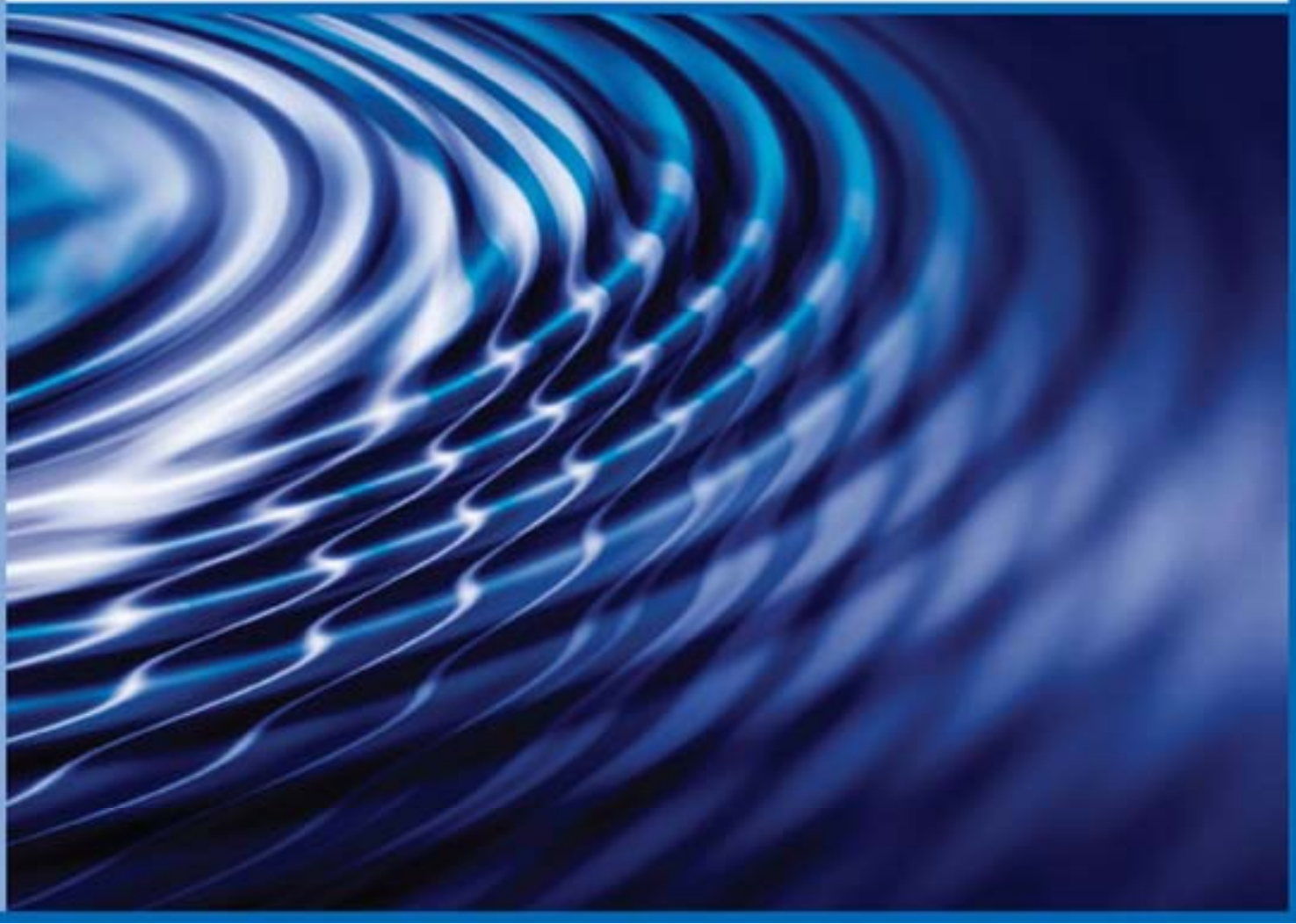
UNIX Client および Server

ユーザガイド

Attachmate®

Reflection®

for Secure IT



RSIT7.1UCSUG1.0J

ユーザガイド

Reflection for Secure IT

UNIX Client および Server

バージョン 7.1

Printed on 24 July, 2009

目次

Attachmate Corporation
1500 Dexter Avenue North
Seattle, WA 98109
USA
+1.206.217.7100
http://www.attachmate.com

第 1 章

インストール

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。

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)

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。

Reflection for Secure IT サーバは、上記のすべての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 (以下の注記を参照)
- sftp-server (サーバによって使用されるファイル転送サブシステム)

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。

注記: 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。

システム要件

Reflection for Secure IT UNIX Client および Server は以下のプラットフォームに対応しています。

- HP-UX 11i v1 (PA-RISC)
- HP-UX 11i v2 (PA-RISC および Itanium)
- HP-UX 11i v3 (Itanium)
- IBM AIX 5.2 (IBM Power5)
- IBM AIX 5.3 (IBM Power5)
- IBM AIX 6.1 (IBM Power5)
- Red Hat Enterprise Linux 3 (Itanium、Intel x86、および x86-64)
- Red Hat Enterprise Linux 4 (Itanium、Intel x86、x86-64、および zSeries 64)
- Red Hat Enterprise Linux 5 (Itanium、Intel x86、および x86-64)
- Sun Solaris 8 (SPARC)
- Sun Solaris 9 (SPARC)
- Sun Solaris 10 (SPARC、Intel x86、および x86-64)
- SUSE Linux Enterprise Server 9 (Itanium、Intel x86、x86-64、および S/390)
- SUSE Linux Enterprise Server 10 (Intel x86 および x86-64)

追加要件

- すべての Itanium システム (HP-UX、Red Hat Enterprise Linux、および SUSE Linux Enterprise Server) では、ライブラリ `libunwind` を必要とします。
- SPARC 搭載の Solaris では、Ultra SPARC を必要とします。
- PA-RISC 搭載の HP-UX 11i v1 では、以下のパッチを必要とします。
  - PHCO_28605 s700_800 11.11 libnss_files 累積パッチ
  - PHCO_31923 s700_800 11.11 libc 累積ヘッダファイルパッチ
  - PHCO_34275 s700_800 11.11 libc 累積パッチ
  - PHKL_34805 s700_800 11.11 JFS3.3 パッチ; mmap
  - PHSS_33033 s700_800 11.11 ld(1) およびリンカツール累積パッチ
- IBM AIX 5.3 では、保守レベル 5300-5 を必要とします。
- Kerberos 認証については、「*Kerberos システム要件* 『42 ページ』」を参照してください。

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、以前のリリース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。ファイルおよびディレクトリの保護が十分でないと、公開鍵認証は失敗します。詳細については、「*公開鍵認証におけるファイルおよびディレクトリの権限* 『48 ページ』」を参照してください。

注記: **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。最大限のセキュリティレベルを確保するために、現在、この設定は既定で有効になっています。一部のファイル権限は、この設定が無効になっている場合でも実施されます。

---

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、以下の注記を参照)。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。

注記:

- 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 へのインストール

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 使用しているコンピュータにインストールパッケージファイルをコピーして、そのファイルが格納されてい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-client-7.1.0.999-i386-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`

Linux 上の既定外の場所へのインストール

rpm --relocate オプションを使用して、インストールファイルの新規のターゲット場所を指定できます。以下の 2 つの変更を指定できます。

- 既定で `/etc/ssh2` にインストールされる構成ファイルおよび鍵の新規のターゲット場所を指定します。
- 既定で `/usr` にインストールされるバイナリおよびマニュアルページの新規のターゲット場所を指定します。

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。

非標準の場所にインストールするには

1 ターゲットディレクトリを作成します。

2 **rpm --relocate** オプションを使用してターゲットディレクトリを指定します。一般的な構文は以下のとおりです。

```
rpm --install --relocate /usr=PrefixDir --relocate /etc/ssh2=SysConfDir
package_file.rpm
```

例えば、以下のようになります。

```
rpm --install --relocate /usr=/opt/rsit --relocate /etc/ssh2=/opt/rsit/etc
rsit-server-7.1.0.999-i386-rhel.rpm
```

注記:

- **--relocate** 変更は、上記で説明したインストール以外では使用しないでください。その他の変更を使用すると、インストールしても使用できない状況になる可能性があります。
- 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。

Sun Solaris へのインストール

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 使用しているコンピュータにインストールパッケージファイルをコピーして、そのファイルが格納されてい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-7.1.0.999-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、以下のようになります。

```
pkgadd -d rsit-client-7.0.0.999-sparc-solaris10.pkg
```

注記: 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』」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 RSITsshs`

Sun Solaris 上の既定外の場所へのインストール

Reflection for Secure IT を既定外の場所にインストールするには、応答ファイル (インストーラパッケージに情報を提供するテキストファイル) を作成し、PREFIX 変数を使用してインストール用のターゲットディレクトリを識別します。PREFIX 変数は以下のように適用されます。

- 既定で `/etc/ssh2` にインストールされる構成ファイルおよび鍵が `$PREFIX/etc/ssh2` に移動されます。
- 既定で `/usr` にインストールされるバイナリおよびマニュアルページが `$PREFIX` に移動されます。

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。

非標準の場所にインストールするには

1 ターゲットディレクトリを作成します。

2 インストールをターゲットディレクトリ (この例では `/opt/rsit`) にリダイレクトする応答ファイル (この例では `rsp`) を作成します。

```
echo "PREFIX=/opt/rsit" > rsp
```

3 インストール時に移動情報を提供するには **pkgadd -r** オプションを使用します。

```
pkgadd –r rsp -d rsit-server-7.1.0.999-x64-solaris10.pkg
```

注記: 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。

HP-UX へのインストール

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 使用しているコンピュータにインストールパッケージファイルをコピーして、そのファイルが格納されてい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-7.1.0.999-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-7.1.0.999-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
```

注記: 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 へのインストール

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 使用しているコンピュータにインストールパッケージファイルをコピーして、そのファイルが格納されてい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-7.1.0.999-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。

注記: 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

注記: 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にある `logs` ディレクトリに作成されている移行ログを確認します (既定で、このログには「info」レベルのデータが記録されます。このレベルは **-d** を使用して高めることができます)。

注記: 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必要がある場合は、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。このアプリケーションは追加の料金なしで提供されています。

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 使用しているコンピュータにインストールパッケージファイルをコピーして、そのファイルが格納されているディレクトリに移動します。

3 **gzip** を使ってパッケージを圧縮解除します。

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

例えば、以下のようになります。

```
gzip -d pkid_1.0.0.999-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
```

これにより、パッケージ名をベースとするディレクトリが作成されます。例えば、以下のようになります。

```
pkid_1.0.0.999--i386-solaris/
```

5 上記のディレクトリに切り替えます。例えば、以下のようになります。

```
cd pkid_1.0.0.999-i386-solaris
```

6 インストールスクリプトを実行します。

```
./install.sh
```

7 インストールの場所を指定するよう求められます。既定の場所 (推奨値) を使用するには、これらのプロンプトに対して [Enter] キーを押します。

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。既定のパスは以下のとおりです。

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

第 2 章

基本的な操作

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
```

注記: **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

注記: 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 Sun 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。

- Sun 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
```

- Sun 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
```

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、「*ホスト公開鍵のフィンガープリントの表示* 『36 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、(`$HOME/.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* 『35 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* 『135 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* 『137 ページ』」を参照してください。

双方向の sftp セッションを開くには

1 リモートホストに接続します。例えば、次のようになります。

```
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。

正常に接続が確立されると、以下のプロンプトが表示されます。

```
sftp>
```

2 次のいずれかを実行します。

目的	使用するコマンド
指定できるコマンドの一覧を表示する	**help**; 例えば、次のようになります。 `sftp> help`
指定できるコマンドの詳細を表示する	**help** *コマンド*; 例えば、次のようになります。 `sftp> help put`
ファイルを転送および管理する	*指定できるコマンド* 『137 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、次のようになります。 `sftp> put demo`
セッションを終了する	**quit**; 例えば、次のようになります。 `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* 『18 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

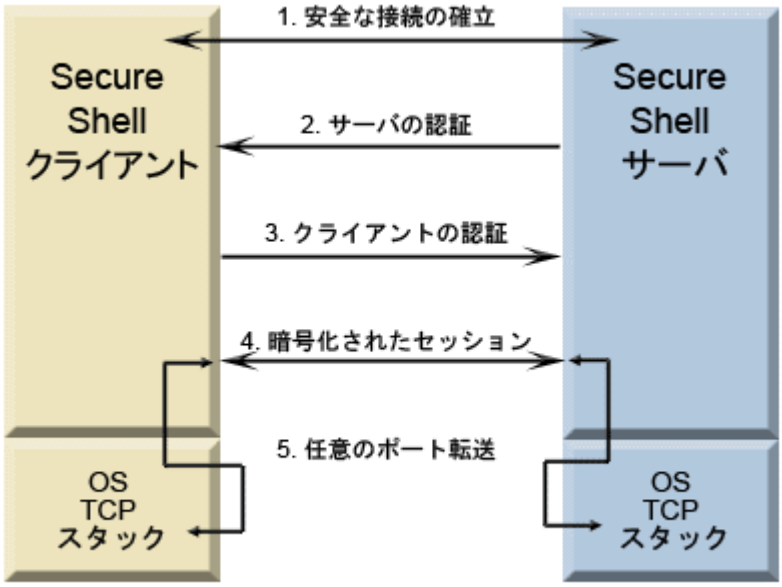
- 開始するには、以下のような例を使用します。

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

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* 『131 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。 追加情報については、「*データ保護* 『29 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。 追加情報については、「*サーバ認証* 『33 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。 追加情報については、「*クライアント認証* 『45 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。 追加情報については、「*安全なファイル転送* 『61 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。 追加情報については、「*ポート転送* 『71 ページ』」を参照してください。

第 3 章

構成ファイル

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。

```
$HOME/.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:`$HOME/.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* 『97 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/000095399/basedefs/xbd_chap09.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。

注記: 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、

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* 『23 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。

注記:

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 『85 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 『86 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

データ保護

暗号化

暗号化によって、転送中のデータの秘密性が保護されます。これは、送信前のデータを秘密鍵および Cipher を使って暗号化することで実施されます。受信されたデータは同じ鍵と Cipher を使用して解読される必要があります。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。

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-sha512

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	使用可能な値は、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-ripemd160」、「hmac-sha256」、および「hmac-sha512」です。これらの値すべてに対応するには「AnyMac」を使用します。「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、および「hmac-md5-96」に対応するには「AnyStdMac」を使用します。追加のオプションには「none」、「any」(「AnyMac」+「none」に相当)、および「AnyStd」(「AnyStdMac」+「none」に相当) があります。また、複数の MAC をカンマ区切りの一覧として指定することもできます。「none」が、合意された MAC である場合、メッセージ認証コードは使用されません。この場合はデータの整合性が保護されないため、「none」を含むオプションの使用はお勧めできません。

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

米国政府の連邦情報処理規格 (FIPS) 140-2 では、暗号化モジュールに対するセキュリティ要件が定められています。暗号化製品は、特定の要件に照らし合わせて検証され、米国政府の認定を受けた独立系の試験機関により、11 の項目についてテストされます。この検証内容は、米国標準技術局 (NIST) に提出されます。検証内容はここで確認され、証明書を発行されます。加えて、暗号化アルゴリズムも、その他の FIPS 仕様に基づいて検証され、認定を受けることもあります。認定を受けた製品のリストと、ベンダが定めたセキュリティポリシー (モジュールが、どのような動作について認定を受けたかを定義するもの) については、次のサイトを参照してください。
http://csrc.nist.gov/cryptval/vallists.htm

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。

注記: サーバ上の **FipsMode** 設定を変更した場合、この変更を完全に有効にするにはサーバを再起動する必要があります。SIGHUP 信号によって新規のセッションは FIPS モードになりますが、これは既存の接続に影響を与えません。

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。
- ユーザ鍵とホスト鍵の両方で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 (`/etc/ssh2/sshd2_config`) で PAM を無効にします (`UsePAM=no`)。

第　5　章

サーバ認証

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、ホスト鍵のフィンガープリントを提供します。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フィンガープリントを表示します。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 『174 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* 『17 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フィンガープリントが含まれています。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、正しいフィンガープリント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「man-in-the-middle (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フィンガープリントを表示します。

```
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フィンガープリントが、実際のホスト鍵のフィンガープリント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 `$HOME/.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 (`$HOME/.ssh2/ssh2_config`) に追加できます。

```
StrictHostKeyChecking=yes
```

ホスト公開鍵のフィンガープリントの表示

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フィンガープリント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フィンガープリント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フィンガープリントを表示するには

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フィンガープリントを表示します。

```
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。証明書を使うと、サーバ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。

Reflection PKI Services Manager は PKI 設定の集中管理に対応しています。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、対応しているすべての Attachmate 製品に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。

サーバ証明書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。これらの鍵と証明書は、サーバ認証の場合は、サーバ上にインストールおよび構成され、ユーザ認証の場合は、クライアント上にインストールおよび構成されている必要があります。

鍵とそれに関連付けられた証明書を取得するにはいくつかの方法があります。使用する方法は、既存の鍵用の証明書を取得するか、新規の鍵を生成してその鍵用の証明書を取得するか、秘密鍵と証明書の両方を CA から取得するかに応じて異なります。

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、要求ファイルが 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、秘密鍵と、そ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なります。

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

これにより、要求ファイルが 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

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。詳細については、「Reflection PKI Services Manager ユーザガイド (http://support.attachmate.com/manuals/)」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。

注記: 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 root として Reflection PKI Services Manager サーバにログインします。

2 *Reflection PKI Services Manager のインストール* 『15 ページ』.

