローカルファイルシステムのファイル上で動作します。

put [**--preserve**] | [**-p**] *local_file* [*local_file ...*]

指定された 1 つまたは複数のファイルを現在のリモート作業ディレクトリにコピーします(別の場所にコピーするには、**cd** を使用してリモートの作業ディレクトリを変更します)。同じ名前のファイルがすでに存在する場合は、既存のファイルが上書きされます。ワイルドカードに対応していますが、名前の置換はファイル名のみに適用され、ディレクトリには適用されません。ファイル属性とタイムスタンプを予約するには、 **--preserve** または **-p** を使用します。

pwd

リモート作業ディレクトリを表示します。

quit

sftp を終了して、接続を終了します。

rename *source destination*

ファイル名を *source* から *destination* に変更します。転送先ファイルが既に存在する場合、名前の変更は行われません。

rm *file*

指定の 1 つまたは複数のリモートファイルを削除します。ワイルドカードに対応しています。

rmdir *directory*

指定のリモートディレクトリを削除します。

setext

自動モードが有効になっている場合に ASCII ファイル転送を使用するファイル拡張子の現在の一覧を指定します。複数の拡張子を指定するには、カンマまたは空白区切りの一覧を使用します。このコマンドでは値は累積されません。ワイルドカード (zsh-glob) 文字を指定できます。ファイル拡張子の前にピリオドを付けてはなりません。空白を含む拡張子を指定するには、その拡張子を引用符で囲むか、エスケープ文字としてバックスラッシュを使用します。自動モードを有効にするには **auto** を使用します。現在のリストを表示するには **getext** を使用します。既定のリストを変更するには、クライアントキーワード **FileCopyAsciiExtensions** を使用します。

symlink *linked_path target_path*

linked_path からリモートホストの *target_path* にシンボリックリンク (ソフト) を作成します。

verbose

デバッグレベルを冗長モードに設定します。これは、デバッグレベルを 2 に設定することと同じです。冗長モードを無効にするには、「debug disable」を使用します。

version

使用可能な SFTP プロトコルのバージョンを表示します。

付 録 M

ssh-add コマンドラインオプション

ssh-add の構文は次のとおりです。

ssh-add [**-c**] [**-d**] [**-D**] [**-h**] [**-l**] [**-L**] [**-p**] [**-t** *timeout*] [**-U**] [**-V**] [*file1 file2* ...]

file1, *file2...* を使用して、エージェントに追加する鍵を指定します。鍵ファイルの指定はオプションです。鍵ファイルを指定しない場合、**ssh-add** によって、ID ファイル (既定では `~/.ssh2/identification`) に指定したすべての鍵が追加されます。

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

-c

エージェントが鍵を使用する前に確認を要求するように指定します。

-d

エージェントから指定された 1 つ以上の鍵を削除します。 鍵ファイルを指定するには、*file* 引数を使用します。

-D

エージェントからすべての ID を削除します。

-h

コマンドオプションの簡単な概要を表示します。

-l

鍵エージェントに現在読み込まれているすべての ID を一覧表示します。

-L

鍵エージェントをロックします。 パスワードの入力を求められます。これは、エージェントのロックを解除するのに必要になります。 ロックを解除するには **-U** を使用します。

-p

stdin からパスフレーズを読み取ります。 これは、パイプ上で実行できます。

-t <*timeout*>

鍵の待ち時間を設定します。 制限しない場合は、ゼロ (0) を使用します。 鍵は指定した待ち時間が経過すると削除されます。

既定では、待ち時間の値は分単位で設定します。次の構文を使用して、他の単位を指定できます。

n<単位>[n<単位>…]

ここで、単位は、s (秒)、m (分)、h (時)、d (日)、または w (週) です (単位の指定は大文字でも小文字でもかまいません。同じ意味になります)。たとえば、以下のようになります。

`3600s` = 3600 秒 (1 時間)

`2w` = 2 週間

`2d4h` = 2 日と 4 時間

-U

ロックされたエージェントを **-L** を使用してロック解除します。 必須のパスワードの入力を求められます。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-x

追加される鍵ファイルが X.509 証明書に関連付けられるように指定します。 ファイルを指定せずに **-x** を使用した場合、Reflection for Secure IT では ID ファイル (既定で `~/.ssh2/identification`) が読み込まれ、**CertKey** キーワードによって識別されるすべての鍵が追加されます。 証明書は、関連付けられた秘密鍵と同じディレクトリ内にあり、同じベース名に `.crt` ファイル拡張子が付けられた名前である必要があります。

付 録 N

ssh-agent コマンドラインオプション

ssh-agent の構文は次のとおりです。

ssh-agent [**-c**] [**-d** *debug_level*] [**-h**] [**k**] [**-s**] [**-t** *timeout*] [**-V**] [*command*]

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

-c

シェルを強制的に csh にします。既定では、**ssh-agent** は SHELL 環境変数を使用して起動するシェルを判断します。このオプションは、既定の動作を上書きします。

-d *debug_level*

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。 1、2、3、または 99 を使用します (値 4 〜 98 もエラーにはなりませんが、3 と同じものとして扱われます)。

-h

コマンドオプションの簡単な概要を表示します。

-k

(SSH_AGENT_PID 環境変数によって示された) 現在のエージェントを削除します。

-s

シェルを強制的に sh にします。既定では、**ssh-agent** は SHELL 環境変数を使用して起動するシェルを判断します。このオプションは、既定の動作を上書きします。

-t *timeout*

エージェントに追加されるすべての鍵の既定待ち時間を設定します。鍵は指定した待ち時間が経過すると削除されます。既定では、鍵には待ち時間の制限がありません。これは待ち時間の値にゼロ (0) を設定することと同じです (また、鍵を追加する時に **ssh-add -t** オプションを使用して待ち時間を指定することもできます。鍵追加時の指定は、この設定を上書きします)。

既定では、待ち時間の値は分単位で設定します。次の構文を使用して、他の単位を指定できます。

n<単位>[n<単位>…]

ここで、単位は、s (秒)、m (分)、h (時)、d (日)、または w (週) です (単位の指定は大文字でも小文字でもかまいません。同じ意味になります)。たとえば、以下のようになります。

3600s = 3600 秒 (1 時間)

2w = 2 週間

2d4h = 2 日と 4 時間

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

付 録 O

sshd コマンドラインオプション

-4

接続に IPv4 のアドレスのみを使用します。また、次を使用して、IP アドレスの要件を構成することもできます (**AddressFamily** キーワード)。

-6

接続に IPv6 のアドレスのみを使用します。また、次を使用して、IP アドレスの要件を構成することもできます (**AddressFamily** キーワード)。

-b

このオプションを使用すると、 **sshd** は切り離されず、デーモンにはなりません。これは監視用に使用できます。

-d *level*

デバッグレベルを設定し、デバッグ出力を stderr に送信します。 1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。 このオプションでは、 **sshd** は、1 つのクライアント接続の情報のみをログ記録し、このクライアント接続が閉じると終了します。

注意: 記録の設定を 99 にすると、セキュリティのリスクが高まります。このレベルでは、暗号化されていないプロトコル情報が書き出されることがあるため、情報の漏洩が懸念されます。また、大量に書き込まれる情報によりディスクスペースが急速に占有され、ホストや Reflection for Secure IT の応答が停止する可能性があります。

-D *level*

デバッグレベルを設定し、デバッグ出力をファイルに送信します。 1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。 このオプションでは、 **sshd** は、クライアント接続が閉じても終了しません。この設定は、ルートのみが使用できます。

注意: 記録の設定を 99 にすると、セキュリティのリスクが高まります。このレベルでは、暗号化されていないプロトコル情報が書き出されることがあるため、情報の漏洩が懸念されます。また、大量に書き込まれる情報によりディスクスペースが急速に占有され、ホストや Reflection for Secure IT の応答が停止する可能性があります。

出力ファイルの格納場所は `/etc/ssh2/logs` で、ファイル名の形式は `debugYYMMDD_HHMMSS_uniqueID`です。ここで、YY は年、MM は月、DD は日、HH は時間、MM は分、SS は秒を表し、uniqueID は同時に起動されたサーバで別のログを使用するための一意の値を表します。

-f *config_file*

サーバ構成ファイルの代替名と場所を指定します。サーバは、既定ファイルの代わりに指定されたファイルを読み取ります。(既定の構成ファイルは `/etc/ssh2/sshd2_config`。)

-g *login_timeout*

クライアント認証に許可される秒数を設定します。クライアントが指定された秒数内にユーザを認証できないと、サーバは接続を切断して終了します。 制限しない場合は、ゼロ (0) を使用します。

-m *file*

このオプションを使用して、F-Secure 構成ファイルか以前のバージョンの Reflection for Secure IT から設定を移行します。移行するサーバ構成ファイルの名前を指定する必要があります。

-h *host_key_file*

サーバの認証に使用する秘密鍵のファイル名と格納場所を指定します。 完全修飾されていない場合、パスは次への相対パスであると見なされます (/etc/ssh2)。既定値は次のとおりです (/etc/ssh2/hostkey)。

-o *option*

構成ファイルのキーワードを使用して構成できる任意のオプションを設定します。キーワードの一覧とその意味については、sshd2_config(5) を参照してください。コマンドラインで構成されたオプションは、構成ファイルで構成されたオプションを上書きします。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
-o key=value1,value2
-o key="value1, value2"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

-p *port*

サーバがリッスンするポートを指定します。既定は 22 で、これは Secure Shell 接続の標準ポートです。 コマンドラインの値が構成ファイルで設定された値より優先されます。ポートは 1 つのみ許可されます。複数のポートを構成した場合、最後に構成されたポートが使用されます。