3 トラストアンカとして指定する証明書のコピーをローカルストアに保存します。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 PKI Services Manager 構成ファイルをテキストエディタで開きます。既定の名前と場所は以下のとおりです。

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

5 **TrustAnchor** キーワードを使用して、トラストアンカを識別します。例えば、以下のようになります。

```
TrustAnchor = trustedca.crt
```

または、

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

注記: 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。

---

6 証明書破棄の確認を構成します。例えば、以下のようになります。

目的	構成のサンプル
LDAP サーバ上に格納された CRL を使用する	`RevocationCheckOrder = crlserver` `CRLServers=ldap://crlserver`
OCSP レスポンダを使用する	`RevocationCheckOrder = ocsp` `OCSPResponders = http://ocspresponder`

注記: 既定で、PKI Services Manager はローカルストア内の CRL を検索します。この構成を使用する場合は、CRL をローカルストアにコピーする必要があります。

7 証明書の信頼チェーンで中間証明書が必要とされる場合は、中間証明書へのアクセスを構成します。例えば、以下のようになります。

目的	構成のサンプル
ローカルストアに追加した中間証明書を使用する	`CertSearchOrder=local`
LDAP サーバ上に格納された証明書を使用する	`CertSearchOrder=certserver` `CertServers=ldap://ldapserver`

8 構成ファイルに変更内容を保存します。

9 *PKI Services Manager マップファイル*『159 ページ』をテキストエディタで開きます。既定の名前と場所は以下のとおりです。

```
/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。例えば、以下のようになります。

```
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*『165 ページ』」を参照してください。

注記: 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) 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 構成であるかどうかをテストします。

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

No errors. Configuration is valid:
```

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

```
/usr/local/sbin/pkid restart
```

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* 『37 ページ』をインストールします。例えば、以下のようになります。

```
/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。* 『17 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、以下のようになります。

```
/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、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 ホストおよびアドレスを指定します。例えば、以下のようになります。

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

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。

```
HostKeyAlgorithms=x509v3-sign-rsa,x509v3-sign-dss,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。

- 認証方式が `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。
- 認証方式が `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. 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
```

注記: **-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

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。

注記: 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`

3 (オプション) **AuthKbdInt.Retries** を使用して、ユーザが許可されるキーボード対話型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。

```
AuthKbdInt.Retries=5
```

4 (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)* 『55 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を構成する必要があります。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 (`$HOME/.ssh2/identification`) に、秘密鍵を識別する行を追加します。
3. 公開鍵をサーバ上のユーザのディレクトリ (`$HOME/.ssh2`) にコピーします。
4. サーバ上のユーザの「authorization」ファイル (`$HOME/.ssh2/authorization`) に、公開鍵を識別する行を追加します。

クライアント上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。

既定では公開鍵認証が許可されているため、この手順は、この既定の設定を変更する場合だけ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。例えば、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/秘密鍵のペアを生成します。

例えば、次のコマンドでは、現在の作業ディレクトリに既定の (2048-bit RSA) 鍵ペア (`mykey` と `mykey.pub`) が作成されます。この鍵作成処理中に、パスフレーズを入力するよう求めるメッセージが表示されます。パスフレーズを入力した場合、この鍵を使用して認証を行う時は必ずその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、パスフレーズ保護されていない鍵を作成します。このオプションは安全性は高くありませんが、スクリプトおよびバッチファイルで使用する場合は有効なことがあります。**-t** は鍵の種類 (この例では DSA) を指定します。鍵の名前が指定されていないため、鍵は、既定の名前と場所 (この例では `$HOME/.ssh2/id_dsa_1024_myhost_a`、ここで myhost は **hostname** コマンドによって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 クライアント識別ファイルを作成 (または編集) します。このファイルの既定の名前と場所は `$HOME/.ssh2/identification` になります。ユーザのみが書き込むことができるように (推奨値 600) このファイルを構成します。

4 識別ファイルに、作成した秘密鍵の行を追加します。鍵の項目の形式では `IdKey` の後に秘密鍵の名前が続きます。例えば、次のようになります。

```
IdKey /home/joe/mykey
IdKey id_dsa_1024_myhost_a
```

注記: パス情報が提供されない場合、クライアントは $HOME/.ssh2/ に示される鍵を検索します。

サーバ上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。

既定では公開鍵認証が許可されているため、この手順は、この既定の設定を変更する場合だけ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。例えば、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 $HOME/.ssh2 です。

3 このユーザ用の鍵の「authorization」ファイルをこのサーバ上に作成します。このファイルには、サーバのユーザ認証で受け入れられる鍵の一覧が含まれます。既定の名前と場所は $HOME/.ssh2/authorization です。ユーザのみが書き込むことができるように (推奨値 600) このファイルを構成します。

4 「authorization」ファイルに、コピーした公開鍵の行を追加します。鍵の項目の形式では Key の後に公開鍵の名前が続きます。例えば、次のようになります。

```
Key /pathto/mykey.pub
Key id_dsa_1024_myhost_a.pub
```

この一覧にある鍵はいずれも、サーバでユーザ認証に使用できます。絶対パスを指定しないかぎり、鍵はユーザ固有の構成ディレクトリ (既定では $HOME/.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。

公開鍵認証のためのファイルとディレクトリのパーミッション

公開鍵認証をよりセキュアに実装するために関連するファイルとディレクトリのパーミッションと所有者(owner) を正しく構成します。下表に示す基準に満たない場合は、その接続動作時に公開鍵認証が失敗します。

注記:

- **StrictModes** の設定は、サーバもクライアントも既定値は有効(=yes) で、更に厳密になります。
- 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。

ファイル/ディレクトリ	パーミッション推奨値	StrictModes = no の場合	StrictModes = yes の場合
ユーザディレクトリ (`$HOME/.ssh2/` 既定値) とその親ディレクトリ	ユーザディレクトリ = 700	条件なし	ユーザのみが「書き」と「実行」権限 (744)
秘密鍵ファイル	600	ユーザのみが「読み」と「書き」権限 (400 または 600)	ユーザのみが「読み」と「書き」権限 (400 または 600)
クライアント側 `identification` ファイル	600	条件なし	ユーザのみが「書き」権限 (600 または 644)
サーバ側 `authorization` ファイル	600	ユーザのみが「書き」権限 (600 または 644)	ユーザのみが「書き」権限 (600 または 644)

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。

注記: 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

- **ssh-add** を使用して、例えば、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。

注記:

- 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。

注記: Reflection PKI Services Manager は PKI 設定の集中管理に対応しています。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、対応しているすべての Attachmate 製品に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

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。詳細については、「Reflection PKI Services Manager ユーザガイド (http://support.attachmate.com/manuals/)」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。

注記: 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 root として Reflection PKI Services Manager サーバにログインします。

2 *Reflection PKI Services Manager のインストール* 『15 ページ』.

3 トラストアンカとして指定する証明書のコピーをローカルストアに保存します。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 PKI Services Manager 構成ファイルをテキストエディタで開きます。既定の名前と場所は以下のとおりです。

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

5 **TrustAnchor** キーワードを使用して、トラストアンカを識別します。例えば、以下のようになります。

```
TrustAnchor = trustedca.crt
```

または、

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

注記: 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。

6 証明書破棄の確認を構成します。例えば、以下のようになります。

目的	構成のサンプル
LDAP サーバ上に格納された CRL を使用する	`RevocationCheckOrder = crlserver` `CRLServers=ldap://crlserver`
OCSP レスポンダを使用する	`RevocationCheckOrder = ocsp` `OCSPResponders = http://ocspresponder`

注記: 既定で、PKI Services Manager はローカルストア内の CRL を検索します。この構成を使用する場合は、CRL をローカルストアにコピーする必要があります。

7 証明書の信頼チェーンで中間証明書が必要とされる場合は、中間証明書へのアクセスを構成します。例えば、以下のようになります。

目的	構成のサンプル
ローカルストアに追加した中間証明書を使用する	`CertSearchOrder=local`
LDAP サーバ上に格納された証明書を使用する	`CertSearchOrder=certserver` `CertServers=ldap://ldapserver`

8 構成ファイルに変更内容を保存します。

9 *PKI Services Manager マップファイル*『159 ページ』をテキストエディタで開きます。既定の名前と場所は以下のとおりです。

```
/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。例えば、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*『165 ページ』」を参照してください。

注記: 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) 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 構成であるかどうかをテストします。

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

No errors. Configuration is valid:
```

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

```
/usr/local/sbin/pkid restart
```

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。例えば、以下のようになります。

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

注記: 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。例えば、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 ホストおよびアドレスを指定します。例えば、以下のようになります。

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

注記: 既定で、PKI Services Manager はポート 18081 をリッスンします。

5 公開鍵認証を許可/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。* 『37 ページ』.

2 証明書および秘密鍵をインストールします。例えば、以下のようになります。

```
$HOME/.ssh2/userkey

$HOME/.ssh2/userkey.crt
```

注記: 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 ID ファイル (既定は $HOME/.ssh2/identification) を作成 (または編集) します。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 $HOME/.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)。例えば、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
```

プラグ可能な認証モジュール (PAM)

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`

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。例えば、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。

注記: 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
```

注記: 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。
	注記: 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

注記: 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
```

注記: 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*『17 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。

注記: 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
```

注記: 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

Reflection for Secure IT は、安全なファイル転送プログラムとして、**sftp** および **scp** コマンドを用意しています。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コピー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** の安全な代替方法を提供します。**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* 『97 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* 『135 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* 『137 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。

双方向の sftp セッションを開くには

1 リモートホストに接続します。例えば、次のようになります。

```
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。

正常に接続が確立されると、以下のプロンプトが表示されます。

```
sftp>
```

2 次のいずれかを実行します。

目的	使用するコマンド
指定できるコマンドの一覧を表示する	**help**; 例えば、次のようになります。 `sftp> help`
指定できるコマンドの詳細を表示する	**help** *コマンド*; 例えば、次のようになります。 `sftp> help put`
ファイルを転送および管理する	*指定できるコマンド* 『137 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、次のようになります。 `sftp> put demo`
セッションを終了する	**quit**; 例えば、次のようになります。 `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、ファイル管理を安全な方法で自動化できます。

sftp バッチファイルを作成および実行するには

1 クライアントとサーバを、GSSAPI などの非対話型クライアント認証方式またはパスフレーズ保護なしの公開鍵に対応するよう構成します。

注記: **sftp** バッチファイルオプション (**-B**) の使用時は、対話を必要とする認証方式は使用できません。

2 クライアントで、**sftp** バッチファイルを作成します。バッチファイルでは、任意の双方向コマンド対応の **sftp** 『137 ページ』 を使用できます。例えば、以下のコマンドを使って `demo` という名前のファイルを作成します。

```
get demo/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
```

クライアントでバッチファイル内のコマンドが実行され終了します。

注記:

- ログインに成功後、**sftp** によって、**bye**、**exit**、または **quit** コマンドが検出されるまでバッチファイル内の各コマンドが実行され、接続が終了します。
- バッチファイル内のコマンドが失敗すると、**sftp** では残りのコマンドが引き続き実行されてから、失敗した最初のコマンドのエラーコードが返されます。ただし、「-」(ダッシュ) が前に付いたコマンドは、実行に失敗した時も常に 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には、**scp** を使用します。

---

sftp 転送方式 (ASCII またはバイナリ) の構成

SFTP は、ASCII とバイナリの 2 つの転送方式に対応しています。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 Window コンピュータ間での ASCII 転送時は、改行文字が、各システムに応じた適切な形式で変換されます (必要な場合は、改行文字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。
`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

sftp コマンド	機能
	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。

転送元と転送先のファイル名には、ファイルのコピー元またはコピー先を示すためにホストとユーザの指定を含めることができます。2 つのリモートホスト間のコピーが許可されます。ワイルドカード文字を使用できます。再帰がオフになっている場合 (既定)、名前の置換はファイル名でのみ行われ、ディレクトリでは行われません。再帰が有効になっている場合 (**-r** を使用)、名前の置換はファイルとディレクトリで行われます。既定で、既存のファイルは上書きされます。この上書きの動作を制御するには、**--overwrite** を使用します(両ファイルが同一である場合は、この設定の値に関係なく転送は行われません)。

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* 『97 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

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。

中断されたファイル転送の自動再開

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。

注記: 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。

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。このキーワードは、**all**、**none**、**browse**、**download**、**upload**、**delete**、**rename** のうちの 1 つ以上のオプションからなるカンマ区切りの一覧に対応しています。**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。

注記: 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

注記: 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。最初のファイル (`file1`) については、サーバ上でユーザ、グループ、およびその他の読み取り/書き込みアクセス (666) が許可され、2 番目のファイル (`file2`) については、サーバ上でユーザの読み取り/書き込みアクセスと、グループおよびその他の読み取り専用アクセス (644) が許可されています。ダウンロード後、両方のファイルについて、クライアント上でユーザのみの読み取り/書き込みアクセス (600) が許可されます。

```
$ 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

sftp または **scp** を使用してサーバへファイルをアップロードした場合、そのアップロード済みファイルのファイル権限は、サーバ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 (`$HOME/.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。

注記: 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、サーバ上の環境ファイルでも、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	`$HOME/.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) について、クライアント (abchost) 上でユーザ、グループ、およびその他の読み取り/書き込みアクセス (644) が許可されています。アップロード後、そのファイルについて、サーバ (xyzhost) 上でユーザのみの読み取り/書き込みアクセス (600) が許可されます。

```
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

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

用語	定義
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以下のように機能します。

1. Secure Shell 接続が確立されると、Secure Shell クライアントは、指定された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 『173 ページ』を開きます。ほとんどの場合、このソケットは、Secure Shell クライアントホストで実行されているアプリケーションのみが使用できます。

   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、(直接、アプリケーションサーバホストおよびポートへではなく) 転送されたポートへ接続するように構成されます。当該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へ送信され、次に Secure Shell クライアントへ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て、Secure Shell チャネルを介して Secure Shell サーバへ安全に送信します。
4. Secure Shell サーバはデータを受信し、復号化して、そのデータをアプリケーションサーバによって使用される着信先ホストおよびポートへリダイレクトします。

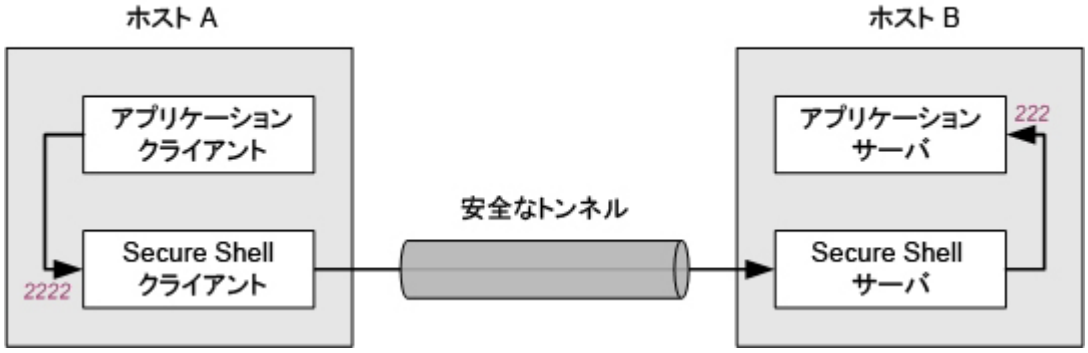
   注意: 最終宛先ホストおよびポートが Secure Shell サーバホスト上にない場合、Secure Shell ホストとアプリケーションサーバホスト間でデータは平文で送信されます。

5. アプリケーションサーバから返されたデータは Secure Shell サーバへ転送されます。Secure Shell サーバはそのデータを暗号化して、SSH トンネルを介して Secure Shell クライアントへ安全に送信します。Secure Shell クライアントはデータを復号化し、そのデータを元のアプリケーションクライアントへ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
```

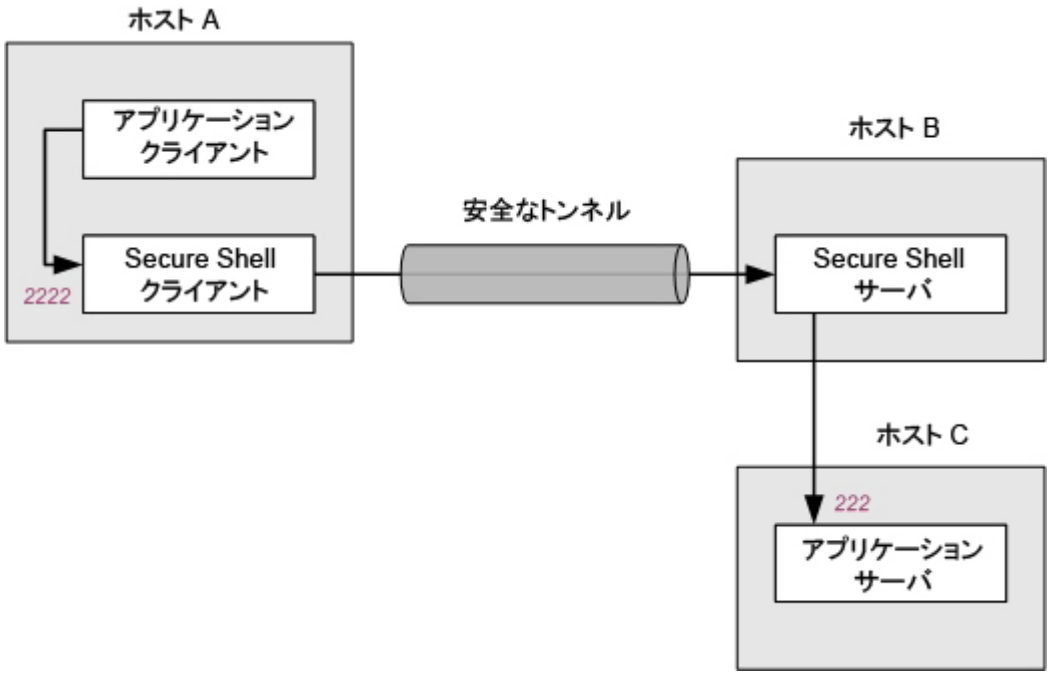
以下に示す図は、ローカルポート転送の 2 とおりの使用法を示しています。



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、以下のように行われます。

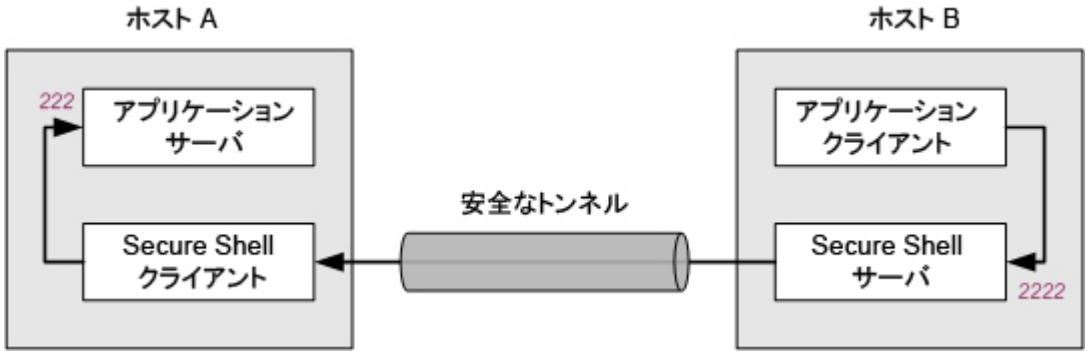
1. Secure Shell 接続が確立されると、Secure Shell サーバは、受信を待っている指定のポートを使用して Secure Shell サーバホスト上に受信待ちの *ソケット* 『173 ページ』を開きます。
2. Secure Shell サーバホスト上で実行されているクライアントアプリケーションは、(直接、アプリケーションサーバのホストおよびポートではなく) 受信を待っているポートに接続するよう構成されます。そのクライアントが接続を確立すると、すべてのデータが受信待ちのポートに送信され、次に、Secure Shell サーバへリダイレクトされます。
3. データは Secure Shell サーバで暗号化され、SSH トンネルを介して Secure Shell クライアントに安全に送信されます。
4. Secure Shell クライアントでは、データが受信され、解読されて、サーバアプリケーションで使用される宛先のホストおよびポート (Secure Shell クライアントホスト上) へリダイレクトされます。

5. サーバアプリケーションからの戻りデータが Secure Shell クライアントに送信されます。データはここで暗号化され、Secure Shell サーバに SSH トンネルを介して安全に送信されます。データは Secure Shell サーバで解読され、元のクライアントアプリケーションにリダイレクトされます。

リモート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 -R listening_port:app_host:hostport user@sshserver
```

以下に示す図は、考えられる 1 つのリモートポート転送構成を示しています。



アプリケーションサーバと Secure Shell クライアントの両方はホスト A で実行されます。Secure Shell サーバとアプリケーションクライアントの両方はホスト B で実行されます。ホスト B のポート 2222 へ送信されたすべてのデータは ホスト A 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、以下のコマンドによって構成します。

```
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、FTP クライアント (Secure Shell クライアントと同じコンピュータ上にある) から送信された FTP 通信は、myhost.com 上で実行されている FTP サーバに転送されます。この構成では、localhost:2121 に接続するよう FTP クライアントを構成します。

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

または、

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

注記: 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、FTP クライアント (Secure Shell サーバと同じコンピュータ上にある) から送信された FTP 通信は、FTP サーバ (Secure Shell クライアントと同じコンピュータ上にある) に転送されます。この構成では、ポート 3333 に接続するよう FTP クライアントを構成します。

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

または、

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

注記: 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 サーバ間で、セキュリティ保護された通信を実現できます。既定では X 転送は有効になっています。X 転送は以下のようにして行われます。

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. サーバが X 転送に対応している場合、サーバはサーバ自身をサーバホスト上のプロキシ X サーバとして設定し、クライアントシェル内の DISPLAY 環境変数を、そのプロキシの X ディスプレイをポイントするように設定します。
3. サーバホスト上で X クライアントプログラムを実行すると、プロキシのディスプレイに接続します。

4. Secure Shell クライアントがプロキシ X クライアントとして機能し、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 プロトコル情報は Secure Shell チャネルを介して転送されます。

X11 設定関連

クライアントの設定 **ForwardX11** は X11 転送を有効または無効にします(既定値は yes)。サーバの設定 **TrustX11Applications** 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されているとみなすかどうかを指定します(既定値は no)。

状況によっては、これらの設定から成る構成が X クライアントアプリケーションの起動速度に影響を及ぼすことがあります。これは、3 つ以上のシステムが関係している場合に発生します。例えば、次のようになります。

> システム 1 では X サーバと Secure Shell クライアントが実行されています。
> システム 2 では X クライアントアプリケーション、Secure Shell クライアント、および Secure Shell サーバが実行されています。
> システム 3 では X クライアントアプリケーションと Secure Shell サーバが実行されています。

ユーザが、X11Forwarding を yes (既定値) に、TrustX11Applications を no (既定値) に設定してシステム 1 から システム 2 への **ssh** 接続を確立する時は、X アプリケーションを起動するのに遅れは生じません。

しかし、ユーザが、X11Forwarding を yes (既定値) に、TrustX11Applications を no (既定値) に設定して新しいシェルからシステム 3 への後続の **ssh** 接続を確立した場合は、ユーザの認証後、システム 3 から起動された X アプリケーションが、システム 1 で実行されている X サーバ上に表示されるまでの間に長い遅延 (6 秒程度) が発生することになります。この遅延は、xauth アプリケーションが X サーバとの通信を試行し、新しいクッキーを登録する間に生じます。この遅延が発生しないようにして、システム 3 でアプリケーションを実行するには、2 番目の接続に対して TrustX11Applications を yes に設定します。

注記: 2 番目の接続に対して TrustX11Applications を yes に設定しても、システム 1 で実行されている X サーバでセキュリティ上のリスクが高まることはありません。この理由は、xauth アプリケーションが、システム 2 での最初の X11 転送 (システム 1 から開始) で作成された既存のクッキー (TrustX11Applications が no に設定されている) を登録するためです。

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*	Open the specified port on the Secure Shell server host (*listening_port*) and forward data to the destination *host* and *hostport*.
-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 `$HOME/.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 プログラムの完全パスを指定します。既定値は `/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* 『108 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

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の制御に使用できるサーバ設定の概要を示します。

既定では、サーバホスト上にアカウントを持つクライアントユーザはすべて、サーバに接続して、端末セッションを開き、自分のユーザアカウントに対して許可されたすべてのローカルファイルおよびディレクトリに任意のクライアントコンピュータからアクセスできます。サーバ構成ファイル (`/etc/ssh2/sshd2_config`) を編集することで、クライアントユーザ、クライアントグループ、およびクライアントコンピュータに関してアクセスをカスタマイズできます。

目的	使用するコマンド
接続の最大数を設定する	**MaxConnections**
指定されたセッションの種類のみにアクセス許可を制限する	**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 Wrapper を使用してアクセスを制御する	**LibWrap**
sftp および **scp** ユーザまたはグループを限ら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。

サーバでは以下のロジックを使って、接続を許可するかどうかを決定します。

1. 指定のアクセスの分類 (ホスト、ユーザ、グループ、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確認し、クライアントが、いずれかの拒否の式に一致した場合は、アクセスを拒否します。
2. 同じ分類に対して「Allow」キーワードが構成されているかどうかを確認します。
   - 「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。
   - 構成されている「Allow」キーワードが*ある*場合、サーバは、クライアントが、許可の式に一致する場合*のみ*アクセスを許可します。

例

以下では、アクセスの許可または拒否、またはその両方の組み合わせを使用する例を示します。

目的	例
「abc」で始まる名前のユーザのみへのアクセスを許可する	`AllowUsers= abc.*`
123.156.78 で始まる IP アドレスを持つクライアントホストへのアクセスを拒否し、ほかの任意のクライアント上のユーザへのアクセスを許可する	`DenyHosts=123\.156\.78\..*`
badpc を除き、acme.com ドメインに属するすべてのホストへのアクセスを許可し、その他のドメインのクライアントへのアクセスを拒否する	`AllowHosts=.*\.acme\.com` `DenyHosts=badpc\.acme\.com`

mypc を含めて、acme.com ドメインに属するすべてのホストへのアクセスを拒否し、その他のドメインのクライアントへのアクセスを許可する	`DenyHosts=.*\.acme\.com`
	`AllowHosts=mypc\.acme\.com`
	`AllowHosts=.*`
	注記: 最後の行がないと、クライアントはアクセスを許可されません。この理由は、クライアントが許可一覧に追加された後は、許可の式に一致した場合のみそれらのクライアントはアクセスを許可されるからです。

注記: *副構成ファイル* 『25 ページ』を使用してユーザ固有およびホスト固有の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。

第 10 章

デバッグのログ記録および監査

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

デバックは、ssh、sftp、および scp コマンドラインで構成できます。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`) で設定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。
		注記: **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 `$HOME/.ssh2/ssh2_config` です。)

キーワード	説明
LogLevel	デバッグレベルを設定します。メッセージは `syslog` に記録されます。
QuietMode	**-q** および **LogLevel** = quiet に相当します。
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。

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	使用する設定	出力場所	注記
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	デバッグファイルは、ファイル名形式 debugYYMMDD_HHMMSS を使用して作成されます。ここで、YY= 年、MM= 月、DD= 日、HH= 時、MM= 分、SS= 秒です。 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。

監査 (メッセージの記録)

Reflection for Secure IT サーバは、以下の監査サービスを提供します。これらのサービスは常に有効になっています。

- 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

出力場所はプラットフォームに依存します。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、11.23) PARISC	/var/adm/wtmp	/etc/utmp	/var/adm/btmp
HPUX (11.23、11.31) Itanium	/var/adm/wtmps	/etc/utmpx	/var/adm/btmps
AIX 5.2、5.3、6.1	/var/adm/wtmp /etc/security/lastlog	/etc/utmp	/etc/security/failedlogin /etc/security/lastlog
Solaris 8、9、10	/var/adm/wtmpx	/var/adm/utmpx	/var/adm/loginlog
RHEL 3、4、5	/var/log/lastlog /var/log/wtmp	/var/run/utmp	/var/log/btmp
SLES 9、10	/var/log/wtmp	/var/run/utmp	/var/log/btmp

監査の構成用のキーワード

目的	使用コマンド	出力場所	注記
サーバイベントメッセージを表示する	**LogLevel**	syslog	構成ファイルの読み込み後、メッセージは syslog に出力されます。
	SftpLogCategory	syslog (既定値)	
sftp-server イベントメッセージを表示する	**SftpSyslogfacility** **LogLevel**	--	sftp- server メッセージを (既定の機能ファイルではなく) 指定の代替ファイルへ送信するには **SftpSyslogFacility** を使用します。この操作は、既定のファイルにこれらのメッセージが読み込まれないようにするために行います。
既定のファシリティーを変更する	**SyslogFacility**	N/A	この設定は、サーバからのメッセージの記録に使用されるファシリティーを指定します。既定は「AUTH」です。この値は、syslogd がどのように構成されているかに対応している必要があります。

注記:

- 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。

付録

このセクション内

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、システム全体の構成ファイルと同じになります。推奨されるパーミッション = 644。

`/etc/ssh2/ssh2_config`

システム全体の構成ファイル。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、既定値がコメント行として表示されます。システム全体の設定を変更するにはこのファイルを編集します。キーワードおよび指定できる値の詳細については、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、ユーザが未知のホストに関する確認メッセージに対して「yes」と応答すると鍵がこの場所に自動的に追加されます(この動作は、構成ファイル内で **StrictHostKeyChecking** キーワードを使用して変更することができます)。バージョン 7.0 以降では、ホスト鍵のファイル名には以下の形式が使用されます。

key_*port*_*host*,*IP*.pub

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 IP アドレスです。

以前のバージョンでは、key_*port*_*host*.pub が使用されていました。この形式には現在も引き続き対応しています。

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

システム全体の既知のホスト。この一覧内の鍵を持つホストは、コンピュータの全ユーザで信頼されています。鍵はこの場所に自動的にインストールされません。システム全体の信頼されたホストを追加するには、このディレクトリを作成して、ホストの公開鍵のコピーをそのディレクトリ内に保存します。
`$HOME/.ssh2/hostkeys/key_*.pub` に対して前述のファイル名の形式を使用します。