-q

クワイエットモードを有効にします。このモードでは、エラーのみがシステムログに記録されます。(**-d** および **-D** は、 **-q** を同一のコマンドラインで使用すると無視されます。)

-v

デバッグレベルを冗長モードに設定します。これは、「-d 2」を使用することと同じです。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

付 録　P

ssh-certview コマンドリファレンス

同義語

ssh-certview [**-C**] [**-c**] [**-d** *debug_level*] [**-h**] [**-q**] [**-V**] [**-v**] [*file ...*]

説明

X.509 証明書の内容 (PEM 形式または DER 形式のいずれか)、CRL 一覧、または PKCS#10 要求を表示する場合は、**ssh-certview** を使用します。さらに、許可される ID への証明書のマッピングに Reflection PKI Services Manager で使用される pki_mapfile(5) 用の構文のサンプルを出力することもできます。

証明書フィールドの **ssh-certview** 出力は RFC2253 に準拠しています。この規格に準拠するために、[Subject] (サブジェクト) フィールドと [Issuer] (発行者) フィールドは共通名 (CN) で始まります (「CN = Secure CA, O = Secure Corporation, C = US」など)。この形式は Reflection PKI Services Manager によっても使用されます。

注意: ほかのユーティリティ (旧バージョンの Reflection for Secure IT を含む) では、[Subject] (サブジェクト) フィールドの出力内容の順番が逆になります。逆順の形式は同じ意味にはならないので、PKI Services Manager マップファイルで使用された場合、一致は発生しません。

オプション

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

-C

出力において各行の先頭にコメントマーク (#) を追加するよう指定します。

-c

証明書から内容を抽出し、pki_mapfile(5) に含めるための正しい構文を出力します。さらに **-q** も指定されないかぎり、標準出力も含まれ、その前にコメントマークが付けられます。

-d *debug_level*

デバッグレベルを設定します。値を上げると、表示される情報量が増えます。1、2、3、または 99 を使用します (値 4 ～ 98 もエラーにはなりませんが、3 と同じものとして扱われます)。

-h

コマンドオプションの簡単な概要を表示します。

-q

マップファイルの構文を除き、すべての出力の表示をオフにします。このオプションを **-c** とともに使用すると、コメントアウトされた証明書情報を省略してマップファイルの構文だけを出力できます。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-v

出力データの冗長レベルを高めます。

-v を使用して証明書を表示している場合、出力には、Issuer (発行者)、Serial Number (hex) (シリアル番号 (16 進数))、Subject (サブジェクト)、サブジェクト代替名 (Subject Alternative Name)、Validity period (有効期間) があります。また設定によっては、Extensions (拡張) (Key usage (鍵使用法)、Constraint (制約)、CDP、AIA、Policy OID (ポリシ OID) を含む)、Public key type (公開鍵の種類)、および Public key fingerprint (公開鍵の指紋) が含まれる場合もあります。このオプションを指定しないと、出力には Issuer (発行者)、Serial Number (hex) (シリアル番号 (16 進数))、および Subject (サブジェクト) が含まれます。

-v を使用して CRL を表示している場合は、出力に、破棄された証明書の一覧全体が含まれます。このオプションを指定しないと、出力には発行者と更新情報が含まれます。

例

証明書の拡張に関する全情報を含めて、指定された証明書の内容を表示するには

```
ssh-certview -v sample.crt
```

証明書 cacert.pem、証明書要求 cacert.pem.p10、および証明書破棄ファイル example.revoke.crl の内容を表示するには

```
ssh-certview cacert.pem cacert.pem.p10 example.revoke.crl
```

PKI Services Manager マップファイルに含めるために、指定された証明書からサンプル出力を抽出するには

```
ssh-certview -q -c cacert.pem
```

付 録 Q

ssh-certtool コマンドリファレンス

同義語

ssh-certtool [*options*] *action* [*arguments*]

その *action* の値は、 **pkcs10** (PKCS#10 証明書要求の作成) または **pkcs12** (PKCS#12 パッケージの作成) です。次に示すように、適用可能な引数は指定する操作によって異なります。

ssh-certtool [*options*] pkcs10 *subject* [*keyUsage*] [*extendedKeyUsage*]

ssh-certtool [*options*] pkcs12 [*file1*] ... [*fileN*]

これら 2 つのオプションのヘルプをそれぞれ表示するには、次のコマンドを使用できます。

```
ssh-certtool --help pkcs10
```

```
ssh-certtool --help pkcs12
```

説明

次を使用して (**ssh-certtool**)、PKCS#10 証明書要求や、秘密鍵と 1 つ以上の証明書を含む PKCS#12 パッケージを作成できます。

PKCS#10 証明書要求の作成

PKCS#10 ファイルを作成する一般的な構文は以下のとおりです。

ssh-certtool [*options*] pkcs10 *subject* [*keyUsage*] [*extendedKeyUsage*]

注意: **req** は、次の同義語として対応しています (**pkcs10**)。

サブジェクトとして指定した値は、証明書の Subject (サブジェクト) フィールドを定義します。サブジェクト名は必須です。RFC2253 で規定されている識別名構文を使用します。Subject 要素 (RDN) を区切るにはカンマを使用します。RDN は、標準の省略形 (CN) または OID (2.5.4.3) を使用して指定できます。サブジェクト名に埋め込まれた空白が含まれる場合は、引用符が必要です。

注意: subject 引数の SubjectName 要素は、大文字 (CN、DC、OU、O、C) で指定する必要があります。例えば、"CN=Steve Kille,O=Isode Limited,C=GB" のようになります。

生成された証明書要求のファイル名は、 **-o** オプションで指定された接頭語に `.pkcs10` を付加して作成されます。生成された秘密鍵の既定のファイル名 (**-o** が指定されていない場合) は、次になります (`output.pkcs10`)。

既存の秘密鍵を使用して要求を作成するには、 **-p** で鍵を指定します。要求の新しい秘密鍵を生成するには、次を省略します (**-p** オプション)。既定では、 **ssh-certtool** で 2048 ビットのRSA 鍵が作成されます。鍵の種類を指定するには、次の種類を使用します (**-N**)。鍵のサイズを指定するには、次を使用します (**-b**)。生成された秘密鍵のファイル名は、**-o** オプションで指定された接頭語に .ssh2 を付加して作成されます。生成された秘密鍵の既定のファイル名 (**-o** が指定されていない場合) は、次になります (`output.ssh2`)。同じ名前の鍵がすでに存在する場合は、既存の鍵を上書きするかどうか確認を求められます。上書きしないことを選択すると、戻りコード 0 が返されて **ssh-certtool** が終了します。

オプションのフラグを使って、keyUsage および extendedKeyUsage フィールドを設定できます。項目を区切るにはカンマ、空白、またはタブを使用します。すべての Key Usage (鍵使用法) および Extended Key Usage (拡張鍵使用法) フラグは、PKCS#10 要求内で Critical (重要) とマーク付けされます。有効な keyUsage フラグは、digitalSignature、nonRepudiation、keyEncipherment、dataEncipherment、keyAgreement、keyCertSign、cRLSign、encipherOnly、および decipherOnly です。この引数を省略すると、既定で digitalSignature および keyEncipherment フラグが設定されます。有効な extendedKeyUsage フラグは anyExtendedKeyUsage、serverAuth、clientAuth、codeSigning、および emailProtection です。既定で extendedKeyUsage フラグは設定されません。

PKCS#12 パッケージの作成

PKCS#12 パッケージを作成する一般的な構文は以下のとおりです。

ssh-certtool [*options*] pkcs12 [*file1*] ... [*fileN*]

これにより、 *file* 引数から読み込まれた 1 つの秘密鍵と複数の証明書を含む PKCS#12 パッケージファイルが作成されます。PKCS#12 パッケージファイルは、秘密鍵とすべての証明書が含まれる 1 つの金庫を保持します。生成されたパッケージファイル名は、 **-o** オプションで指定された接頭語に .p12 を付加して作成されます。生成された PKCS#12 パッケージの既定のファイル名 (**-o** が指定されていない場合) は `output.p12`です。PKCS#12 パッケージは HMAC によって保護されており、パッケージを作成する前に **ssh-certtool** によってパスフレーズの入力が求められます。秘密鍵が含まれる

- 引数は、パスフレーズ保護されていない PKCS#8 形式、ssh2 PEM 形式、または OpenSSH PEM 形式で読み込むことができます。鍵がパスフレーズで保護されている場合、 **ssh-certtool** によってパスフレーズの入力が求められます。証明書を含む - 引数は DER 符号化形式と PEM 符号化形式の両方で認識されます。

既定で個々の秘密鍵と証明書は、既定の PBE 保護方式を使用して PKCS#12 出力ファイルに保存されます。鍵暗号化の既定の方式は pbeWithSHA1And3-KeyTripleDES-CBC です。安全な暗号化の既定の形式は pbeWithSHA1And40BitRC2-CBC です。\ **-z** オプションを使用すると、異なる PBE 保護方式を構成できます。

オプション

オプションには、1 文字の形式 (**--b** など) と、同様の意味を持つ記述語 (**----bits**) の両方があります。1 文字のオプションを以下に示します。同様の意味を持つ記述語を表示するには、**--h** コマンドラインオプションを使用してください。

-b *bits*

生成される鍵に使用される鍵サイズを指定します。既定では、RSA 鍵は 2048 ビットで、DSA 鍵は 1024 ビットです。DSA 鍵の値は 64 の整数倍でなければなりません。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