```
$HOME/.ssh2/identification
```

ユーザ認証に公開鍵または証明書を使用する場合は、識別ファイルが必要です(これは、既定のファイル名と場所を示しています。識別ファイルの名前および場所は、**ssh** コマンドラインで **-i** を使用するか、構成ファイルで **IdentificationFile** キーワードを使用することで再定義できます)。識別ファイルには、クライアントユーザが保持する 1 つ以上の秘密鍵の一覧が含まれます。この一覧にある鍵はいずれも、クライアントでユーザ認証に使用できます。複数の鍵が一覧に含まれている場合、クライアントはまず一覧内の最初の鍵を試してから、以降、順番にほかの鍵を試していきます。パス情報が提供されない場合、クライアントは `$HOME/.ssh2/` に示される鍵を検索します。このファイルは、ユーザのみが書き込むことができるようにする必要があります (権限 = 600 または 644)。

標準の鍵の場合、以下の構文を使用して鍵を一覧に追加します。

```
IdKey <keyname>
```

例えば、次のようになります。

```
IdKey id_dsa_2048_a
```

n X.509 証明書に関連付けられた鍵の場合は、以下の構文を使用します。

```
CertKey <keyname>
```

関連付けられた証明書は指定の鍵と同じディレクトリ内にある必要があり、同じ名前をベースに `.crt` ファイル拡張子を付加する必要があります。

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 (in /etc/ssh2) が使用されます。ユーザ固有のディレクトリ (既定では $HOME/.ssh2) 内のファイルは、個々のクライアントユーザからの接続に適用されます。

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、グループまたはほかのユーザが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する設定については、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。このファイルの読み取りおよび書き込みは root だけが可能である必要があります。また、このファイルは、ユーザのみが読み出す/書き込むことができるよう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およびホスト固有の副構成ファイル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副構成ファイル。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は、このサーバへのすべての Secure Shell クライアント接続に使用する環境変数が設定されます(設定可能な環境変数は、キーワード **SettableEnvironmentVars** によって制御されます)。推奨されるパーミッション = 644。注記: このファイルで指定された環境変数設定は、/etc/default/login や /etc/environment などの、標準のシステムファイルで構成済みのすべての値よりも優先されます。ユーザ固有の環境ファイル ($HOME/.ssh2/environment) に構成された設定は、このグローバルファイルに構成された設定よりも優先されます。シャープ記号 (#) はコメント行を示しています。構文は次のようになります。

environment_variable=value

/etc/nologin

ログインを root に限定します。このファイルが存在する場合、root のみにログインが許可されます。ほかのユーザがログインしようとすると、テキスト nologin が表示されます。

`<piddir>/sshd2_22.pid`

外部からの接続を受け付ける処理の PID が含まれます。この PID ディレクトリは、使用しているオペレーティングシステムによって決まります。この名前の中で符号化されているポート番号 (既定で 22) は **Port** キーワードの値によって決まります。別の名前または場所を指定するには、**PidFile** キーワードを使用します。

`/etc/motd`

motd (message of the day、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。このメッセージ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 つまたは複数の RADIUS 認証サーバの一覧を示す、ユーザが作成したファイル。上記に示したファイル名は必要ありません。このファイルを作成したら、**RadiusFile** キーワードを使用してファイル名を指定します。RADIUS サーバごとに、名前、ポート、および共有する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。構文は次のようになります。

server1:*port1*:*shared_secret1*

server2:*port2*:*shared_secret2*

ユーザ固有のサーバファイル

`$HOME/.ssh2`

サーバ上にあるユーザ固有のファイルの既定のディレクトリ(別の場所を指定するには、**UserConfigDirectory** キーワードを使用します。)推奨されるパーミッション = 700。

`$HOME/.ssh2/authorization`

既定のクライアント「authorization」ファイル(別のファイルを指定するには、**AuthorizationFile** キーワードを使用します)。このファイルは、クライアントユーザの Secure Shell 公開鍵認証で必要とされます。各ユーザは、そのユーザのディレクトリに「authorization」ファイルが存在している必要があります。このファイルは、ユーザのみが書き込むことができるよう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で使用され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。 キーワードは大文字と小文字が区別されません。シャープ記号 (#) はコメント行を示しています。指定できるキーワードは以下のとおりです。

key

サーバがこのユーザについて受け入れる鍵を指定します。鍵の項目の形式では、「key」の後に、公開鍵を含むファイルの名前が続きます。絶対パスを指定しないかぎり、鍵はユーザ固有の構成ディレクトリ (既定では `$HOME/.ssh2`) 内にあるとみなされます。例えば、以下に示す行では、指定の鍵のうちのいずれかを使用したユーザの認証を承認します。

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

options

このオプションのキーワードは、前に指定した鍵に適用されるオプションを指定するために使用します。指定の鍵のすべてのオプションは 1 行内に構成する必要があります。空白を使用できます。また、オプションは、鍵を含む行のすぐ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 つの **Options** キーワード **command**、**allow-from**、**deny-from** を指定できます。

command *command*

指定されたコマンドがリモートホスト上で実行され、次に接続が閉じます。例えば、この構成では、mykey.pub が認証に使用される時は常に、スクリプト「myscript」が実行されます。

```
key mykey.pub
options command="sh myscript"
```

allow-from *IP-address*

この鍵は、指定の IP アドレスからの接続でのみ許可されます。

deny-from *IP-address*

この鍵は、指定の IP アドレスからの接続で許可されません。

`$HOME/.hushlogin`

このファイルが存在する場合は、ユーザの前回のログイン、その日のメッセージ、および電子メールの確認が表示されないようになります。

`$HOME/.ssh2/environment`

このファイルが存在する場合は、このユーザのログイン時に設定される環境変数が設定されます(設定可能な環境変数は、キーワード **SettableEnvironmentVars** によって制御されます)。推奨されるパーミッション = 644。注記: このファイルで指定された環境変数設定は、`/etc/default/login` や `/etc/environment` などの、標準のシステムファイルで構成済みのすべての値より優先されます。また、グローバルファイル (`/etc/ssh2/environment`) に構成された設定よりも優先されます。シャープ記号 (#) はコメント行を示しています。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 `$HOME/.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

クライアントが対応するアドレス形式を指定します。使用可能な値は、「any」(どのアドレス群を使用するかはシステムが決定できる)、「inet」(IPv4 のみを受け入れる)、および「inet6」(IPv6 が優先されるが、IPv4 も受け入れる) です。既定値は「inet」です。また、**-4** および **-6** コマンドラインオプションを使用して、アドレス群の優先順位を構成することもできます。

AllowedAuthentications

クライアントが試行する認証方式、およびそれらの試行順序を指定します。対応している方式は「gssapi-keyex」、「gssapi-with-mic」、「publickey」、「keyboard-interactive」、および「password」です。対応している方式を指定するにはカンマ区切りの一覧を使用します。クライアントは、一覧の先頭から最後まで順番に認証方式を試行します。接続に使用される認証技術は、クライアントの優先順位の最上位にある方式であり、サーバもまたこの最上位の方式を許可します。複数の方式を要求するようサーバが構成されている場合は、1 つの接続を確立するのに、複数の認証方式が必要になる場合があります。自動化されたスクリプトに対応するには、対話が最小限で済む方式を一覧の先頭に置く必要があります。既定値は、「gssapi-with-mic, publickey, keyboard-interactive, password」です。

AuthenticationSuccessMsg

認証が正常に完了した時に「Authentication successful (認証が完了しました)」というメッセージを表示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。

BatchMode

パスワードやパスフレーズの入力画面を含め、ユーザ入力に関するすべての問い合わせを無効にするかどうかを指定します。このキーワードは、スクリプトおよびバッチジョブの場合に役立ちます。**¥fBStrictHostKeyChecking** が「ask」に、**BatchMode** が「yes」に設定されている場合、クライアントでは未知のホスト鍵に関するすべての問い合わせに対する応答を「no」とみなします。許可される値は「yes」および「no」です。既定は「no」です。

CheckHostIP

公開鍵のファイル名でコード化されているホスト名と 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、その鍵が、`¥fIkey_port_host,IP.pub` という形式を使用して未知のホストの格納場所に追加されます。**CheckHostIP** が有効な場合は、指定されたホストの実際の IP が、そのホストのコード化された 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 偽装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。

注記: v. 7.0 以前のバージョンを使用してホスト鍵の格納場所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。前の形式の鍵を使用する場合は、**CheckHostIP** を無効にします。

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。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。 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、**ssh** コマンドラインで **-c** オプションを使用して暗号化アルゴリズムを指定することもできます。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。注記: **scp** と **sftp** 利用の場合は、この設定の値に関係なく、転送されたすべてのポートが自動的にクリアされます。

Compat.RSA.HashScheme

公開鍵または X.509 証明書認証で使用される RSA 鍵の電子署名を確認するために MD5 ハッシュアルゴリズムに対応するかどうかを指定します。使用可能な値は「yes」と「no」です。このキーワードを「no」に設定すると (既定)、SHA-1 ハッシュを含む署名のみが受け入れられます。「yes」に設定すると、SHA-1 ハッシュまたは MD5 ハッシュのいずれかを含む署名が受け入れられます。

Compression

圧縮を有効にするかどうかを指定します。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。 許可される値は「yes」および「no」です。既定は「no」です。 圧縮は、**ssh** コマンドラインで **-C** オプションを使用しても無効にできますが、構成ファイルでのみ有効にすることができます。

ConnectionReuse

新規の **ssh**、**scp**、および **sftp** セッションで、確立済みの接続を再使用できるかどうかを指定します。この機能を使用すると、再認証を必要とせずに新規の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、対象のホスト、ポート、およびユーザがすべて、確立済みの接続で使用されているものと同一である場合に新規のセッションで既存のトンネルが再利用されます。「no」に設定すると、クライアントではセッションごとに新規の接続が確立されます。つまり、新規の接続ごとに認証処理が繰り返され、変更された接続固有の設定 (転送や Cipher など) がある場合は、それらの設定が適用されます。

注記: サーバ管理者が、**ChrootSftpGroups** または **ChrootSftpUsers** を使用して制限付きのディレクトリアクセスを構成している場合は、接続の再使用が失敗することがあります。

ConnectionTimeout

サーバへの接続試行時にクライアントが待機する最大時間 (秒) を指定します。既定値は 0 に設定されます。これは、クライアントによって制限が設定されず、実際の制限はオペレーティングシステムによって決まることを意味しています。

DefaultDomain

既定のドメイン名を指定します。コマンドラインには短いホスト名を入力する一方で、完全修飾ドメイン名を送信して接続を確立できるようにする場合は、この設定を構成ファイルに追加できます。**DefaultDomain** の値を構成済みで、「.」(ドット) 文字を含まないホスト名を入力する場合は、**DefaultDomain** の値は、「.」文字を使用するホスト名と結合されます(注記: **DefaultDomain** 文字列の先頭にオプションのドットを含めることができます。この場合、文字列の先頭の「.」は無視されます)。英数字は値として受け入れられます。例えば、**DefaultDomain** が「`acme.com`」または「`.acme.com`」のいずれかに設定されている場合、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、**ssh** コマンドラインで **-n** オプションを使用してこの設定を構成することもできます。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、**ssh** の Man ページの「ESCAPE SEQUENCES」を参照してください。また、**ssh** コマンドラインで **-e** オプションを使用してエスケープ文字を設定することもできます。

ExitOnForwardFailure

要求されたすべての動的な、ローカルおよびリモートのポート転送を構成できない場合に **ssh** によって接続が終了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「no」です。

FileCopyAsciiExtensions

自動モード転送が有効になっている場合に **sftp** セッションの間 ASCII 転送を使用するファイルの種類を指定します。ほかのすべてのファイルではバイナリ転送を使用します。カンマまたは空白区切りの一覧を指定します。ワイルドカード (zsh-glob) 文字を使用できます。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、拡張子を引用符で囲むか、またはバックスラッシュをエスケープ文字として使用します。既定値は「txt, htm*, pl, php*」です(**sftp** セッションの間に **setext** を使用して、そのセッション用に別のファイル一覧を指定できます。現在の一覧を表示するには **getext** を使用します)。

注記: この設定は、自動転送が有効になっている場合のみ適用されます。既定で転送方式はバイナリに設定されます。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

FIPS 140-2 標準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がすべての通信で使用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「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、**ssh** コマンドラインで **-t** オプションを使用してこの設定を構成することもできます。

ForwardAgent

認証エージェントへの接続 (確立されている場合) がリモートマシンへ転送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。

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、**ssh** コマンドラインで **-X** オプションを使用してこの設定を構成することもできます。

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、**ssh** コマンドラインで **-g** オプションを使用してこの設定を構成することもできます。

GoBackground

ポート転送が構成済みであり、Secure Shell セッションをバックグラウンドで実行する場合はこのキーワードを使用します。使用可能な値は「yes」、「no」、および「oneshot」です。既定値は「no」です。1 つ以上のポート転送ルールが構成されている場合、「yes」と「oneshot」のいずれの値でも、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」を指定すると、Secure Shell セッションはバックグラウンドで維持され、手動で処理が強制終了されるまで転送要求が無期限に認可され続けます (これは、**ssh** コマンドラインで **-f** を使用した場合と同じ動作になります)。
「oneshot」を指定すると、バックグラウンドセッションは、転送された 1 つの接続が開始されるのを待機し、その接続が閉じられると直ちに終了します(これは、**ssh** コマンドラインで **-fo** を使用した場合と同じ動作になります)。

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。既定値は空文字列です。このキーワードをホストスタンザの式と組み合わせて、ホストへの接続用の別の名前を作成することができます。このキーワードがスタンザ外に表示されている場合は、接続の既定のホストを指定するために使用できます。

HostCA

このキーワードの使用は現在推奨されません。これは、**TrustAnchor** と同義です。

HostCANoCRLs

このキーワードの使用は現在推奨されません。これは、**TrustAnchor** と同義です。注記: 証明書の取り消し確認は、Reflection for Secure IT 構成ファイルでは設定できません。取り消し確認を構成するには Reflection PKI Services Manager を使用します。

HostCertNameCheck

証明書を使っ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。
HostCertNameCheck を「yes」に設定すると、接続に指定されたホスト名または IP アドレスが、証明書に設定された許可の識別情報に含まれている場合のみ認証が成功します(許可の識別情報を構成するには PKI Services Manager マップファイルを使用します)。**HostCertNameCheck** を「no」に設定すると、クライアントでは、設定された許可の識別情報が無視され、有効な証明書がすべて受け入れられます。
HostCertNameCheck を「ask」(既定値) に設定すると、ユーザは、サーバ名が、許可の識別情報に該当しない場合に、操作を続行するかどうかを確認するメッセージを受信します。

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。この設定は、サーバが証明書認証と標準のホスト鍵認証の両方を行うように設定されている場合に有効です。Secure Shell プロトコルでは、ホスト認証は 1 回だけ試行できます。ホストで証明書が提示され、クライアントが証明書を使用したホスト認証を行うように設定されていない場合は、接続に失敗します(複数回の認証試行に対応しているユーザ認証とは異なります)。既定値は「x509v3-sign-rsa,x509v3-sign-dss,ssh-rsa,ssh-dss」です。

HostKeyAlias

ホスト鍵が、クライアントの既知のホスト鍵のディレクトリに保存される場合に実際のホスト名の代わりに使用する別名を指定します。ホスト鍵は `key_port_host,IP.pub` 形式の名前で保存されます。指定する値によって、保存されたホスト鍵名の `host` 部分が上書きされます。このオプションは、Secure Shell 接続をトンネリングしたり、単一ホストで複数のサーバを実行する場合に役立ちます。

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 `$HOME/.ssh2/identification` です。詳細については、以下の「FILES」セクションを参照してください。また、**¥fBssh** コマンドラインで **-i** オプションを使用してこの設定を構成することもできます。

IdentityFile

このキーワードの使用は現在推奨されません。これは、**IdentificationFile** と同じ動作になります。

KeepAlive

クライアントがサーバへ 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。このキーワードの使用は現在推奨されません。代わりに **ServerAliveInterval** を使用してください。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する鍵交換アルゴリズムを指定します。指定できる値は「diffie-hellman-group1-sha1」および「diffie-hellman-group14-sha1」です。複数のアルゴリズムをカンマ区切りの一覧として指定できます。既定値は「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ために使用します。この設定を構成する構文は以下のとおりです。

[*protocol*/][*listening_host*:]*listening_port*:*host*:*hostport*

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 (*listening_port*) を使用して Secure Shell クライアントホスト上にソケットを開きます(複数のインタフェースが存在するクライアントホスト上では、インタフェースを指定するために *listening_host* を使用します)。(接続先のホストとポートに直接送信するのではなく) 転送されたソケットにデータを送信するようにアプリケーションクライアント (データの転送元のクライアント) を構成します。そのクライアントによって接続が確立されると、転送されたポートに送信されたすべてのデータが、安全なトンネルを介して Secure Shell サーバへリダイレクトされます。Secure Shell サーバは、それらのデータを解読して、接続先のソケット (host,hostport) へ送信します。ゲートウェイポートオプションが有効になっている場合、転送されたローカルポートは、Secure Shell クライアントと同じコンピュータ上で実行されているクライアントに対してのみ使用できます。オプションの *protocol* は **tcp** または **ftp** のいずれかになります。

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。

以下の例では、ローカル転送を使用して、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メールクライアントと、Secure Shell サーバと同じコンピュータ上で実行されているメールサーバとの間の安全な電子メール通信を実現します。ローカルのメールクライアントは、ローカルポート 14300 と通信するよう構成されています。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、サーバでポート 143 にリダイレクトされます。

```
LocalForward=14300:localhost:143
```

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

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

また、**ssh** コマンドラインで **-L** オプションを使用してローカル転送を構成することもできます。

LogLevel

syslog に記録される **ssh** メッセージに使用される冗長レベルを指定します。使用可能な値は「fatal」、「error」、「quiet」、「info」、「verbose」、「debug1」(「debug」と 1 に相当)、「debug2」(2 に相当)、「debug3」(3 に相当)、および「trace」(「debug99」と 99 に相当) です。これらの値に関連付けられた syslog レベルは、fatal には CRIT、error と quiet には ERROR、info と verbose には INFO、debug1/debug2/debug3 と trace には DEBUG が対応します。 既定値は「info」です。

MACs

クライアントが対応する MAC (メッセージ認証アルゴリズム) を指定します。 使用可能な値は、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-ripemd160」、「hmac-sha256」、および「hmac-sha512」です。これらの値すべてに対応するには「AnyMac」を使用します。「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、および「hmac-md5-96」に対応するには「AnyStdMac」を使用します。追加のオプションには「none」、「any」(「AnyMac」+「none」に相当)、および「AnyStd」(「AnyStdMac」+「none」に相当) があります。また、複数の MAC をカンマ区切りの一覧として指定することもできます。「none」が、合意された MAC である場合、メッセージ認証コードは使用されません。この場合はデータの整合性が保護されないため、「none」を含むオプションの使用はお勧めできません。

また、**ssh** コマンドラインで **-m** オプションを使用して MAC を構成することもできます。既定値は「AnyStdMac」です。

NoHostAuthenticationForLocalHost

このオプションは、クライアントが localhost に接続している時にホスト認証を無効にします。このオプションは、ホームディレクトリがコンピュータ間で共有されている場合に役立ちます。このような状況では、localhost がコンピュータごとに異なるホストを示し、クライアントユーザは、ホスト鍵が変更されていることについて多くの警告を受け取ります。このオプションを「yes」に設定すると、localhost の認証が無効になるため、ユーザにこれらの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

応答の際にパスワード入力を求める回数を指定します。注記: サーバでは、許容されるパスワード入力の最大試行回数も指定できます。**¥fBNumberOfPasswordPrompts** をサーバで構成済みの値より大きな値に設定した場合は、サーバ側の上限に達した時に接続が失敗します。既定値は 3 です。

PasswordPrompt

パスワード認証で表示されるメッセージを指定します。2 つの変数オプションを指定できます。`%r` はユーザ名に置き換えられ、`%h` はホスト名に置き換えられます。既定値は「`%r@%h's password:`」です(この設定はキーボード対話型認証には適用されません)。

PkidAddress

PKI Services Manager への接続に使用されるポートを指定します。既定値は localhost:18081 です。

PkidPublicKey

Reflection PKI Services Manager の識別情報の確認に使用される公開鍵の名前と場所を指定します。既定値は `/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。既定は 22 で、これは Secure Shell 接続の標準ポートです。 また、**ssh** コマンドラインで **-p** オプションを使用してこの設定を構成することもできます。

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、**ssh** コマンドラインで **-q** オプションを使用してこの設定を構成することもできます。

RekeyIntervalSeconds

新規のセッション鍵についてネゴシエーションを開始するまでにクライアントが待機する秒数を指定します。値には、整数を指定する必要があります。既定値は 3600 です。この鍵を **RekeyLimit** と組み合わせて使用できます。この場合、クライアントは、最初の上限に達した時に新しい鍵交換を開始します。

RekeyLimit

クライアントが新規のセッション鍵についてネゴシエーションを開始するまでに送信可能なデータの最大量を指定します。引数は、バイト数とオプションの接尾語「K」、「M」、または「G」(それぞれ、キロバイト、メガバイト、ギガバイトを示す) になります。既定値を使用する場合はこの値を 0 に設定します。既定値は Cipher に応じて「1G」から「4G」の間の数値になります。この鍵を **RekeyIntervalSeconds** と組み合わせて使用できます。この場合、クライアントは、最初の上限に達した時に新しい鍵交換を開始します。

RelaySignals

クライアントがサーバに中継する信号を指定します。**RelaySignals** では、信号 ABRT、ALRM、FPE、HUP、ILL、INT、PIPE、QUIT、SEGV、TERM、USR1、USR2 のうちのいずれかから成るコンマ区切りの一覧を使用できます。信号 KILL と STOP は捕捉/ブロック/無視できないため、使用できません。既定では信号は中継されません。

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ために使用します。この設定を構成する構文は以下のとおりです。

`[`*`protocol`*`/][`*`listening_host`*`:]`*`listening_port`*`:`*`host`*`:`*`hostport`*

Secure Shell 接続が確立されると、Secure Shell サーバは、指定のリモートポート (*`listening_port`*) を使用してホスト (Secure Shell サーバを実行しているコンピュータ) 上にソケットを開きます(複数のインタフェースが存在するサーバホスト上では、インタフェースを指定するために *`listening_host`* を使用します)。(接続先のホストとポートに直接送信するのではなく) 転送されたソケットにデータを送信するようにクライアントアプリケーション (データの転送元のアプリケーション) を構成します。クライアントによって接続が確立されると、転送されたポートに送信されたすべてのデータが、安全なトンネルを介して Secure Shell クライアントへリダイレクトされます。Secure Shell クライアントは、それらのデータを解読して、転送先のソケット (`host,hostport`) へ送信します。オプションの *`protocol`* は **tcp** または **ftp** のいずれかになります。

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

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

また、**ssh** コマンドラインで **-R** オプションを使用してリモートポート転送を構成することもできます。

SendNOOPPackets

クライアントで Secure Shell チャネルを介してサーバへ NOOP メッセージを送信するかどうかを指定します。このオプションを「yes」に設定した場合は、**ServerAliveCountMax** を 3、**ServerAliveInterval** を 600 に設定した場合の動作と同じになります。許可される値は「yes」および「no」です。既定は「no」です。

ServerAliveCountMax

この設定は、応答しなくなったサーバへのセッションを終了するために使用します。**ServerAliveInterval** が非ゼロ値に設定された場合のみ適用されます。**¥fBServerAliveCountMax** では、クライアントが、サーバからの返信メッセージを受信し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。既定値は 3 です。例えば、**ServerAliveInterval** が 600 に、**ServerAliveCountMax** が 3 に設定されている場合、クライアントは、サーバが無応答の状態が続く時に 3 つのメッセージを送信するまで 10 分おきにサーバへメッセージを送信します。つまり、クライアントは、約 30 分後に無応答状態の接続を閉じます。

ServerAliveInterval

Secure Shell チャネルを介してサーバへ NOOP メッセージを送信する時間間隔を秒単位で設定します。指定された時間間隔内にサーバからデータが受信されない場合は、クライアントによりサーバへメッセージが送信されます。この値を非ゼロ値に設定することで、クライアントが引き続き動作していることを Secure Shell サーバと TCP スタックに通知し、Secure Shell 接続が引き続き稼働していることをすべてのネットワーキングハードウェア (ルータや NAT など) に通知することができるようになり、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可能になります。この設定を **ServerAliveCountMax** と組み合わせると、応答しなくなったサーバへの接続を終了することができます。既定値は 0 です。つまり、クライアントは、メッセージを送信しないよう構成されます。

SetRemoteEnv

シェルまたはコマンドの実行前にサーバ上に設定する環境変数を指定します。値の形式は `VAR=val` にする必要があります。ここで、`val` は空白のままにできます。引数では大文字と小文字が区別されます。

注記: このキーワードで設定される値は累積されます。つまり、このキーワードを 1 つまたは複数の構成ファイルに複数回構成することで複数の変数を設定することができます。

StrictHostKeyChecking

このキーワードは、ホストが認証で未知の鍵を提示した場合のクライアントの動作を決定します。可能な値は以下のとおりです。

「yes」 - ホスト鍵が手動でユーザのホスト鍵ディレクトリ (`$HOME/.ssh2/hostkeys`)、またはシステム全体のホスト鍵ディレクトリ (`/etc/ssh2/hostkeys`) にコピーされている場合のみ接続は成功します。クライアントによって、ホスト鍵はユーザのコンピュータに追加されません。これはもっとも安全なオプションです。

「ask」 - これが既定値です。クライアントによって、ユーザが未知のホストからの鍵を受け入れるかどうかを確認するメッセージが表示されます。この確認メッセージには、ホストの識別情報の確認に使用できるホスト鍵の指紋が表示されます。ユーザが「yes」と応答すると、クライアントによってそのホスト鍵が、ユーザのディレクトリ (`$HOME/.ssh2/hostkeys`) の既知のホスト鍵に追加され、後続の接続におけるホスト識別情報の確認でこの鍵が使用されます。

「no」 - 未知のホスト鍵が自動的にユーザのホスト鍵ディレクトリ (`$HOME/.ssh2/hostkeys`) に追加され、後続の接続におけるホスト識別情報の確認で使用されます。ユーザは、いつ未知のホスト鍵が提示されたかを認識しません。

StrictModes

公開鍵認証で必要とされるファイルおよびディレクトリ権限を指定します。許可される値は「yes」および「no」です。既定は「yes」です。 「yes」に設定されている場合、ユーザディレクトリ (`$HOME/.ssh2/`) とすべての親ディレクトリは、ユーザのみが書き込み可能および実行可能である必要があります (モード 744 が認可される)。推奨されるユーザディレクトリの権限は 700 です。この場合、ユーザ識別ファイル (既定で `$HOME/.ssh2/identification`) は、ユーザのみが読み出す/書き込むことができるように構成される必要があります (600 が推奨されるが、644 も認可される)。

注記: すべての秘密鍵に対して追加のファイル権限による制限が実施されます。これらの鍵は、現在の **StrictModes** の設定に関係なく、ユーザのみが読み出すことができるように構成される必要があります。秘密鍵へのアクセス制限が十分でない場合、公開鍵認証は常に失敗します。推奨される秘密鍵の権限は 600 です。

TrustAnchor

このキーワードはオプションであり、サーバ認証に証明書を使用する場合のみ適用されます。既定では、Reflection PKI Services Manager は、認証で提示される証明書を、構成済みのすべての信頼アンカーを使用して検証します。このキーワードを使用して、どの Reflection PKI Services Manager 信頼アンカーを証明書の検証に使用できるかを制限できます。PKI Services Manager 格納場所で提供される証明書の件名の DN (識別名) を指定するか、証明書のファイル名を使用できます。ただし、指定した信頼アンカーは、Reflection PKI Services Manager に対しても構成する必要があります (PKI Services Manager **TrustAnchor** キーワードを使用)。

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ているとみなす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。 「yes」に設定すると、リモートの X11 クライアントに X11 ディスプレイへの完全なアクセス権が与えられます。「no」に設定すると、X11 アプリケーションは信頼されていないとみなされます。この設定では、信用できないホストへの接続で、そのホスト上のアプリケーションが、転送された X11 接続を使用して入力操作を「探り出す」リスクを回避できます。

User

リモートサーバ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に定義することで、異なるホストに対して異なるユーザ名を構成することができます。既定値は、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、**ssh** コマンドラインで **-v** オプションを使用してこの設定を構成することもできます。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。既定値はシステムによって異なります (`/usr/X11R6/bin/xauth` など)。

付 録　D

サーバ構成のキーワード

サーバ構成ファイルで以下の設定を構成できます。既定のファイルは `/etc/ssh2/sshd2_config` です。

AccountManagement

ユーザアカウントの検証に **sshd** で使用されるアカウント管理システムを構成します。アカウント管理サービスにより、アカウントがアクティブかどうか、およびパスワードがまだ有効かどうかが決定されます。使用可能な値は「password」、「pam」、および「none」です。既定値は「pam」です。

pam - アカウント管理に PAM を使用します。使用される認証方式には関係なく、PAM アカウント管理が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。

none - アカウント検証を使用しません。問題の対策時のみアカウントを検証します。

AddressFamily

この設定は、サーバで、受信を待っているか、セッション用か、転送時の TCP ソケットを作成する時に使用されます。使用可能な値は、「any」(どのアドレス群を使用するかはシステムが決定できる)、「inet」(IPv4 のみを受け入れる)、および「inet6」(IPv6 のみを受け入れる) です。既定値は「inet」です。注記: **ListenAddress** の現在の値もまた、サーバが IPv4 または IPv6 アドレス使用の接続を受け入れるかどうかに影響する場合があります。

AllowAgentForwarding

エージェント転送が許可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。

AllowedAuthentications

サーバが対応する認証方式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。 (1 つまたは複数の認証方式を要求するには **RequiredAuthentications** を使用します **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

このキーワードには現在対応していません。前のバージョンでこのキーワードを使用している場合は、手動で設定を移行する必要があります。キーワード **AllowedAuthentications**、**RequiredAuthentications**、および **AuthKbdInt.Required** を参照してください。

AllowGroups

このキーワードは、指定されたグループのメンバであるユーザのみにログインを許可するために使用します。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* 『86 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は、すべてのグループがログインを許可されます。

AllowHosts

このキーワードは、指定のクライアントホストのみにログインを許可するために使用します。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* 『86 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は、すべてのクライアントホストが許可されます。

注記:

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。

AllowSftpCommands

ユーザが **sftp** および **scp** を使用して実行可能な操作の種類を制御します。このキーワードでは、「all」、「none」、「browse」、「download」、「upload」、「delete」、「rename」のうちの 1 つ以上の値から成るカンマ区切りの一覧を使用できます。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 オプションを使用すると、ユーザはファイルの内容の読み出しが可能になります。既定値は「all」です。

AllowTCPForwarding

このキーワードは、すべてのクライアントユーザへのポート転送を許可/拒否するために使用します。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への転送) とリモート (サーバからクライアントへの転送) の両方を制御できます。より詳細なレベルの制御を行うには **ForwardAcl** を使用します。

AllowTCPForwardingForGroups

このキーワードは、指定されたグループのメンバであるユーザのみにポート転送を許可するために使用します。正規表現に対応しています。

AllowTCPForwardingForUsers

このキーワードは、指定のユーザのみにポート転送を許可するために使用します。正規表現に対応しています。

AllowUsers

このキーワードは、指定のユーザのみにログインを許可するために使用します。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* 『85 ページ』」を参照してください。

AllowX11Forwarding

X11 転送が許可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。

AuthFailureErrorMessages

「no」に設定された場合、認証失敗に関する情報はクライアントへ送信されません。この設定は SSH の規則に適合します。「yes」に設定された場合、クライアントは失敗の理由に関する情報を受信します。警告:この設定では、情報を攻撃者に提供することになり、セキュリティ上のリスクが高まります。許可される値は「yes」および「no」です。既定は「no」です。

AuthImmediateDisconnect

「no」に設定すると、サーバは、失敗したすべての認証試行に対して同じ応答を返します。この設定は SSH の規則に適合します。「yes」に設定すると、ブロックされたアカウントが直ちに切断されます。警告:この設定では、有効なアカウント名に関する情報をクライアントに提供することになり、セキュリティ上のリスクが高まります。許可される値は「yes」および「no」です。既定は「no」です。

AuthKbdInt.Required

キーボード対話型認証に使用する認証方式を指定します。ユーザの認証が正常に完了するには、この指定された認証方式が成功する必要があります。使用可能な値は「pam」、「password」、および「radius」です。既定値は「password」で、ユーザの応答を標準のログインパスワードとして処理します。「pam」が指定されると、PAM モジュールが、認証およびパスワード管理に使用されます。「radius」が指定されると、1 台以上の RADIUS 認証サーバが認証に使用されます。

AuthKbdInt.Retries

キーボード対話型認証で許容される最大試行回数を設定します。既定値は 3 です。

AuthKbdInt.Verbose

サーバで、冗長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

公開鍵認証用にユーザ鍵を構成するために使用されるファイルの名前を指定します。ファイルは、全体パスが指定されないかぎり、`$HOME/.ssh2` に相対している (または **UserConfigDirectory** として設定されている場所) とみなされます。公開鍵認証が成功するには、クライアントユーザによって認証用に提示された鍵がこのファイル内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、「FILES」セクションを参照してください。

既定のファイルは `$HOME/.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 であり、この値より大きい値は使用できません。受け入れ可能な値の範囲は 512 ～ 32769 です。0 の指定は、既定値を使用した場合と同じになります。

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 であり、この値より小さい値は使用できません。0 の指定は、既定値を使用した場合と同じになります。

AuthPublicKey.Retries

公開鍵認証用にサーバで許容される最大試行回数を指定します。この数に達したら、これ以降、公開鍵を使用した認証試行は拒否されます。ただし、接続は切断されません。クライアントは、次の許可される方式を使って認証を試行できます。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。既定では `/etc/ssh2/ssh_banner_message` になります。

ChrootSftpGroups

属するユーザが、sftp プロトコル接続のホームディレクトリに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。 パターンは、GID ではなく、グループ名と照合されます。

注意: この設定は、Reflection for Secure IT クライアントからの **sftp** 接続と **scp** 接続の両方に影響します。端末セッションおよび OpenSSH **scp** 接続は、指定された式に一致するクライアントユーザの場合機能しません (Reflection for Secure IT **scp** 接続は、sftp サブシステムを使用します。OpenSSH **scp** 接続は、このサブシステムを使用しません)。

ChrootSftpUsers

sftp プロトコル接続のホームディレクトリに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。 パターンは、UID ではなく、ユーザ名と照合されます。

注意: この設定は、Reflection for Secure IT クライアントからの **sftp** 接続と **scp** 接続の両方に影響します。端末セッションおよび OpenSSH **scp** 接続は、指定された式に一致するクライアントユーザの場合機能しません (Reflection for Secure IT **scp** 接続は、sftp サブシステムを使用します。OpenSSH **scp** 接続は、このサブシステムを使用しません)。

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。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。 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機構により、サーバは、クライアントがいつ非アクティブな状態になったかを決定できるようになります。**ClientAliveCountMax** では、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状態でこの数に達すると、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。メッセージの送信間隔を指定するには、**ClientAliveInterval** を使用します。既定値は 3 です。

注記: これらの設定は SSH 接続に適用され、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、クライアントが無応答である場合、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チャネルを介してメッセージを送信します。クライアントが無応答である時にセッションを終了しクライアントを切断するまでにサーバが送信するメッセージの最大数を指定するには **ClientAliveCountMax** を使用します。既定値は 0 (無効) です。

Compat.RSA.HashScheme

公開鍵または X.509 証明書認証で使用される RSA 鍵の電子署名を確認するために MD5 ハッシュアルゴリズムに対応するかどうかを指定します。使用可能な値は「yes」と「no」です。このキーワードを「no」に設定すると (既定)、SHA-1 ハッシュを含む署名のみが受け入れられます。「yes」に設定すると、SHA-1 ハッシュまたは MD5 ハッシュのいずれかを含む署名が受け入れられます。

Compression

圧縮を有効にするかどうかを指定します。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。 許可される値は「yes」および「no」です。既定は「yes」です。

DenyGroups

このキーワードは、指定のユーザグループのログインを拒否するために使用します。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* 『86 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は、すべてのグループがログインを許可されます。

DenyHosts

このキーワードは、指定のクライアントホストのログインを拒否するために使用します。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* 『86 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は、すべてのクライアントホストが許可されます。

注記:

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。 さらに、ドメイン名を解決できないホストからアクセスできないようにするには **RequireReverseMapping** を「yes」に設定します。

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。

DenyTCPForwardingForGroups

このキーワードは、指定のユーザグループのポート転送を拒否するために使用します。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* 『86 ページ』」を参照してください。

DenyTCPForwardingForUsers

このキーワードは、指定のユーザのポート転送を拒否するために使用します。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* 『85 ページ』」を参照してください。

DenyUsers

このキーワードは、指定のユーザのログインを拒否するために使用します。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* 『85 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は、すべてのユーザがログインを許可されます。

FipsMode

FIPS 140-2 標準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がすべての通信で使用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「no」です。

ForwardACL

このキーワードは、ポート転送へのクライアントアクセスを詳細に制御するために使用します。正規表現に対応しています。構文は次のようになります。