-c *comment*

秘密鍵ファイルに記述するコメントを指定します。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

-d *debug_level*

デバッグ出力を有効にします。1、2、3、または 99 を使用します。(4 ～ 98 の値も使用できますが、3 と同じになります。)

-f

FIPS モードを有効にします。このモードでは、 **ssh-certtool** を使用して生成されたすべての鍵が FIPS 仕様に準拠するよう強制され、作成するすべての PKCS#10 証明書要求に FIPS 仕様に準拠する鍵が含まれます。注意: このオプションを使用した場合、PKCS#12 パッケージに含まれる鍵に制限はありません。

-h [*action*]

コマンドオプションの簡単な概要を表示します。 操作 (**pkcs10** または **pkcs12**) を指定して、その操作に関する追加情報を表示します。

-N *algorithm*

鍵の生成に使用されるアルゴリズムを指定します。設定できる値は「rsa」および「dsa」です。既定値は「rsa」です。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

-o *output_file_prefix*

出力ファイルのファイル名の最初の部分を指定します。絶対パスを指定して、別の場所にファイルを生成することができます。既定値は「output」です。(ファイル名の接尾語はファイルの種類に応じて生成されます。PKCS#10 ファイルの場合は .pkcs10、PKCS#12 の場合は .p12、秘密鍵の場合は .ssh2 になります。)

-p *private_key*

証明書要求内で使用する秘密鍵を指定します。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

-P

秘密鍵を空のパスフレーズで保存します。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

--passphrase *passphrase*

秘密鍵のパスフレーズを指定します。 このオプションは PKCS#10 ファイルの作成でのみ有効です。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-z *Key=Value*

PKCS#10 要求の証明書オプション、および PKCS#12 パッケージの暗号化オプションを指定します。

PKCS#10 要求の場合、 *Key* は **DNS**、**Email**、**UPN**、または **IP** です。これらを使用して、証明書の SubjectAltName フィールドに対応する拡張子の値を設定します。これらの拡張は Critical (重要) とマーク付けされていません。等号の前と後を含め、このオプションには空白が含まれていてはなりません。ただし、値の中で名前に文字として空白文字が含まれている場合を除きます。IP の場合は、有効な IPv4 アドレスまたは IPv6 アドレスを指定します。複数の拡張子を構成できますが、拡張子の種類ごとに 1 つの値しか設定できません。複数の拡張子を指定するには、次を繰り返します (**-z** オプション)。例えば、次のように入力します。

```
-z Email=joe@attachmate.com -z IP=10.10.10.10
```

PKCS#12 パッケージの場合は、 *Key* が **KeyPBE** または **SafePBE** (大文字と小文字が区別されない) である必要があります。等号の前と後を含め、このオプションには空白が含まれていてはなりません。KeyPBE は鍵の暗号化および hmac 方式を設定し、SafePBE は安全な暗号化および hmac 方式を設定します。以下に値の一覧を示します。既定では、**KeyPBE** は PBE-SHA1-3DES です。既定では、**SafePBE** は PBE-SHA1-RC2-40 です。括弧内の長い名前でも同様に機能します。

None

PBE-SHA1-RC4-128 (pbeWithSHA1And128BitRC4)

PBE-SHA1-RC4-40 (pbeWithSHA1And40BitRC4)

PBE-SHA1-3DES (pbeWithSHA1And3-KeyTripleDES-CBC)

PBE-SHA1-2DES (pbeWithSHA1And2-KeyTripleDES-CBC)

PBE-SHA1-RC2-128 (pbeWithSHA1And128BitRC2-CBC)

PBE-SHA1-DES (pbeWithSHA1AndDES-CBC)

PBE-SHA1-RC2-40 (pbeWithSHA1And40BitRC2-CBC)

PBE-MD2-RC2-64 (pbeWithMD2AndRC2-CBC)

PBE-MD5-RC2-64 (pbeWithMD5AndRC2-CBC)

例

新規に生成された鍵を使用して PKCS#10 要求を作成するには

```
ssh-certtool -n RSA -z DNS=steves.dns.server.com
-z Email=steved@myorg.org pkcs10 CN=steved,O=myorg.org,OU=rsit,
C=US DigitalSignature,nonRepudiation ServerAuth,ClientAuth
```

PKCS#12 パッケージファイルを作成して、鍵と金庫の暗号化を指定するには

```
ssh-certtool -z keyPBE=default -z safePBE=PBE-SHA1-RC4-40
-ofile pkcs12 id_rsa.crt id_rsa
```

付 録 R

winpki および pkid コマンドリファレンス

PKI Services Manager サービスを構成、起動、および停止する場合や、証明書の有効性と許可される ID を確認する場合は、**winpki** (Windows 上) または **pkid** (UNIX システム上) を使用します。

同義語

Windows:
`winpki [command [command args]] [options...]`

UNIX:
`pkid [command [command args]] [options...]`

command = **start** | **stop** | **restart** | **reload** | **ping** | **validate <cert>**

options = [**-b** *path*] [**-c** *cert*] [**-d** *level*] [**-f** *file*] [**-h**] [**-i**] [**-k**]

[**-m** *path*] [**-p**] [**-o** *key=value*] [**-t** *host*] [**-u** *user*] [**-V**] [**-w**]

コマンド

start

サービスを起動します。

stop

サービスを停止します。

restart

サービスを停止して再起動します。

reload

サービスを停止せずに構成を再度読み込みます。 構成ファイルを再度読み込むと、証明書と CRL のダウンロードに使用される内部のメモリ内キャッシュがクリアされます。証明書と CRL の有効期間はキャッシュによって維持されます。ただし、発行元が証明書や CRL を期限切れ前に更新した場合は、有効期間を手動でクリアする必要がある場合があります。 注意: 再読み込みすればほとんどの設定が使用可能になりますが、一部の設定では再起動が必要です。

ping

サービスの状態およびサービスで使用されるポートを表示します。

validate *certificate*

証明書を検証し、オプションで許可されている ID に関する情報を提供します。サービスを実行している必要があります。例えば、sample.crt が有効であるかどうかを判別するには、以下のようになります (UNIX 構文)。

```
pkid validate sample.crt
```

次を使用します (**-u**、 **-t**、または **-w**)。指定された証明書で許可される ID に関する情報を取得するには、証明書名の後にこれを使用します。例えば、ユーザ joe が sample.cer を使用して認証可能かどうかを判別するには、以下のようになります (Windows 構文)。

```
winpki validate sample.cer -u joe
```

オプション

-b *path* **--baseDir** *path*

PKI Services Manager 構成に使用されるデータディレクトリを指定します。

-c *cert* **--cert** *cert*

指定された証明書を検証します。このオプションは、サービスが実行中でない場合に使用できます。サービスの実行時に証明書を検証する場合は、 **validate** コマンドを使用します。

-d *level* **--debug** *level*

ログに送信される情報量を指定します。 使用可能な値は、 「error」、「warn」、「info」、「debug」、および「trace」です。 既定は「error」です。

-f *file* **--config_file** *file*

既定以外の構成ファイルを使用して起動します。

-h **--help**

コマンドオプションの簡単な概要を表示します。

-i **--init**

このオプションが必要とされることはほとんどありません。PKI Services Manager は、このオプションによる初期化後、サーバ用の鍵ペアを作成し、ユーザデータのディレクトリおよびファイルを作成します。初期化は、UNIX システムへのインストール時および Windows システムでの初回実行時に自動的に行われます。使用しているシステムがすでに初期化されている場合は、このオプションを使用しても適用されません。注意: 既存の鍵 (`pki_key` および `pki_key.pub`) を削除してからこのオプションを使用することで、新規の鍵を作成することができます。この場合、既存の構成ファイルは影響を受けません。

-k **--check-config**

構成ファイルおよびマップファイルにエラーがないか確認してから終了します。

-m *path* **--migrate** *path*

証明書認証設定を Reflection 構成ファイルから F-Secure 構成ファイルへ移行します。ここで *path* でディレクトリを指定した場合、PKI Services Manager はそのディレクトリでサーバ (`sshd2_config`) とクライアント (`ssh2_config`) の構成ファイルを検索し、これらのファイルから設定を移行します。ここで *path* でファイルを指定した場合、PKI Services Manager は指定したファイルの設定を移行します。ファイルとディレクトリの両方でフルパス情報が必要です。注意: 移行先のフォルダ内の `pki_config` ファイルにすでにトラストアンカが構成済みである場合、移行は発生しません。 これは、すでに構成済みの変更が移行によって上書きされないようにするためです。

設定は、次に移行されます。 `pki_config` および `pki_map` ファイル (PKI Services Manager が使用)。ここで **-b** スイッチを使用すると、移行した設定を含むファイルが、指定されたディレクトリに作成されます。このスイッチを省略すると、ファイルは既定の PKI Services Manager構成ディレクトリに作成されます。

移行ログは、次に作成されます。 `logs` ディレクトリ (PKI Services Managerデータディレクトリにあります)。既定で、このログには、「info」レベルのデータが記録され、エラーまたは警告が発生したかどうかが表示されます。このレベルは、次を使用して高めることができます (**-d**)。

-o *key=value* **--option** *key=value*

構成ファイルのキーワードを使用して構成できる任意のオプションを設定します。この方法で構成されたオプションは、構成ファイルの設定を置き換えます。キーワードの一覧とその意味については、次を参照してください (pki_config)。代替構文を以下に示します。スペースを含む式を囲むには、引用符を使用します。

```
-o key1=value
-o key1="sample value"
-o "key1 value"
-o key=value1,value2
-o key="value1, value2"
```