```
ForwardACL allow|deny local|remote user_ex forward_ex [origin_ex]
```

user_ex は、ポート転送へのアクセスを許可/拒否されるユーザ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* 『85 ページ』」を参照してください。"

forward_ex は、*host%port* 形式の正規表現です。このオプションの意味は、構成されている制限がローカル転送に関するものか、またはリモート転送に関するものかによって異なります。ローカル転送制御を構成している場合、このオプションは対象のホストとポートを示します。リモート転送制御を構成している場合、ホストはサーバコンピュータ、ポートは、サーバがクライアントへ転送しているポートになります。

origin_ex は、IP アドレスを識別する正規表現です。このオプションの意味は、構成されている制限がローカル転送に関するものか、またはリモート転送に関するものかによって異なります。ローカル転送制御を構成している場合、このオプションは転送要求を行うクライアントマシンを示します。リモート転送制御を構成している場合、このオプションは、サーバ上の転送されたポートに接続しているコンピュータを示します。

GatewayPorts

クライアント用に転送されたポートへの接続をリモートホストに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「no」です。

HostCertificateFile

サーバ認証に使用される X.509 証明書を指定します。関連付けられた秘密鍵を指定するには **HostKeyFile** を使用します。

HostKeyFile

Specifies the filename and location of the private key used to authenticate the server. 既定では `/etc/ssh2/hostkey` になります。

HostSpecificConfig

ホスト固有の副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、サーバでは指定された副構成ファイルが使用されます。

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。

IdleTimeout

サーバが接続を終了するまでに接続が非アクティブ状態であり続ける時間を指定します。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、何も指定しません。時間は分単位 (m)、時間単位 (h)、日単位 (d)、または週単位 (w) 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 システムのみに適用されます。/etc/security/user 内の「rlogin」属性が無視されるか、適用されるかを指定します。使用可能な値は「yes」と「no」です。既定値は「no」です。つまり、サーバは現在の「rlogin」値を受け付けます。注記: /etc/security/user 内の「login」属性は、Secure Shell クライアントを使用して行われたリモートログインには適用されません。この動作は、**IgnoreRlogin** の値に関係ありません。

KeepAlive

システムで TCP キープアライブメッセージを他方の側に送信する必要があるかどうかを指定します。サーバでは、メッセージの送信頻度に関してシステム全体の値が使用されます。許可される値は「yes」および「no」です。既定は「yes」です。注記: **ClientAliveCountMax** および **ClientAliveInterval** は SSH 接続に適用され、メッセージは SSH トンネルを介して送信されます。**KeepAlive** 設定は TCP 接続に適用されます。この場合、TCP メッセージは安全なトンネルを介して送信されないので、偽装に対してより脆弱な状態になります。

KEXs

サーバが対応する鍵交換アルゴリズムを指定します。指定できる値は「diffie-hellman-group1-sha1」および「diffie-hellman-group14-sha1」です。複数のアルゴリズムをカンマ区切りの一覧として指定できます。既定値は「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

この設定は、GSSAPI (Kerberos 5) 認証を使用する場合に使用します。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。

注記: サーバは、libkrb5.so (または .sl on HP-UX PARISC) という名前のライブラリを要求します。この名前のライブラリが存在しない場合は、実際のライブラリをポイントする 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 Wrapper に動的に対応します。TCP Wrapper への対応を有効にするには、libwrap 共有ライブラリ (`LibWrap=/usr/lib/libwrap.so` など) への完全修飾パスを指定します。libwrap ファイルは、静的ライブラリではなく、共有ライブラリである必要があります。既定で、このキーワードは空白のままであり、TCP Wrapper 機能は無効になります。

注記: このキーワードを使用する前に、指定され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。これは、許可されたユーザのみが接続可能であるようにするために重要です。指定され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、ファイルが存在する場合も、**sshd** は起動しますが、そのファイルが有効なライブラリであることを確認しません。接続ごとに、**sshd** 処理では、指定されたファイルのロードが試行されます。ファイルが有効なライブラリではない場合、サーバではエラーメッセージがログに記録され、ユーザは接続を許可されます。

ListenAddress

sshd サーバソケットのバインド先のインタフェースのアドレスを指定します。IPv4 または IPv6 形式を使用する 1 つ以上のカンマ区切りの値を指定するか、「any」(既定) を使用できます。値「any」を使用した場合、サーバは、使用可能な任意の IPv4 または IPv6 アドレス (「`[::],0.0.0.0`」と等価) の受信を待つように構成されます。IPv4 アドレスのみを指定した場合、クライアントは接続に IPv4 アドレスを使用する必要があります。IPv6 形式のみを指定した場合、大多数のオペレーティングシステムでは引き続き IPv4 クライアントによる接続が可能です。この動作は、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ます。オプションで、コロンまたは空白と、その後にポート番号を付加することでアドレス内にポートを指定できます。このポートの値は **Port** キーワードの設定よりも優先されます。IPv6 アドレスを指定している場合は、アドレスを角カッコで囲む必要があります。例えば、次のようになります。

IPv4 の構文:`ListenAddress=209.85.171.99:6666`

IPv6 の構文:`ListenAddress=[::D155:AB63]:6666`

ListenAddress と **AddressFamily** 設定は相互に関連しています。**AddressFamily** が inet の場合、**ListenAddress** の値「any」は「`0.0.0.0`」に等しくなります。**AddressFamily** が inet6 の場合、ListenAddress の値「any」は「`[::]`」に等しくなります。**AddressFamily** が「inet」または「inet6」のいずれかに設定され、**ListenAddress** に別のアドレス群が指定されている場合、**sshd** は、構成ファイルエラーのため起動に失敗します。**ListenAddress** に IP アドレスではなくホスト名を指定した場合、**AddressFamily** の制限によって、そのホスト名が、該当するアドレス群に関連付けられていることと、サーバがそのアドレスにバインドされていることが要求されます。

LogCertificateSubject

認証に使用される証明書の [シリアル番号] と [件名] をシステムログに記録するかどうかを指定します。メッセージは、認証試行に成功した場合と失敗した場合の両方で記録されます。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」と 1 に相当)、「debug2」(2 に相当)、「debug3」(3 に相当)、および「trace」(「debug99」と 99 に相当) です。これらの値に関連付けられた syslog レベルは、fatal には CRIT、error と quiet には ERROR、info と verbose には INFO、debug1/debug2/debug3 と trace には DEBUG が対応します。 既定値は「error」です。

LogPublicKeyFingerPrint

認証に使用される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。メッセージは、認証試行に成功した場合と失敗した場合の両方で記録されます。許可される値は「yes」および「no」です。既定は「yes」です。

MACs

サーバでデータの整合性の確認に使用可能な MAC (ハッシュメッセージ認証コード) を指定します。使用可能な値は、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-ripemd160」、「hmac-sha256」、および「hmac-sha512」です。これらの値すべてに対応するには「AnyMac」を使用します。「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、および「hmac-md5-96」に対応するには「AnyStdMac」を使用します。追加のオプションには「none」、「any」(「AnyMac」+「none」に相当)、および「AnyStd」(「AnyStdMac」+「none」に相当) があります。また、複数の MAC をカンマ区切りの一覧として指定することもできます。「none」が、合意された MAC である場合、メッセージ認証コードは使用されません。この場合はデータの整合性が保護されないため、「none」を含むオプションの使用はお勧めできません。 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。

MaxStartups

認証されていない同時接続試行の最大許容回数を指定します。この制限に達すると、認証が成功するか、接続試行の LoginGraceTime 制限に達するまで追加の接続が切断されます。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。

PasswordGuesses

ユーザがパスワード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。

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

ルート権限を持つクライアントユーザがログイン可能かどうか指定します。使用可能な値は「yes」、「no」、および「without-password」です。「without-password」を指定すると、ユーザは、ユーザの認証に「public key」または「GSSAPI」認証方式が使用されている場合にかぎり、ルート権限なしでログインできます。既定値は「yes」です。つまり、すべての認証方式でルートのログインを許可します。

PidFile

sshd デーモンの処理 ID を含むファイルを指定します。完全修飾パスを使用します。ファイル名に文字列 `%s` が含まれている場合、こ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 への接続に使用されるポートを指定します。既定値は localhost:18081 です。

PkidPublicKey

Reflection PKI Services Manager の識別情報の確認に使用される公開鍵の名前と場所を指定します。既定値は `/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。

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` からの motd (message of the day、その日のメッセージ) テキストが出力されるかどうかを指定します(この設定は `/etc/issue` の表示を無効にしません)。許可される値は「yes」および「no」です。既定は「yes」です。

ProtocolVersionString

初期接続のプロトコル内でサーバが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(この文字列の最初の部分は常に「SSH-2.0-」になります。これは、サーバが対応する SSH バージョンを示しています。この情報はプロトコルの RFC で必要とされ、編集することはできません)。文字列に空白が含まれている場合は二重引用符を使用します。**ProtocolVersionString** が空の文字列である場合 (既定)、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含まれます。この番号は、サーバソフトウェアのアップグレード時に自動的に更新されます。

注記: クライアントの多くが、サーバの種類を識別し、互換機能を有効にするために、このプロトコル文字列を使用します。そのため、既定値を変更すると、結果として公開鍵認証が失敗する場合があり、また、サーバ間で異なるその他の機能が影響を受けることがあります。

QuietMode

このキーワードの使用は現在推奨されません。**LogLevel** を使用してください。

RadiusFile

RADIUS 認証の構成に使用されるファイルの名前を指定します。ファイルは、全体パスが指定されないかぎり、`/etc/ssh2` に相対しているとみなされます。ファイルの構文については、「FILES」セクションの `/etc/ssh2/radius_config` を参照してください。このキーワードに既定値はなく、値を指定しないこともできます。

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

このキーワードは、1 つ以上のクライアント認証方式を要求するために使用します。ユーザが認証されたとみなされるには、指定されたすべての認証方式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。

注記: **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の確認時に DNS ルックアップに成功する必要があるかどうかを指定します。この機能を有効にするには、**ResolveClientHostName** を「yes」に設定する必要もあります。許可される値は「yes」および「no」です。既定は「no」です。

ResolveClientHostname

ドメイン名へのクライアント IP アドレスの解決をサーバが試行するかどうかを指定します。「yes」に設定すると、接続時間が長くなることがありますが、任意のキーワードを、IP アドレスではなくドメイン名に基づいてホスト名に一致するよう構成している場合、この設定は必須になります(**AllowHosts**、**DenyHosts**、**UserSpecificConfig**、および **HostSpecificConfig** を参照してください)。また、このキーワードを「yes」に設定した場合は、IP アドレスではなく DNS 名がログに表示されるようになります。許可される値は「yes」および「no」です。既定は「yes」です。

注記: **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

サーバで許可されるセッションの種類を指定します。使用可能な値は「shell」(端末のシェルセッションを許可)、「exec」(クライアントによるサーバ上のコマンドの実行を許可)、および「subsystem」(**sftp** および **scp** 転送に対応する場合に必要) です。既定値は「shell, exec, subsystem」です。

SettableEnvironmentVars

クライアントセッションに対して構成可能な環境変数を指定します。この値により、クライアント上のクライアントキーワード **SetRemoteEnv**、ユーザ固有の環境ファイル (`$HOME/.ssh2/environment`)、およびサーバ上のグローバル環境ファイル (`/etc/ssh2/environment`) の適用範囲が制限されます。このキーワードは、既定の構成ファイル内で有効になっており、「LANG,LC_ALL,LC_COLLATE,LC_CTYPE,LC_MONETARY,LC_NUMERIC,LC_TIME, PATH,TERM,TZ,UMASK」に設定されています。

SftpLogCategory

SftpSysLogFacility で指定された機能に送信される sftp サーバメッセージの分類を決定します。カンマ区切りの一覧を使用します。既定値は「loginlogout,directorylistings,downloads,modifications,uploads」であり、すべての分類に対するログ記録が構成されます。これらのオプションのほかに「all」または「none」を指定できます。

SftpSysLogFacility

sftp-server サブシステムからのメッセージのログ記録に使用される機能コードを指定します。この値は既定で空白のままで、ログ記録は行われません。有効な値はプラットフォームによって異なります。syslog(3) を参照してください。「auth」に設定すると、**sshd** の既定値と同じ機能でログメッセージが格納されます。

StrictModes

公開鍵認証で必要とされるディレクトリ権限を指定します。許可される値は「yes」および「no」です。既定は「yes」です。 「yes」に設定されている場合、ユーザディレクトリ (`$HOME/.ssh2`) とすべての親ディレクトリは、ユーザのみが書き込み可能および実行可能である必要があります (モード 744 が認可される)。推奨されるユーザディレクトリの権限は 700 です。これらの条件が満たされないと、公開鍵認証は失敗します。

注記: 現在の **StrictModes** の設定に関係なく、各ユーザの「authorization」ファイル (`$HOME/.ssh2/authorization`) に対して、追加のファイル権限による要件が実施されます。このファイルは、グループや公開の書き込みアクセスができないように構成する必要があります (600 が推奨されるが、644 も認可される)。「authorization」ファイルのアクセス制限が十分でない場合、公開鍵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。引数には、クライアントがサブシステムを要求する時に実行されるコマンドを指定します。このキーワードの後の区切り文字は、ダッシュ、等号、または空白のいずれかになります。

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。次に示す既定の構成では、子処理内部で sftp サービスが実行されます。