複数のオプションを構成するには、複数の **-o** スイッチを使用します。

```
-o key1=value -o key2=value
```

-p **--showkey**

公開鍵の指紋を表示し、フルパスと鍵の名前を示します。

-t *Host* **--hostName** *Host*

このオプションは、**validate** コマンドに続く証明書名の後に使用します。PKI Services Manager はマップファイルを読み込んで、指定されたホストが、検証対象のホスト証明書で許可される ID であるかどうかを報告します。

-u *user* **--userID** *user*

このオプションは、**validate** コマンドに続く証明書名の後に使用します。PKI Services Manager はマップファイルを読み込んで、指定されたユーザが、検証対象のユーザ証明書で許可される ID であるかどうかを報告します。 サーバ名 (user@server 形式) を指定している場合、PKI Services Manager は、指定されたサーバに対してユーザの認証が許可されるかどうかを報告します。 ユーザ名のみを指定した場合、PKI Services Manager は、ホスト固有の条件について確認せずに、この証明書でユーザの認証が許可されるかどうかをテストします。

-V **--version**

製品名とバージョンを表示します。

-w [*host*] **--whoAmI** [*host*]

このオプションは、**validate** コマンドに続く証明書名の後に使用します。PKI Services Manager は ID マップファイルを読み込んで、認証対象の証明書で許可されるすべての ID の一覧を返します。 このオプションの後にサーバ名を指定した場合は、返される一覧が、そのサーバへの接続で許可されるユーザに制限されます。サーバ名が指定されていない場合、PKI Services Manager はホスト固有の条件について確認しません。

付 録 S

pkid_config 構成ファイルリファレンス

Reflection PKI Services Manager コンソールでは、設定を構成ファイルに保存できます。また、この構成ファイルを手動で表示および編集することもできます。既定のファイルの場所は以下のとおりです。

- UNIX
  `/opt/attachmate/pkid/config/pki_config`
- Windows Server 2008:
  `\ProgramData\Attachmate\ReflectionPKI\config\pki_config`

ファイル形式

構成ファイルは、キーワードと、その後に続く値から構成されます。 キーワードと値は、タブまたはスペースで区切るか、またはスペースと1 つの「=」で区切ることができます。 シャープ記号 (#) で開始する行はすべてコメントです。 空の行は無視されます。一部のキーワードは複数回指定でき、これらの設定は累加的に適用されます。 設定の変更は、設定を再読み込みするかサービスを再起動するまで有効になりません。 (再起動が必要な場合は、キーワードの説明にその情報が付与されます。)

ファイルには、すべての検証クエリーに適用される設定を保持するグローバルセクションが含まれます。 また、証明書固有の設定を構成するスタンザを作成することもできます。 **TrustAnchor** キーワードは各トラストアンカスタンザの開始部分にマークを付けます。 **TrustAnchor** キーワードの下の設定はそのトラストアンカのみに適用されます。スタンザは次の **TrustAnchor** キーワードに達すると終了します。

一部の設定は、トラストアンカスタンザの外部で構成する必要があります。 これらの設定はすべての検証クエリーに適用されます。 グローバルとスタンザ内の両方で設定に対応している場合、トラストアンカスタンザ内の値がグローバルな値に優先します。

キーワード

AllowClientStats

PKI Services Manager で、クライアントによる PKI Services Manager 実行時統計の要求を可能にするかどうかを指定します。このキーワードをスタンザ外部で 1 回構成します。 許可される値は「yes」および「no」です。 既定は「yes」です。

AllowMD5InFipsMode

FIPS モードが有効な場合でも、MD5 ハッシュを使用して署名された証明書を許可します。このキーワードをスタンザ外部で 1 回構成します。 許可される値は「yes」および「no」です。 既定は「yes」です。 この設定を変更した場合は、サービスを再起動する必要があります。

AllowVers1

PKI Services Manager で、トラストアンカとしてバージョン 1 証明書を許可するかどうかを指定します。注意: この設定の値に関係なく、中間証明書はバージョン 3 でなければなりません。 このキーワードをスタンザ外部で 1 回構成します。 許可される値は「yes」および「no」です。 既定は「no」です。

AllowWhoAmI

PKI Services Manager を使用した証明書の検証時に、PKI Services Manager で、マッピングされた ID に関するクエリー (**-w** または **--whoAmI** を使用) をクライアントが実行可能かどうかを指定します。このキーワードをスタンザ外部で 1 回構成します。許可される値は「yes」および「no」です。 既定は「yes」です。

CertSearchOrder

PKI Services Manager が、証明書の検証に必要な中間証明書を検索する場所を指定するカンマ区切りの一覧。指定された場所が順番に検索されます。オプションは「local」、「certserver」、「aia」、および「windows」です。既定は「local、certserver」です(注意: 「windows」を選択した場合、PKI Services Manager は、現在のユーザ用にインストールされた証明書ではなく、ローカルコンピュータ用にインストールされた証明書のみを使用します。ローカルコンピュータ用の証明書を表示および管理するには、Microsoft 管理コンソールを使用します。証明書のスナップインを追加して、そのコンピュータのアカウントで証明書を管理するように構成します)。このキーワードをスタンザ外部で 1 回構成します。

CertServers

「certserver」が **CertSearchOrder** 一覧に含まれている場合に PKI Services Manager が中間証明書を取得できるサーバを指定しますHTTP サーバまたは LDAP サーバのいずれかを指定できます。 (例えば、(`ldap://certserver:10389` または `http://certserver:1080` など)。このキーワードは、スタンザ外部で複数回構成できます。 値は累積されます。

CRLServers

「crlserver」が **RevocationCheckOrder** 一覧に含まれている場合に PKI Services Manager が CRL (Certificate Revocation List) を取得できるサーバを指定しますHTTP サーバまたは LDAP サーバのいずれかを指定できます。 (例えば、(`ldap://crlserver:10389` または `http://crlserver:1080` など)。このキーワードは、スタンザ外部で、スタンザごとに複数回構成できます。 値は累積されます。

ClientDebugging

証明書の検証を要求しているアプリケーションが PKI Services Manager からのデバッグメッセージを要求および受信できるかどうかを指定します。このキーワードをスタンザ外部で 1 回構成します。 許可される値は「yes」および「no」です。 既定は「no」です。注意: これらのメッセージを表示するには、呼び出し側のアプリケーションに十分に詳細なデバッグレベルを設定する必要もあります。Reflection for Secure IT Windows サーバでは、[Protocol details] (プロトコルの詳細) 以上に指定し、Reflection for Secure IT UNIX クライアントおよびサーバでは、デバッグレベル 3 以上に指定します。

EnforceDODPKI

PKI マネージャが、米国防省 PKI 要件を満たす設定を実施するかどうかを決定します。許可される値は「yes」および「no」です。 既定は「no」です。 この設定が「yes」の場合は、条件「**FipsMode** = yes」、「**AllowMD2Certificates** = no」、「**AllowMD5InFipsMode** = no」、「**AllowVers1** = no」、「**CertSearchOrder** に「windows」が含まれていないこと」、「**RevocationCheckOrder** に 1 つ以上のオプションが指定され、「none」が含まれていないこと」が満たされないかぎり、サービスは起動しません。

ExplicitPolicy

PKI Services Manager がアプリケーションポリシを実施するかどうかを決定します。このキーワードは、スタンザ外部で、スタンザごとに 1 回構成できます。 許可される値は「yes」および「no」です。 既定は「no」です。 値が「yes」の場合は、**PolicyOID** キーワードを使用して実施される 1 つ以上のアプリケーションポリシを指定する必要があります。各アプリケーションポリシはポリシ ID (OID) によって指定されます。(注意: 提示される証明書、または信頼チェーン内の証明書によってポリシが必要となることがあります。)

FipsMode

FIPS 140-2 仕様に準拠するセキュリティプロトコルおよびアルゴリズムを実施します。許可される値は「yes」および「no」です。 既定は「yes」です。 このキーワードをスタンザ外部で 1 回構成します。 この設定を変更した場合は、サービスを再起動する必要があります。

KeyFilePath

Reflection PKI Services Manager の識別に使用される秘密鍵へのパスを指定します。パスが指定されないと、PKI Services Manager 構成ディレクトリに相対するパスまたはファイル名が使用されます。このキーワードをスタンザ外部で 1 回構成します。この設定は必須です。 If **KeyFilePath** が指定されていないか、鍵が存在しない場合、PKI Services Manager サービスは起動しません。既定は「pki_key」です。この設定を変更した場合は、サービスを再起動する必要があります。 PKI Services Manager では、設定の初期化時に鍵ペアが作成されますが、**ssh-keygen** (または別のツール) によって作成された鍵ペアを使用することもできます。RSA 鍵のみが許可されます。

ListenAddress

PKI Services Manager が検証要求についてリッスンするポートを指定します。構文は `host:port` になります。ホスト名を指定するには、IP アドレスか、ホスト名のいずれかを使用できます。IP アドレスには、IPv4 形式または IPv6 形式を使用できます。IPv6 アドレスは角括弧で囲む必要があります。例えば、次のようにします。`[::D155:AB63]:18081`既定値は `0.0.0.0:18081` で、これにより、サーバは、使用可能なすべてのネットワークアダプタを使用してポート 18081 をリッスンするよう構成されます。 この設定は必須です。 この設定を変更した場合は、サービスを再起動する必要があります。