```
Subsystem-sftp internal://sftp-server
```

SyslogFacility

サーバからのメッセージのログ記録に使用される機能コードを指定します。既定値は「AUTH」です。有効な値はプラットフォームによって異なります。syslog(3) を参照してください。

TrustAnchor

このキーワードはオプションであり、ユーザ認証に証明書を使用する場合のみ適用されます。既定では、Reflection PKI Services Manager は、認証で提示される証明書を、構成済みのすべての信頼アンカーを使用して検証します。このキーワードを使用して、どの Reflection PKI Services Manager 信頼アンカーを証明書の検証に使用できるかを制限できます。PKI Services Manager 格納場所で提供される証明書の件名の DN (識別名) を指定するか、証明書のファイル名を使用できます。ただし、指定した信頼アンカーは、Reflection PKI Services Manager に対しても構成する必要があります (PKI Services Manager **TrustAnchor** キーワードを使用)。

UseLogin

双方向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。

注記:

login(1) は、リモートのコマンド実行に使用されることはありません。

この設定を有効にすると **X11Forwarding** が無効になります。この理由は、login(1) では xauth(1) クッキーの処理方法が認識されないからです。

login(1) を使用することで特権の分離を無効にします。既定で、**sshd** では、認証の成功後に認証されたユーザの特権を持つ新しい処理が作成されます。これは、破損を特権のない処理内に制限することによって権限のエスカレーションを防ぐためです。**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、PAM を使用するようサーバを構成する方法 の 1 つです。使用可能な値は「yes」と「no」です。**UsePam** が構成されていない場合、サーバでは、**AuthKbdInt.Required**、**AccountManagement**、および **UsePamSessions** の現在の値が使用されます。このキーワードを「yes」に設定した場合は、`AuthKbdInt.Required=pam`、`AccountManagement=pam`、および `UsePamSessions=yes` の場合と同じ動作になります。このキーワードを「no」に設定した場合は、`AuthKbdInt.Required=password`、`AccountManagement=password`、および `UsePamSessions=no` の場合と同じ動作になります。

UsePAMAcctMgmt

このキーワードの使用は現在推奨されません。このキーワードを「yes」に設定した場合は、`AccountManagement=pam` の設定と同じ動作になります。

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「no」です。

UserConfigDirectory

ユーザ固有の情報用のディレクトリを指定します。このディレクトリには、認証ファイル (鍵認証に必要) および「ファイル」セクションに一覧表示されているその他のユーザ固有のファイルが含まれます。マクロ %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 UID)、%IG (= ユーザの GID) が認識されます。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、接続を試行するユーザに一致する場合、サーバでは指定された副構成ファイルが使用されます。

注記: この式のホスト部分を、IP アドレスではなくホストのドメイン名に基づいて一致するよう構成している場合は、さらに **ResolveClientHostName** を「yes」に設定する必要もあります。

VerboseMode

このキーワードの使用は現在推奨されません。Use **LogLevel**.

X11DisplayOffset

サーバで X11 転送に使用可能な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

サーバで、X11 転送がループバックアドレスにバインドされるか、ワイルドカードアドレスにバインドされるかを指定します。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。既定値はシステムによって異なります (`/usr/X11R6/bin/xauth` など)。

付 録　E

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**] [**-x**] [**-X**] [**-Y**]
[[*username*@]*host*[#*port*]] [*remote_command* [*arguments*] ...]

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-4

接続で IPv4 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成することもできます。

-6

接続で IPv6 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成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。 既定で、認証エージェント転送は、**ForwardAgent** キーワードが「yes」に設定され、有効になっています。**-a** を使用して構成ファイルの設定を無効にすることができます。

-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。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。 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 **Ciphers** キーワード (既定値「AnyStdCipher」) を使用して暗号化アルゴリズムを構成することもできます。

-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 **Compression** キーワードを使用して圧縮を有効にすることができます。**-C** を使用すると、構成ファイルの設定が上書きされます。

-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)。

-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「ESCAPE SEQUENCES」を参照してください。また、**EscapeChar** キーワードを使用して、構成ファイルにエスケープ文字を設定することもできます。

-f

ポート転送が構成済みであり、Secure Shell セッションをバックグラウンドで実行する場合はこのオプションを使用します。1 つ以上のポート転送ルールが構成されている場合、このオプションを指定すると、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。セッションはバックグラウンドで維持され、手動で処理が強制終了されるまで転送要求が無期限に認可され続けます(これは、構成ファイル内の設定 `GoBackground=yes` と同じ動作になります)。

-fo

これらのオプションは **-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 つの接続が開始されるのを待機し、その接続が閉じられると直ちに終了します(これは、構成ファイル内の設定 `GoBackground=oneshot` と同じ動作になります)。

-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 (`$HOME/.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。リモートホストは、ローカル転送ポートへの接続を許可されます。 また、**GatewayPorts** キーワードを使用して、構成ファイルでこの設定を構成することもできます。

-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 `$HOME/.ssh2/identification` です。 **IdentificationFile** キーワードを使用して、構成ファイルに識別情報ファイルを指定することもできます。

-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。 **Username** キーワードを使用して構成ファイルにユーザ名を指定することもできます(注記: ホストの指定の一部にオプションの [user@] を含めた場合、指定したユーザ名は **-l** によって上書きされます)。

-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 接続が確立されると、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上にソケットを開きます(複数のインタフェースが存在するクライアントホスト上では、インタフェースを指定するために *listening_host* を使用します)。(接続先のホストとポートに直接送信するのではなく) 転送されたソケットにデータを送信するようにアプリケーションクライアント (データの転送元のクライアント) を構成します。そのクライアントによって接続が確立されると、転送されたポートに送信されたすべてのデータが、安全なトンネルを介して Secure Shell サーバへリダイレクトされます。Secure Shell サーバは、それらのデータを解読して、接続先のソケット (host,hostport) へ送信します。ゲートウェイポートオプションが有効になっている場合、転送されたローカルポートは、Secure Shell クライアントと同じコンピュータ上で実行されているクライアントに対してのみ使用できます。オプションの *protocol* は **tcp** または **ftp** のいずれかになります。複数のクライアントアプリケーションが、転送されたポートを使用できますが、転送がアクティブな状態にあるには **ssh** が実行されている間だけです。

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、 **LocalForward** キーワードを使用して、構成ファイルにローカル転送を設定することもできます。

-m *mac_algorithm*

クライアントが対応する MAC (メッセージ認証アルゴリズム) を指定します。 使用可能な値は、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-ripemd160」、「hmac-sha256」、および「hmac-sha512」です。これらの値すべてに対応するには「AnyMac」を使用します。「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、および「hmac-md5-96」に対応するには「AnyStdMac」を使用します。追加のオプションには「none」、「any」(「AnyMac」+「none」に相当)、および「AnyStd」(「AnyStdMac」+「none」に相当) があります。また、複数の MAC をカンマ区切りの一覧として指定することもできます。「none」が、合意された MAC である場合、メッセージ認証コードは使用されません。この場合はデータの整合性が保護されないため、「none」を含むオプションの使用はお勧めできません。

また、**MACs** キーワード (既定値は「AnyStdMac」) を使用して構成ファイルに MAC を設定することもできます。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、**DontReadStdin** キーワードを使用して構成ファイルにこの設定を構成することもできます。

-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
```

-p *port*

サーバ上の接続先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。 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、**QuietMode** キーワードを使用して、構成ファイルでこの設定を構成することもできます。

-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 接続が確立されると、Secure Shell サーバは、指定のリモートポート (*listening_port*) を使用してホスト (Secure Shell サーバを実行しているコンピュータ) 上にソケットを開きます(複数のインタフェースが存在するサーバホスト上では、インタフェースを指定するために *listening_host* を使用します)。(接続先のホストとポートに直接送信するのではなく) 転送されたソケットにデータを送信するようにクライアントアプリケーション (データの転送元のアプリケーション) を構成します。クライアントによって接続が確立されると、転送されたポートに送信されたすべてのデータが、安全なトンネルを介して Secure Shell クライアントへリダイレクトされます。Secure Shell クライアントは、それらのデータを解読して、転送先のソケット (host,hostport) へ送信します。オプションの *protocol* は **tcp** または **ftp** のいずれかになります。

また、 **RemoteForward** キーワードを使用して、構成ファイルにリモート転送を構成することもできます。

-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、**ForcePTTYAllocation** キーワードを使用して、構成ファイルでこの設定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、「-d 2」を使用した場合と同じ動作になります。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。

-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、**ForwardX11** キーワードを使用して、構成ファイルでこの設定を構成することもできます。

-x

X11 接続の転送を無効にします。 また、**ForwardX11** キーワードを使用して、構成ファイルでこの設定を構成することもできます。

-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。

付 録　F

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。注記: 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*	ローカル転送を要求します。
-R[*bind_address*:]*port*:*host*:*hostport*	リモート転送を要求します。
-KL[*bind_address*:]*port*	ローカル転送をキャンセルします。
-KR[*bind_address*:]*port*	リモート転送をキャンセルします。

~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　G

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　H

ssh-keygen コマンドラインオプション

ssh-keygen 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** *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**`$HOME/.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 `¥fI/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。

-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。注記: 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 **-p**および **-N** と組み合わせて使用します。NULL パスフレーズに変更するには、このオプションを **-P** と組み合わせて使用します。

-F *key*

指定された鍵のフィンガープリントを 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 **-e** と組み合わせて使用します。

-o *key_name*

生成された秘密鍵のファイル名を指定します(公開鍵もまた作成されます。公開鍵の名前は、常に、秘密鍵と同じ名前に .pub ファイル拡張子を付けたものになります)。注記: **ssh-keygen** コマンドの末尾に 1 つ以上の鍵ファイル名を指定しても、鍵ファイルに名前を付けることができます。

-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。

-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お勧めします。

-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。既定は「rsa」です。

-V

ssh-keygen のバージョン情報を表示します。

-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 I

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**] [**-b** *buffer_size*] [**-c** *cipher*] [**-d**] [**-D** *debug_level*] [**-F** *file*] [**-h**] [**-i** *file*] [**-N** *max_requests*] [**-o** *option*] [**--overwrite**] [**-p**]

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-4

接続で IPv4 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成することもできます。

-6

接続で IPv6 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成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」のいずれかを指定できます。既定では、*newline_type* に指定した値によって、転送先の改行変換が設定されますが、引数の前に「src:」または「dest:」を付けることで、転送元または転送先のいずれかの変換を指定することができます。例えば、次のようになります。

scp -a src:unix –a dest:dos unixhost:src_file winhost:dest_file

既定値は「dest:unix」または「src:unix」になります。転送先と転送元の種類が同じである場合、変換は行われません。それ以外の場合は、「src」と「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。

転送先または転送元の変換を指定しないで **-a** を使用した場合、クライアントは、接続先のサーバから転送元または転送先の行末変換の取得を試みます。サーバがこの機能に対応していない場合は、DOS 行末変換が採用されます。

-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。最大値は、使用しているコンピュータの物理メモリ量によって制限されます。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。これにより、ユーザ入力に関するすべての問い合わせが無効になります。このオプションは、スクリプトおよびバッチジョブの場合に役立ちます。このオプションの使用時は、ユーザとの対話が必要とされる認証方式に対応できません。バッチモードでは、**--overwrite** が「no」に設定されないかぎり、**scp** によって既存の転送先ファイルが常に上書きされます。

-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。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。 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)。

Cipher が指定されていない場合は、Secure Shell 構成ファイル ssh2_config(5) 内の **Ciphers** キーワードによって Cipher が決定されます。既定値は「AnyStdCipher」です。

-d

転送先が、すでに存在するディレクトリになるようにします。例えば、以下に示すコマンドは、 `destination` という名前のディレクトリが存在する場合、`source_file` をそのディレクトリにコピーします。このディレクトリが存在しない場合、`¥fIsource_file` は `demo` ディレクトリにコピーされ、ファイル名として `destination` が付けられます。

```
scp source_file joe@myhost:~/demo/destination
```