LocalStore

ローカルストアは、証明書検証に必要とされる項目を保持するために使用されます。 使用している構成に応じて、ローカルストアには、信頼されるルート証明書、中間証明書、および CRL (Certificate Revocation Lists) が含まれることがあります。 ディレクトリまたはファイルを指定できます。ディレクトリを指定すると、指定されたディレクトリ内のすべてのファイルおよびすべてのサブディレクトリがそのストア内に含まれます。ファイルは、バイナリ、base64 符号化 X.509 証明書、または CRL のいずれかである必要があります。 このキーワードは、スタンザ外部で複数回構成できます。 値は累積されます。 この設定は必須です。

LogFacility

ログメッセージの出力場所を指定します。許可される値は「file」および「none」です。既定は「file」です。ログファイルは毎日作成され、PKI Services Manager データディレクトリ内にある `logs` という名前のディレクトリに保存されます。 このキーワードをスタンザ外部で 1 回構成します。 この設定を変更した場合は、サービスを再起動する必要があります。

LogLevel

ログに送信される情報量を指定します。使用可能な値は「error」、「warn」、「info」、「debug」、および「trace」です。ログでは、監査メッセージ (「[audit]」で示される) とデバッグメッセージ (「[debug]」で示される) の両方を保持できます。監視メッセージでは、成功および失敗した検証試行に関する情報が提供され、デバッグメッセージでは、問題解決に役立つ情報が提供されます。既定のログレベルは「error」です。このレベルでは、監査メッセージはログに送信されますが、デバッグメッセージは、通常 PKI Services Manager が適正に構成されていないことが原因で PKI Services Manager エラーが発生した場合にのみ送信されます。その他のオプションを使用すると、監査メッセージに加えて、より高レベルのデバッグメッセージの詳細が記録されます。このキーワードをスタンザ外部で 1 回構成します。

MapFile

PKI Services Manager マップファイルの場所を指定します。マップファイルは、有効な証明書を使用して認証が許可されるユーザまたはコンピュータを構成するために使用します。パスが指定されないと、PKI Services Manager 構成ディレクトリに相対するパスまたはファイル名が使用されます。この設定は必須です。 このキーワードは、スタンザ外部で、スタンザごとに 1 回構成できます。

MaxLogFiles

作成されるログファイルの最大数を指定します。 新しいログファイルが毎日自動的に作成されます。 最大数に達した場合、最も古いログが削除されます。 既定は 10 です。このキーワードをスタンザ外部で 1 回構成します。 この設定を変更した場合は、サービスを再起動する必要があります。

NetworkTimeout

LDAP、HTTP、または OCSP のいずれかを使用したネットワークダウンロードのタイムアウトを指定します。単位はミリ秒で、既定は 20000 です。このキーワードをスタンザ外部で 1 回構成します。このキーワードをスタンザ外部で 1 回構成します。

OCSPCertificate

OCSP 応答の署名を確認するために使用できる証明書を指定します。これは、OCSP 応答に署名者の証明書が含まれていない場合のみ必要とされます。値は証明書ファイルまたは証明書の Subject (サブジェクト) 値 (`OcspCertificate = 「CN = Secure CA, O = Secure Corporation, C = US」` など) のいずれかになります。Subject (サブジェクト) 値を使用する場合、証明書はローカルストア内になければなりません。このキーワードは、スタンザ外部で、スタンザごとに複数回構成できます。 値は累積されます。

OCSPResponders

「ocsp」が **RevocationCheckOrder** 一覧に含まれている場合に、証明書破棄の確認に使用される OCSP レスポンダのアドレスを指定します。レスポンダの識別には HTTP アドレスを使用します (例えば、`http://ocsp.myhost.com:1080`)。このキーワードは、スタンザ外部で、スタンザごとに複数回構成できます。 値は累積されます。

PolicyOID

ExplicitPolicy が「yes」に設定されているため、あるいは提示される証明書、または信頼チェーン内の証明書によってポリシが必要なために、アプリケーションポリシが有効な時に使用する、許可されるポリシ ID (OID) を指定します。**ExplicitPolicy** が「yes」の場合、指定された OID は、証明書チェーンにおける最後のポリシセット内の少なくとも 1 つの OID に一致する必要があります。値 2.5.29.32.0 を指定すると、任意のポリシ ID を使用できます。(注意: 既定値は「no-policy」です。**ExplicitPolicy** の設定を「yes」にした場合、適用される (1 つまたは複数の) ポリシに合わせて **PolicyOID** を変更する必要があります。**ExplicitPolicy** の設定を「yes」にし、**PolicyOID** の設定を「no-policy」にした場合、証明書は検証を通過しません。)このキーワードは、スタンザ外とスタンザ内の両方で複数回構成できます。 構成した値は累積されます。

RevocationCheckOrder

証明書破棄の確認に使用されるソースおよびそれらのソースの確認順序を指定するカンマ区切りの一覧。オプションは「ocsp」、「cdp」、「crlserver」、「local」、および「none」です。既定は「local」です。注意: 「none」だけを指定すると、破棄の確認は行われません。ほかのオプションとともに「none」を指定すると、PKI Services Manager は、指定されたオプションの順序内で「none」に達するまで、それらのオプションを使用して破棄状態の確認を試みます。この時点で証明書の破棄状態が不明な場合は、認証が許可されます。このキーワードは、スタンザ外部で、スタンザごとに 1 回構成できます。

StrictMode

証明書の検証時に厳格な確認ルール (RFC 3280 で規定) が使用されるかどうかを指定します。 多くの証明書は、厳格な確認をパスできません。 許可される値は「yes」および「no」です。 既定は「no」です。 このキーワードは、スタンザ外部で、スタンザごとに 1 回構成できます。

TrustAnchor

Reflection for Secure IT が検証する証明書の信頼チェーン内で最終的な信頼ポイントとして使用する証明書を指定します。これは、中間 CA 証明書、ルート CA 証明書、または自己署名付き証明書 (自己の検証のみが可能) のいずれかであり、ユーザ証明書またはホスト証明書であってはなりません。値は証明書のファイル名、または証明書内に定義された [Subject] (サブジェクト) フィールドの内容 (`TrustAnchor = 「CN = Secure CA, O = Secure Corporation, C = US」` など) のいずれかになります。フルパス情報を含めて、証明書のファイル名を指定した場合、トラストアンカは、**CertSearchOrder** の構成方法に関係なく使用されます。フルパス情報を含めずに証明書のファイル名を指定した場合は、**CertSearchOrder** に「local」が含まれている必要があり、この場合、PKI Services Manager はその証明書についてローカルストア内を検索します。証明書の [Subject] (サブジェクト) フィールドの内容を指定した場合は、**CertSearchOrder** に「local」または「windows」 (およびその両方)が含まれている必要があり、PKI Services Manager はその証明書についてローカルストア内/Windows 証明書ストア内を検索します。この設定は必須です。複数のトラストアンカを構成するには、追加の **TrustAnchor** 行を指定します。

注意: ストア内の証明書の Subject (サブジェクト) 値を表示するには、Windows システムでは PKI Services Manager コンソールを使用し、UNIX システム上では ssh-certview(1) を使用します。

TrustAnchor 設定の下にあるキーワードはすべてスタンザを形成します。トラストアンカスタンザ内に構成される値は、そのトラストアンカ固有の値になります。

付 録 T

pki_mapfile Map File Reference

Reflection PKI Services Managerは、マッピングルールを使用して、証明書を 1 つまたは複数の許可される ID にバインドします。 通常、許可される ID はユーザまたはホストになります。 ユーザを正当に認証するには、検証される証明書内の情報を、許可されるユーザアカウントに結び付けるルールを定義する必要があります。 マッパには、証明書を名前にマッピングするための柔軟なオプションが用意されています。 許可される名前をルールに明示的に指定することも、ユーザ名やホスト名などの情報を証明書から抽出するルールを定義することもできます。 これらのオプションを使用すると、証明書ごとに個別のルールを作成する必要なく、ID を証明書にバインドすることができます。

既定のマップファイル名と場所は以下のとおりです。

- UNIX
  `/opt/attachmate/pkid/config/pki_mapfile`
- Windows Server 2008:
  `\ProgramData\Attachmate\ReflectionPKI\config\pki_mapfile`

注意: Windows システムでは、 Reflection PKI Services Manager サービスマネージャコンソールの **[Identity Mapper]** 画面を使用してマップファイルを変更できます。

ファイル形式

マップファイルは、キーワード設定とルールで構成されます。 ルールはそれぞれ一行からなり、他のルールとは独立しています。 ルールの形式は、以下のとおりです。

{*`Allowed-Identity`*} [*条件式*]

証明書が有効であると判明した後は、ルールが順番に (ルールの種類を基準に使用した後で、順序に従う) が処理されます。 証明書が、条件式に定義された要件を満たす場合 (または、ルールに条件が含まれていない場合)、そのルールに指定された、許可される ID の認証が許可されます。 最初の一致が検出された後は、追加のルールは適用されません。

マップファイル内で **RuleType** キーワードを使用して、証明書を提示するのがユーザであるか、ホストであるかに応じて異なるマッピング基準を適用することができます。
注意: ルールの種類は、ルールの処理順序を決定します。 ユーザ証明書の処理順序は、user-address、user、none の順番です。 ホスト証明書の処理順序は、host、none の順番です。 各ルールの種類内では、ルールは上から下へ順番に処理されます。

許可される ID セット

許可される ID セットは、ルールの必須の構成要素です。許可される ID は、定数値と、証明書から抽出した値の組み合わせを使用して指定できます。許可される ID のセットには、複数の定数値または抽出値、または両方の組み合わせを指定できます。
PKI Services Managerクライアントの識別情報マッピング要件は異なります。次に例を示します。Reflection for Secure IT サーバは、マップルールのドメインユーザ名を特定するための複数フォーマットに対応しています。Reflection for Secure IT ユーザマネージャは、すべての有効な証明書に対し、1 ユーザのみを許可します。詳細は、製品マニュアルから Reflection for Secure IT を使用して検証を構成する方法について参照してください。

定数値を使用して、許可される ID を定義する

定数値は、リテラル文字列です。値を区切るには、空白を使用します。(許可される ID に空白が含まれる場合は、その ID を引用符で囲みます。)例えば、以下のルールでは、任意の有効な証明書を使って root、joe、および fred smith の認証を許可するためにリテラル文字列を使用しています。

```
{ root joe "fred smith" }
```

注意: PKI Services Manager が、ルールに定義された条件に証明書が適合すると判断したら、ルールの処理は停止します。上記のいずれの例でも、条件が定義されていません。つまり、ルールは、いずれかの有効な証明書に適用され、後続のルールは処理されません。同種のルールを作成する場合は、同じルール内に許可されるすべての ID を含める必要があります。

---

2 つのアスタリスクは、単独で使用された場合 ({ ** })、許可される ID セットを定義する時のワイルドカードとして機能します。このオプションは、テスト用途で使用できますが、それ以外の場合は、使用にあたって特別な注意が必要です。このワイルドカードをユーザルールで使用した場合は、サーバ上の任意のユーザアカウントに対して、有効な証明書を提示するすべてのユーザの認証が許可されます。これは、パスワードを要求せずに、root、管理者、またはパワーユーザの各権限を持つアカウントへのアクセスを許可することになるので、セキュリティ上の重大なリスクにつながります。このワイルドカードをホストルールで使用した場合、有効な証明書があるサーバはすべてクライアントに受け入れられます。ワイルドカードを使用することを選択した場合は、以下に示すように、ほかのオプションを使ったアクセスの制限を検討してください。

- 制御している認証局によって署名された証明書である場合のみワイルドカードを使用します。
- 条件により厳しい制限が設けられているルールでのみワイルドカードを使用します。
- サーバ固有のユーザルールでのみワイルドカードを使用します。(次のルール **RuleType** が **user-address**)。
- ユーザアカウントへのアクセスをサーバ側で制限します。例えば、Secure Shell サーバ上では、sftp chroot jail を定義して、コマンドシェルやリモートコマンドのアクセスを許可しないようにすることができます。

証明書から抽出した値を使用する

認証用に提示された証明書の内容に基づき、許可される ID セットを作成するために、抽出した値を使用します。抽出値の前と後には「%」を付加する必要があります。例えば、UPN フィールドの Host 部分で指定されたホストによる認証を許可するには、以下のようになります。

```
{ %UPN.Host% }
```

リテラル文字列と抽出した ID を結合することもできます。(リテラル文字列を抽出した ID の前に追加したり、リテラル文字列の後に追加したりできますが、抽出した複数の値を結合して単一の ID を作成することはできません。)以下の例では、抽出されたユーザ ID に Windows ドメイン名を追加しています。

```
{ windomain\%UPN.User% }
```

注意: 抽出された ID が、結果が空であると評価された場合は、結合された文字列全体が空であるとみなされ、許可される ID のセットに含まれません。許可される ID のセット全体が空である場合、そのルールは失敗したとみなされ、次のルールで処理が続行されます。

指定できる証明書のフィールドは以下のとおりです。

Subject

証明書に定義される Subject (サブジェクト) フィールド。比較は (文字列の比較としてでなく) X.500 ルールに従って行われます。一致に成功するには、形式が、RFC 2253 に記述された規格に準拠している必要があります。この規格に準拠するために、Subject (サブジェクト) フィールドと Issuer (発行者) フィールドは共通名 (CN) で始まります (「CN = Secure CA, O = Secure Corporation, C = US」など)。UNIX システム上では、**ssh-certview** ユーティリティを使用して、この形式の Subject 値を取得できます。Windows システム上では、証明書ビューワの [Details] タブから Subject の内容をコピーして、エディタに貼り付け、改行文字をカンマに置き換えます。

Subject.CN

Subject (サブジェクト) フィールドの共通名部分 (存在する場合)

Subject.Email

Subject の電子メール属性部分 (存在する場合)

DNS

SubjectAltName の DNS 部分 (存在する場合)

IPAddress

SubjectAltName の IP アドレス部分 (存在する場合) (PKI Services Manager バージョン 1.2 以上)

UPN

OID 1.3.6.1.4.1.311.20.2.3 (UPN OID) を使用した、SubjectAltName フィールドの「otherName」表現 (存在する場合)

UPN.User

UPN フィールドの userID 部分

UPN.Host

UPN フィールドのホスト部分

Email

RFC 822 の規定に従った、SubjecAltName の表現

Email.User

Email の userID 部分

Email.Host

Email のホスト部分

SerialAndIssuer

証明書のシリアル番号 (16 進符号化) と、この形式での証明書の Issuer (発行者) フィールドの値

`serial_number  Issuer`

発行者とシリアル番号を区切るには空白を使用します。例えば、次のように入力します。

```
461D07A8 CN = Secure CA, O = Secure Corporation, C = US
```

Cert

これは証明書全体を示します。演算は **Equals** である必要があり、引数は証明書へのファイルパスでなければなりません。注意: マッパでは、Reflection PKI Services Managerで定義されている証明書ストアは使用されません。

subst

このオプションは、ルール内の条件式で **Regex** または **Extern** が使用されている場合に使用できます。

Regex の場合は、キャプチャグループ (丸括弧 () を使用して識別されている) を含む任意の正規表現と組み合わせて **subst** を使用します。正規表現に、指定された証明書フィールドの完全一致が含まれている場合、その表現内の最初のキャプチャグループの値によって、許可される ID セット内の %subst% が置き換えられます。

Extern の場合は、外部アプリケーションによって返される値のプレースホルダとして **subst** を使用します。

条件式

条件式が {Allowed-Identity} に準拠している場合、その条件式が真の場合のみ、許可される ID の認証が可能になります。条件式の使用はオプションですが、ほとんどの場合で使用することをお勧めします。条件式が含まれていないと、有効なすべての証明書で、許可される ID の認証が可能になります。

証明書が有効であると判明した後は、ルールが順番に (ルールの種類を基準に使用した後で、順序に従う) が処理されます。 証明書が、条件式に定義された要件を満たす場合 (または、ルールに条件が含まれていない場合)、そのルールに指定された、許可される ID の認証が許可されます。 最初の一致が検出された後は、追加のルールは適用されません。

条件式の構文は以下のとおりです。

`Field Operation Argument`

対象 *`Field`*には、対応している次の証明書フィールド (上記参照) のいずれかを指定します。Subject、Subject.CN、Subject.Email、DNS、IPAddress、UPN、UPN.User、UPN.Host、Email、Email.Host、SerialAndIssuer、Cert、または subst。

対象 *`Argument`*には文字列値を指定します。

対象 *`Operation`*には、以下のいずれか 1 つを使用します。

Equals

ここで *`Field`* 値と *`Argument`* 文字列が絶対的に等価であるかどうかを確認します。DNS、UPN、および Email オプションの場合、比較では大文字と小文字が区別されます。

Contains

ここで *`Field`* 値が、 *`Argument`* 文字列が絶対的に等価であるかどうかを確認します。DNS、UPN、および Email オプションの場合、比較では大文字と小文字が区別されます。

Regex

その *`Argument`* を正規表現として次に適用します (*`Field`*)。正規表現に *`Field`* の内容の完全一致が含まれている場合、条件は真になります。許可される ID のセットに文字列 **%subst%** が含まれている場合、Regex 一致の最初のキャプチャグループ (定義されている場合) が挿入されます。

Extern

外部アプリケーションを使用して条件をテストします。次を使用します (*`Argument`*)。アプリケーションをポイントします。外部アプリケーションによって返される値のプレースホルダとして、許可される ID セット内で %subst% を使用します。PKI Services Manager によって、外部アプリケーションに指定した *`Field`* 値が送信されます。外部アプリケーション内でのテストに成功した場合、処理は状態 0 で終了します。非 0 値が返された場合は、一致が失敗したことを示しています。

指定する *Field* 値が **Cert**である場合、PKI Services Manager は 2 つの引数を外部アプリケーションに渡します。1 番目の引数には証明書の内容が PEM 形式 (テキスト) で含まれます。2 番目の引数には、証明書のコピーを DER 形式 (バイナリ) で含む一時ファイルのパスが含まれます。外部アプリケーションが終了すると、DER 形式の一時証明書は PKI Services Manager によって削除されます。

条件式を含むルールのサンプル:

```
{ %UPN.Email% } Subject.CN Equals acme.com
```

```
{ joep } Subject Contains "Joe Plumber"
```

ルールの種類のスタンザ

ルールの種類を使用すると、検証される証明書が、ユーザ証明書であるか、ホスト証明書であるかに基づき異なるマッピング基準が適用されます。指定できる各種類の新規のスタンザを作成するには **RuleType** キーワードを使用します。スタンザは、次の **RuleType** キーワード、またはファイルの終わりに達すると終了します。形式は次のとおりです。

RuleType type

有効なルールの種類は以下のとおりです。

none

ルールは、ホスト証明書とユーザ証明書の両方に適用されます。

host

ルールは、ホスト証明書のみに適用されます。

user

ルールは、ユーザ証明書のみに適用されます。

user-address = *server*

ルールは、指定されたサーバに対する認証に使用されるユーザ証明書のみに適用されます。注意: PKI Services Manager は、user-address ルールの評価時に、ユーザの接続先のサーバのサーバ名 (DNS ホスト名ではない) を使用します。サーバ名は、ユーザ証明書の検証の要求時にサーバによって PKI Services Manager へ送信され、PKI Services Manager は、user-address ルールの適用時にその名前を使用します。送信されるホスト名を決定するには、Windows の DOS ウィンドウまたは UNIX の端末セッションから **hostname** コマンドを入力します。

例えば、サーバ acme へ接続しているユーザのみに適用されるルールを作成するには、以下のようになります。

```
RuleType user-address=acme
```

注意: ルールの種類は、ルールの処理順序を決定します。 ユーザ証明書の処理順序は、user-address、user、none の順番です。 ホスト証明書の処理順序は、host、none の順番です。 各ルールの種類内では、ルールは上から下へ順番に処理されます。

キーワード

DynamicFile

PKI Services Manager が、許可される ID について確認するたびにマップファイルを再び読み込むかどうかを指定します。 許可される値は「yes」および「no」です。既定は「no」です。

ExternTimeout

次のオプションを使用するルールのタイムアウトを設定します (**Extern** オプション)。既定値は 0 (ゼロ) であり、この場合はタイムアウトは設定されません。

RuleType

ルールの種類のスタンザの開始部分にマークを付けます。これは、証明書を提示するのがユーザであるか、またはホストであるかに応じて異なるマッピング基準を適用するために使用できます。許可される値は「user」、「host」、「none」、および「user-address = サーバ」です。既定は「none」です。

付 録 U

マッピングルールのサンプル

ルール	説明
`{ guest }`	条件が何も含まれていないため、有効なすべての証明書がユーザ「guest」にマッピングされます。これは既定のルールとして使用できます。この種のルールは、ほかのすべてのルールが先に処理されるようにルールの最後に配置する必要があります。
`{ fred.jones }  UPN.user Equals "fred"`	SubjectAltName の UPN 表現が存在し、そのユーザ部分が「fred」に等しい場合、許可される ID のセットは fred.jones になります。
`{ %UPN.user% } UPN.host Equals "acme.com"`	証明書に SubjectAltName の UPN 表現が存在し、そのホスト名部分が「acme.com」である場合は、UPN のユーザ名部分が、許可される ID のセットとして返されます。
`{ guest %UPN.user% }`	UPN が設定されている場合は、そのユーザ部分が (「guest」とともに)、許可される ID のセットに含まれます。UPN が設定されていない場合、許可される ID のセットは「guest」になります。条件が含まれていないため、このルールは有効なすべての証明書に適用されます。
`{ fred root } Subject.CN Contains "Fred Jones"`	証明書の CN に「Fred Jones」が含まれている場合、許可される ID のセットには「fred」と「root」の 2 つの値が含まれます。
`{ %subst% } Subject.CN Regex [a-zA-Z\.]*([0-9]+)`	許可される ID を、[Subject] フィールドの共通名 (CN) 部分に含まれる最初の数値文字列に等しくなるよう設定します。例えば、CN が「joe.smith.12345」である場合、許可される ID は「12345」に設定されます。
`{ elmer.foo.com } Subject.CN Contains "elmer"`	許可される ID を、短名「elmer」を含む証明書の完全修飾ドメイン名「elmer.foo.com」に設定します。
`{ bob } Cert Equals /temp/certs/bob_cert.crt`	受信した証明書を、ローカルに保存されている証明書と比較します。それらの証明書が等しい場合に、許可されている ID のセットが「bob」になります。
`{ %subst% } Cert Extern /bin/myapp`	PKI Services Manager によって2つの値がアプリケーション「/bin/myapp」に送信されます。1 番目の値には証明書

ルール	説明
	の内容が PEM 形式 (テキスト) で含まれます。2 番目の値には、証明書のコピーを DER 形式 (バイナリ) で含む一時ファイルのパスが含まれます。外部アプリケーションは、これらの形式のいずれも使用できるように構成できます。呼び出されたアプリケーションの終了コードが 0 に等しい場合、許可される ID は、返された結果に等しくなるよう設定されます。
`{ %UPN.User% } UPN Extern /bin/ldap-app`	この場合、外部アプリケーションの終了コード 0 が、その UPN が認証されたユーザであることの確認として使用されます。
`{ %Subject.CN% %DNS% }`	許可される ID のセットに、Subject.CN フィールドの内容か、SubjectAltName の DNS 部分のいずれかが含まれるよう設定します。
`{ windomain\%UPN.User% }`	指定された Windows ドメイン名に属するユーザの認証を、そのユーザ名が UPN ユーザ名と一致する場合に許可します。

付 録 V

RuleType スタンザを含むマップファイルのサンプル

```
RuleType user
# 以下のルールは、ユーザ証明書に対してのみ評価されます。
{ scott } Subject.CN Contains acme
{ joe } Subject.CN Equals acme
{ guest }

RuleType host
# 以下のルールは、ホスト証明書に対してのみ評価されます。
{ elmer.acme } Subject.CN Contains elmer

RuleType user-address=myserver
# 以下のルールは、myserver
# がユーザ証明書の検証を要求した場合にのみ評価されます。
{ good %subst% } Regex UPN "([A-Za-z0-9\.-])@[*.]"

RuleType none
# 「none」は RuleType が指定されていない場合の既定の値です。
# 「user」または「host」からルールが正しく適用されない場合は、
# このルールが評価されます。
{ good } SerialandIssuer contains 123 Subject.CN=foo
```

付 録 W

PKI 設定の移行

Reflection for Secure IT 6.x または F-Secure を使用して証明書認証を構成した場合は以下の情報を参照してください。証明書の設定の中には、引き続き Reflection for Secure IT, Client and Server for UNIX 設定ファイルで対応できる設定もあれば、Reflection PKI Services Manager 設定ファイルに移行する必要がある設定もあります。Reflection for Secure IT 6.x または F-Secure 設定ファイルから設定を移行するには、**pkid** コマンドと **-m** オプションを使用できます。

注意: **-m** オプションの詳細については、「**pkid** コマンドリファレンス 『169ページ』」を参照してください。

以下の表に、以前のバージョンの設定がどのように処理されるかについて概略を示します。**状況**列の項目は、現行バージョンの設定ファイルに以前のバージョンのキーワードがどのように適用されるかを示しています。これらの項目には以下のような意味があります。

- 対応: キーワードの意味は、以前のバージョンと変わりありません。
- 非推奨: キーワードは引き続き適用可能ですが、その意味は変わっている場合があります。
- 無視: キーワードは現在の Reflection for Secure IT 設定ファイルに適用されません。これらの設定は PKI Services Manager 設定ファイルに移行する必要があります。詳細は、移行ログを参照してください。
- 未対応: キーワードは現行バージョンの設定ファイルでは使用できません。このキーワードがファイル内に存在する場合は、関連性を持たず、エラーが発生します。

クライアントの設定

以前のバージョンのキーワード	状況	移行すべきかどうか	同等の PKI Services Manager キーワード
HostCA	非推奨	はい	TrustAnchor
HostCANoCRLs	非推奨	はい	TrustAnchor RevocationCheckOrder = none
HostCertNameCheck	対応	いいえ	--
LDAPServers	無視	はい	CertServers CRLServers (すべてのサーバが両方のキーワードに移行されます。)
LocalPKI	無視	はい	LocalStore
OCSPResponder	無視	はい	OCSPResponders
RevocationChecks	無視	はい	RevocationCheckOrder
RevocationCA	無視	はい	OcspCertificate

サーバの設定

以前のバージョンのキーワード	状況	移行すべきかどうか	同等の PKI Services Manager キーワード
HostCA	非推奨	はい	TrustAnchor
HostCANoCRLs	非推奨	はい	TrustAnchor RevocationCheckOrder = none
HostCertificateFile	対応	いいえ	--
DynamicMapFile	無視	はい	DynamicFile (このキーワードは `pki_mapfile` で構成されます。)
ExternalMapper	無視	はい	条件式内の Extern オプションを使用して、マップファイルのルール内で使用できます。
ExternalMapperTimeout	無視	はい	ExternTimeout (このキーワードは `pki_mapfile` で構成されます。)
LDAPServers	無視	はい	CertServers CRLServers (すべてのサーバが両方のキーワードに移行されます。)
LocalPKI	無視	はい	LocalStore
OCSPResponder	無視	はい	OCSPResponders
RevocationChecks	無視	はい	RevocationCheckOrder
RevocationCA	無視	はい	OcspCertificate
MapFile	無視	はい	MapFile
OcspMode	無視	はい	RevocationCheckOrder
PKI	無視	はい	TrustAnchor
PkiDisableCrls	無視	はい	RevocationCheckOrder =none
PkiIgnoreBasicConstraints	無視	はい	StrictMode
SocksServer	未対応	いいえ	--

付 録 X

PKI Services Manager の戻りコード

Reflection PKI Services Manager は、検証サービスを要求するアプリケーションに以下のコードを返します。

- コード 0：エラーなし。検証に成功しました。
- コード 1 ～ 10：**winpki** または **pkid** のいずれかに関するコマンドラインエラー。
- コード 11 ～ 19：ネットワークエラーまたはプロトコルエラー。
- コード 21 ～ 29：検証エラー。
- コード 31 ～ 39：マッパのエラー（証明書は有効だがマッピング不可能）。
- コード 41-49：CRL またはその他の破棄エラー。

コード	意味
0	エラーなし。
1	一般エラー。原因は不明です。
2	コマンドでの構文エラー。引数が不適切です。
3	PKI Services Manager はすでに実行中です。
4	構成ファイルのエラー。
5	コマンドの実行中にタイムアウトが発生しました。
6	ネットワークエラー (PKI Services Manager に接続できないなど)。
7	アクセスが拒否されました。ユーザに、コマンドを実行する権限がありません。
8	システムエラー。内部エラーです。エラーの詳細を確認するには、-d スイッチを指定して再実行してください。
9	移行または初期化エラー。移行のエラーログを参照してください。
11	呼び出し側のアプリケーションによって不明なコマンドが要求されました。
12	PKI Services Manager で例外が発生しました。詳細については、PKI Services Manager イベントログを参照してください。
13	PKI Services Manager に送信されたコマンドまたはパケットでの構文エラー。
14	コマンドが無視されました (現在使用されていません。内部エラー)。
15	処理エラー。PKI Services Manager に送信された証明書が正しく符号化されていません。
16	コマンドが失敗しました (コマンド: stop、reload、reconfigure)。
17	署名の不一致。送信側が、一致する鍵を使って署名していません。
18	形式エラー。ASN プロトコルの形式が適正に設定されていません。
19	PKI Services Manager が FIPS モードで動作しており、このモードでは証明書は有効ではありません。

コード	意味
21	証明書が無効です (期限切れ、署名なし、不正な鍵など)。
22	パスが不明です。発行側の証明書を見つけることができません。
23	証明書が破棄されました。
24	トラストアンカがありません。パスの終端が既知のトラストアンカではありません。
25	その他の検証エラー。ポリシまたはその他の制約を適用できません。
26	終端の証明書までのパス長が、CA によるパス長の制限を超えています。
27	証明書ポリシが無効であるか、有効な決定に一致しません。
28	証明書の署名が一致しません。
29	証明書または CRL で Critical (重要) と示された不明の拡張に遭遇しました。
31	要求された ID が、許可される ID に一致しません。
32	この証明書に対して許可される ID がありません (一致するマップが存在しません)。
33	呼び出し側のアプリケーションによって、照合用の ID が送信されていません (クライアント側のエラー)。
34	証明書は有効ですが、WhoAmI 処理が要求されています。
41	不明の CRL 処理エラー。
42	delta CRL のベースが不明です。
43	CRL が期限切れです。
44	署名を確認できないか、署名が不正です。
45	不明の CRL 拡張が、Critical (重要) とマーク付けされています。
46	CRL 内の IDP フィールドの不一致。
47	CRL が使用できません。

用語集

G

GSSAPI (Generic Security Services アプリケーションプログラムインタフェース)

プログラムにセキュリティサービスへのアクセスを提供するアプリケーションプログラミングインタフェースです。

K

Kerberos

信頼されたサードパーティを使用して TCP/IP ネットワーク上で安全な通信を実現するプロトコル。このプロトコルは、プレーンテキストのパスワードではなく、暗号化されたチケットを使用してより安全にネットワーク認証を行います。

M

MAC (メッセージ認証コード)

データが送信中に変更されていないことの確認に使用されます。MAC は、データと共有秘密鍵が含まれる任意の長さのパケットを使用して作成されたハッシュです。送信側と受信側は、共有鍵および合意したアルゴリズムを使用して、転送されたデータの各パケットの MAC を独自に計算します。メッセージが送信中に変更されている場合、別のハッシュ値になり、パケットは拒否されます。

P

PKCS

PKCS (Public Key Cryptography Standards) は RSA 研究所によって考案され、発表された、公開鍵の各種暗号法の互換性を維持するための標準のセットです。PKCS の各標準で、それぞれの暗号を使用するための仕様が規定されています。Reflection for Secure IT, Client and Server for UNIX では以下の PKCS 標準を使用します。

- PKCS#7 は署名やメッセージの暗号化に使用できます。署名を保存したり、(例えば、PKCS#10 メッセージの応答として) 書名を配布したりするためにも使用できます。PKCS#7 ファイルから署名を抽出するには、**ssh-keygen** を使用します。
- PKCS#10 は、認証局 (CA) への証明書の要求のために使用されます。PKCS#10 ファイルを作成するには、**ssh-certtool** を使用します。
- PKCS#12 は、証明書や関連の秘密鍵の保存と移送のために使用されます。通常、この形式のファイルは、*.pfx または *.p12 拡張子を使用します。Reflection for Secure IT は、この形式で保存された証明書や鍵を使用した認証に対応しています。

S

Secure Shell

リモートコンピュータへのログインとコマンドの実行を安全に行うためのプロトコル。これは、Telnet、FTP、rlogin、あるいは rsh の代わりとなる安全な方法です。Secure Shell 接続では、ホスト (サーバ) とユーザ (クライアント) の両方の認証が必要です。また、ホスト間の通信はすべて暗号化された通信チャネルを介して行う必要があります。また、Secure Shell では X11 セッションまたは指定の TCP/IP ポートを、安全なトンネルを介して転送することもできます。

ソケット

ホスト名 (IP アドレスまたは DNS 名) とポート番号の組み合わせです。これは、クライアントアプリケーションが通信のエンドポイントとして使用する一意の識別子となります。

データの整合性

データが元のソースから変更されていない保証です。データの整合性を維持する方法は、データが誤って、または悪意を持って変更、改ざん、破壊されていないことを保証するように設計されています。

デジタル署名

送信されたメッセージの信頼性と整合性の確認に使用されます。通常、送信者は公開/秘密鍵のペアのうち秘密鍵を保有し、受信者は公開鍵を保有します。署名を作成するには、送信者はメッセージからハッシュを計算し、この値を自らの秘密鍵で暗号化します。受信者は、送信者の公開鍵を使用して署名を復号化し、受信したメッセージのハッシュを独自に計算します。復号化した値と計算した値が一致した場合、受信者は、送信者が秘密鍵の保有者であり、メッセージが送信中に改ざんされていないことを信頼します。

トラストアンカ

証明書の信頼チェーン内で最終的な信頼ポイントとして使用可能な証明書。注意: PKI Services Manager は、明示的に PKI Services Manager で使用するように構成されているトラストアンカのみを使用して証明書を検証します。トラストアンカは、ルート CA 証明書、中間 CA 証明書、または自己署名付き証明書 (自己の検証のみが可能な証明書) を使用して構成できます。

パスフレーズ

パスフレーズはパスワードに類似していますが、一連の語句、句読点、数字、空白、任意の文字列を組み合わせたフレーズを使用できる点が違います。パスフレーズは、秘密鍵や鍵エージェントなどの保護されたオブジェクトへのアクセスを制限して、セキュリティを向上させます。

ハッシュ

「メッセージダイジェスト」と呼ぶこともあり、ハッシュまたはハッシュの値は可変長のデジタルデータから生成される固定長の数値です。ハッシュは元のデータよりもかなり小さく、計算によって生成されますが、別のデータから同一のハッシュを作成することは統計的にできないようになっています。

ポート転送

安全でないトラフィックを安全な SSH トンネルを介してリダイレクトする方法です。ポート転送には、ローカルとリモートの 2 種類があります。ローカルポート転送 (発信ポート転送) は、指定されたローカルポートから送信された発信データを、安全なチャネルを介して、指定されたリモートポートに送信します。クライアントアプリケーションとサーバとの間でデータを安全に交換するには、関連サーバを実行するコンピュータにクライアントを直接接続するのではなく、リダイレクトされるポートに接続するように構成します。リモートポート転送 (受信ポート転送) は、指定されたリモートポートからの受信データを、安全なチャネルを介して、指定されたローカルポートに送信します。

漢字

暗号

暗号とは暗号化アルゴリズムのことです。選択した暗号によって、Secure Shell 接続の確立完了後に送信されるデータの暗号化に使用される数学アルゴリズムが決定されます。

暗号化

暗号化とは、暗号すなわち秘密のコードを使用してデータを加工し、許可されたユーザ以外には解読できないようにすることです。暗号化されていないデータに比べ、暗号化されたデータははるかに安全です。

公開鍵と秘密鍵

公開鍵と秘密鍵は、データの暗号化または解読に使用される暗号鍵のペアです。公開鍵で暗号化されたデータは、秘密鍵を使用した場合のみ解読できます。また、秘密鍵で暗号化されたデータは、公開鍵を使用した場合のみ解読できます。

正規表現

正規表現は、1 つ以上の一致する文字列を記述する文字列です。よく *regex* と省略されます。正規表現内では、一部の文字は事前に定義された意味を持ち、何が一致と見なされるかを決定します。たとえば、正規表現「t.*t」は、文字 *t* で始まり、かつ終わるすべての単語に一致します。一方、正規表現「text」はそれ自体のみに一致します。

帯域幅

ネットワークでのデータ転送率で、チャネル経由で転送できる情報の最大量 (Kbps または Mbps) を示します。

遅延

操作が開始されてから効果が出るまでの待ち時間です。ネットワークでのデータパケットの受信の遅れにはいくつかの原因があります。たとえば転送メディアや送受信ポイント間にあるネットワークデバイスの数などです。通常、ワークステーションとホスト間の物理的な距離が長くなるほど、遅延が発生する可能性が高くなります。

認証

通信相手の身元を確実に確認する処理。身元の確認は、パスワードなどの既知の情報、または秘密鍵やトークンなど所有しているもの、指紋などの固有の情報を使用して行います。

索引

X

あ

い

き

く

さ

し

て

と

は

ふ

ほ

ゆ

り

ろ

漢字