-d フラグが追加された以下のコマンドは、`source_file` を `destination` ディレクトリにコピーしますが、このディレクトリが存在しない場合は失敗します。

```
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 (`$HOME/.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 `$HOME/.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 16 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。注記: 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

オリジナルのファイルの変更回数およびファイル属性を保存します。

-P *port*

サーバ上の接続先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。 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

静的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

進行状況インジケータの表示を無効にします。

-r

すべてのサブディレクトリを含めて、再帰コピーします。

-u

転送先へのコピーが完了したら転送元の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。

付 録　J

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**] [[*user*@]*host*[#*port*]]

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-4

接続で IPv4 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成することもできます。

-6

接続で IPv6 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成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。最大値は、使用しているコンピュータの物理メモリ量によって制限されます。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*

バッチ処理を行う **sftp** コマンドに使用するファイルを指定します。ログインに成功後、**sftp** によって、**bye**、**exit**、または **quit** コマンドが検出されるまで指定のファイル内の各コマンドが実行されてから、接続が終了します。このモードは、ユーザとの対話が必要とされる認証方式に対応できません。バッチファイルでは、以下に説明する任意の双方向コマンドを使用できます。バッチファイル内のコマンドが失敗すると、**sftp** では残りのコマンドが引き続き実行されてから、失敗した最初のコマンドのエラーコードが返されます。ただし、「-」(ダッシュ) が前に付いたコマンドは、実行に失敗した時も常に 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。 指定したセッションに使用される Cipher は、クライアントの優先順位の最上位にある Cipher であり、サーバもまたこの最上位の Cipher に対応します。 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)。Cipher が指定されていない場合は、Secure Shell 構成ファイル ssh2_config(5) 内の **Ciphers** キーワードによって Cipher 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*

この接続で使用できる MAC (メッセージ認証コード) を指定します。使用可能な値は、「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、「hmac-md5-96」、「hmac-ripemd160」、「hmac-sha256」、および「hmac-sha512」です。これらの値すべてに対応するには「AnyMac」を使用します。「hmac-sha1」、「hmac-sha1-96」、「hmac-md5」、および「hmac-md5-96」に対応するには「AnyStdMac」を使用します。追加のオプションには「none」、「any」(「AnyMac」+「none」に相当)、および「AnyStd」(「AnyStdMac」+「none」に相当) があります。また、複数の MAC をカンマ区切りの一覧として指定することもできます。「none」が、合意された MAC である場合、メッセージ認証コードは使用されません。この場合はデータの整合性が保護されないため、「none」を含むオプションの使用はお勧めできません。

-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 16 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。注記: 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。既定は 22 で、これは 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。

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場合に有効です。改行の既定の処理を無効にする必要がある場合は、*remote_newline* と *local_newline* を使用します。 *remote_newline* に指定できる値は、「DOS」(¥r¥n) と「Unix」(¥n) です。リモートの行末変換に対して明示的な値が指定されていない場合は、リモートホストに対して、変換を実施するよう問い合わせが行われます。リモートホストがこの機能に対応していない場合は、DOS 行末変換が採用されます。*local_newline* に指定できる値は「Unix」(¥n) だけです。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「auto」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。指定のファイル拡張子を含むファイルには ASCII 転送が使用され、その他すべてのファイルにはバイナリ転送が使用されます。既定の ASCII ファイルの種類の一覧は「`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の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します。このモードでは、ファイルは一切変更されずに転送されます。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で ASCII モードをオフにする場合に役立ちます。

bye

これは、**quit** と同義です。

cd *directory*

リモートディレクトリを *directory* に変更します。

chgrp *group file*

file で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権を *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 で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。このモードは数値形式 (664 など) で指定する必要があります。ファイルおよび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 で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 を終了せずにリモートサーバへの接続を閉じます。.

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

これは、**ls** と同義です。

exit

これは、**quit** と同義です。

get [**--preserve**] | [**-p**] *remote-file* [*remote-file ...*]

指定されたファイルを現在のローカル作業ディレクトリへコピーします(別の場所へコピーするには、**lcd** を使用してローカル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している場合は、その既存のファイルが上書きされます。ワイルドカード文字を使用できますが、名前の置換はファイル名でのみ行われ、ディレクトリでは行われません。ファイルの属性およびタイムスタンプを保持するには **--preserve** または **-p** を使用します。

getext

自動モードが有効な場合に 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。この一覧を現在のセッション用に変更するには **setext** を使用します。既定の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。

help | **?** [*command*]

sftp ヘルプを表示します。指定のコマンドの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を一覧表示します。以下のオプションがあります。

-1 (1 列)
-a (隠しファイルを表示)
-f (並べ替えない)
-l (長い一覧形式)
-n (数値のユーザ ID とグループ ID を含む長い一覧形式)
-r (逆順)
-S (サイズ順に並べ替え)
-t (ファイルアクセス時刻順に並べ替え)
-- (後続のハイフンを通常の文字として処理)

lmkdir *directory*

指定されたローカルディレクトリを作成します。

ln

これは、**symlink** と同義です。

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を一覧表示します。オプションは **lls** の場合と同じです (上記を参照)。

mget

これは、**get** と同義です。

mkdir *directory*

指定されたリモートディレクトリを作成します。

mput

これは、**put** と同義です。

open [**-l** | [*user@*]*host*]

指定されたホストへの接続を開きます。ローカルホストへ接続するには **-l** を使用します。この場合は、ローカルコマンドとリモートコマンドのいずれもがローカルファイルシステム上のファイルに適用されます。

put [**--preserve**] | [**-p**] *local_file* [*local_file* ...]

指定されたファイルを現在のリモート作業ディレクトリへコピーします(別の場所へコピーするには、**cd** を使用してリモート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している場合は、その既存のファイルが上書きされます。ワイルドカード文字を使用できますが、名前の置換はファイル名でのみ行われ、ディレクトリでは行われません。ファイルの属性およびタイムスタンプを保持するには **--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、接続を閉じます。.

rename *source destination*

ファイルの名前を *source* から *destination* へ変更します。変更先のファイルがすでに存在する場合、名前の変更は行われません。

rm *file*

指定されたリモートファイルを削除します。ワイルドカード文字を使用できます。

rmdir *directory*

指定されたリモートディレクトリを削除します。

setext

自動モードが有効な場合に 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) 文字を使用できます。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、引用符で拡張子を囲むか、またはバックスラッシュをエスケープ文字として使用します。自動モードを有効にするには **auto** を使用します。現在の一覧を表示するには **getext** を使用します。既定の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。

symlink *linked_path target_path*

リモートホスト上の *linked_path* から *target_path* への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

使用できる SFTP プロトコルバージョンを表示します。

付 録　K

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 `$HOME/.ssh2/identification`) に指定したすべての鍵が追加されます。

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-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。鍵は指定され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 `$HOME/.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　L

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*]

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-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　M

sshd コマンドラインオプション

-4

接続で IPv4 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成することもできます。

-6

接続で IPv6 アドレスのみを使用するようにします。**AddressFamily** キーワードを使用して IP アドレス要件を構成することもできます。

-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)。 このオプションを指定すると、**sshd** は、1 つのクライアント接続のみの情報をログに記録し、そのクライアント接続が閉じると終了します。

-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。この設定は root のみが使用できます。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)。 このオプションを指定すると、**sshd** は、クライアント接続が閉じても終了しません。

出力ファイルは `/etc/ssh2` に保存され、`debugYYMMDD_HHMMSS`(YY は年、MM は月、DD は日、HH は時、MM は分、SS は秒) 形式のファイル名が使用されます。

-f *config_file*

サーバ構成ファイルの別の名前および場所を指定します。サーバでは、既定のファイルではなくこの指定のファイルが読み込まれます(既定の構成ファイルは `/etc/ssh2/sshd2_config` です)。

-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。

-h *host_key_file*

Specifies the filename and location of the private key used to authenticate the server. パスが完全に修飾されていない場合、そのパスは `/etc/ssh2` の相対パスとみなされます。既定では `/etc/ssh2/hostkey` になります。

-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。キーワードとそれらの意味の一覧については、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、構成ファイル内に設定されたオプションよりも優先されます。代替の構文を以下に示します。空白を含む式を指定する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。 このコマンドラインの値は、構成ファイルに設定された値よりも優先されます。1 つのポートだけが許可されます。複数のポートを構成している場合は、最後に構成したポートが使用されます。

-q

静的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(**-q** が同じコマンドラインに指定されている場合は、**-d** と **-D** の両方が無視されます)。

-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」を使用した場合と同じ動作になります。

-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。

付 録　N

ssh-certview コマンドリファレンス

SYNOPSIS

ssh-certview [**-C**] [**-c**] [**-d** *debug_level*] [**-h**] [**-q**] [**-V**] [**-v**] [*file ...*]

DESCRIPTION

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 でも使用されます。

注記: 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。

OPTIONS

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-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 (公開鍵のフィンガープリント) 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表示されます。

EXAMPLES

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　O

ssh-certtool コマンドリファレンス

SYNOPSIS

ssh-certtool [**-b** *key-length*] [**-d** *debug-level*] [**-h**] [**-n** *algorithm*] [**-o** *output-file*] [**-p** *private-key*] [**-V**] [*-z option*] **pkcs10** | **pkcs12** [*arguments*]

ssh-certtool [*options*] **pkcs10** *subject* [*keyUsage*] [*extendedKeyUsage*]

ssh-certtool [*options*] **pkcs12** [*file1*] ... [*fileN*]

DESCRIPTION

PKCS#10 証明書要求を作成する場合や、1 つの秘密鍵と 1 つ以上の証明書を含む PKCS#12 パッケージを作成する場合は、**ssh-certtool** を使用できます。

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*]

注記:　**req** には、**pkcs10** の同義語として対応しています。

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。サブジェクト名にスペースが含まれている場合は引用符が必要です。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。**-o** が指定されていない場合、生成された秘密鍵の既定のファイル名は `output.pkcs10` になります。

既存の秘密鍵を使用して要求を作成するには、**-p** を使用して鍵を指定します。要求用に新規の秘密鍵を生成するには、鍵の種類 (**-n**) か、鍵のサイズ (**-b**) のいずれか、またはその両方を指定する必要があります。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。**-o** が指定されていない場合、生成された秘密鍵の既定のファイル名は `output.ssh2` になります。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。**-o** が指定されていない場合、生成された PKCS#12 パッケージの既定のファイル名は output.p12 になります。PKCS#12 パッケージは HMAC によって保護されており、この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 *File* 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。この鍵がパスフレーズ保護されている場合は、**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 *File* 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。安全な暗号化の形式は、既定で pbeWithSHA1And40BitRC2-CBC になります。**-z** オプションを使用すると、各種の PBE 保護方式を構成できます。

OPTIONS

オプションは、1 文字形式 (**-o** など) およびそれと同等の記述形式 (**--option**) の両方で使用できます。ここでは 1 文字形式を示しています。同等の記述形式を表示するには、**-h** コマンドラインオプションを使用します。

-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 鍵の場合は 2048 ビット、DSA 鍵の場合は 1024 ビットになります。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。

-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 (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。

-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、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。

-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** または **Email** (大文字と小文字が区別される) のいずれか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。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。DNS オプションは DNS 代替名の拡張を設定し、Email オプションは EMail 代替名の拡張を設定します。これらの拡張は Critical (重要) とマーク付けされていません。

PKCS#12 パッケージの場合、*Key* は **KeyPBE** または **SafePBE** (大文字と小文字が区別される) のいずれか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 であり、**SafePBE** は PBE-SHA1-RC2-40 です。括弧内の長名は同義語です。

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)

EXAMPLES

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　P

PKI Services Manager コマンドリファレンス (winpki および pkid)

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*] [**-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。注記: 設定の中には再起動が必要なものがあります。

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、許可される ID についての情報を提供します。サービスは実行中でなければなりません。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
```

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 **-u**、**-t**、または **-w** 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。注記: 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。*path* によってディレクトリが指定されている場合、PKI Services Manager は、そのディレクトリ内のサーバ構成ファイル (`sshd2_config`) およびクライアント構成ファイル (`ssh2_config`) を検索して、それらのファイルから設定を移行します。*path* によってファイルが指定されている場合、PKI Services Manager は、指定されたファイル内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。注記: 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、PKI Services Manager で使用される `pki_config` ファイルおよび `pki_map` ファイルへ移行されます。**-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、PKI Services Manager データディレクトリ内にある `logs` ディレクトリに作成されます。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、**-d** を使用して高めることができます。

-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、*pki_config* を参照してください。設定可能な構文の例を以下に示します。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
```

-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　Q

PKI Services Manager 構成ファイルリファレンス (pkid_config)

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 XP、Windows Server 2003 の場合:
  `\Documents and Settings\all users\Application Data\Attachmate\ReflectionPKI\config\pki_config`
- Windows Vista、Windows Server 2008 の場合:
  `\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。設定への変更は、それらの設定が再度読み込まれるか、サービスが再起動される (再起動が必要な場合は、その情報がキーワードの記述に示されます) まで有効になりません。

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。注記: 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。一覧に示されている順番に場所が検索されます。オプションは「local」、「certserver」、「aia」、および「windows」です。既定は「local, certserver」です(注記: 「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。(`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、スタンザごとに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。(`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。注記: 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」、「**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 マネージャ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する 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) によって示されます。

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 アドレスは IPv4 または IPv6 形式のいずれかになります。IPv6 アドレスは、角括弧で囲む必要があります (`[::D155:AB63]:18081` など)。 既定は `0.0.0.0:18081` で、これにより、サーバは、使用可能な任意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。監査メッセージでは、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。ログファイルは毎日、新規作成されます。この最大数に達すると、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。このキーワードは、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 値（`OcspCertificate = "CN = Secure CA, O = Secure Corporation, C = US"` など）のいずれかになります。Subject 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（`http://ocsp.myhost.com: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。

PolicyOID

ExplicitPolicy が「yes」に設定されている場合に使用する、許可されるポリシ ID (OID) を指定します。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。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。注記: 「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」/「windows」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。

注記: ストア内の証明書の Subject (サブジェクト) フィールドを表示するには、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　R

PKI Services Manager マップファイルリファレンス (pki_mapfile)

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 XP、Windows Server 2003 の場合:
  `\Documents and Settings\all users\Application Data\Attachmate\ReflectionPKI\config\pki_mapfile`
- Windows Vista、Windows Server 2008 の場合:
  `\ProgramData\Attachmate\ReflectionPKI\config\pki_mapfile`

注記: Windows システムでは、Reflection PKI Services Manager コンソールの **[Identity Mapper] (ID マッパ)** 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。各ルールは単一行から成り、ほかのルールから独立しています。ルールの形式は以下のようになります。

{*Allowed-Identity*} [*Conditional Expression*]

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) 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。注記: 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。

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。個々の値を区切るには空白を使用します (許可される名前に空白が含まれる場合は、空白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" }
```

Windows ドメインユーザの場合は、以下の例のように、"*domain\user*" 形式を使用する必要があります。

```
{ windomain\joe "windomain\fred smith"}
```

注記: 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 (**RuleType** が **user-address** であるルール) でのみワイルドカードを使用します。
- 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% }
```

さらに、抽出された ID とリテラル文字列を組み合わせることもできます (抽出された ID の前にリテラル文字列を付加するか、リテラル文字列を追加することができますが、複数の抽出値を組み合わせて 1 つの ID を形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% }
```

注記: 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 [詳細] 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)

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、以下のようになります。

```
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。注記: 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) 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

Field には、対応している証明書フィールド (上記を参照) である Subject、Subject.CN、Subject.Email、DNS、UPN、UPN.User、UPN.Host、Email、Email.Host、SerialAndIssuer、Cert、subst のうちのいずれかを指定します。

Argument には文字列値を指定します。

Operation には、以下のいずれか 1 つを使用します。

Equals

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

Field 値が、*Argument* 文字列内の任意の場所に含まれてい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

Argument を正規表現として *Field* に適用します。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。アプリケーションをポイントするには *Argument* を使用します。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。外部アプリケーション内で一致に成功した場合、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。

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。形式は以下のようになります。

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。注記: 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。

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
```

注記: 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

Extern 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。

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 = *server*」です。既定は「none」です。

付 録　S

PKI Services Manager の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 (サブジェクト) 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。

ルール	説明
`{ 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、許可される ID のセットが「bob」になります。
`{ %subst% } Subject.CN Extern /bin/myapp`	PKI Services Manager は Subject (サブジェクト) フィールドの共通名 (CN) 部分をアプリケーション「/bin/myapp」に送信します。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　T

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 } CN Contains acme
{ joe } 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 } 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 CN=foo
```

付 録　U

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 UNIX Client および Server 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。

注記: **-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*『151 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。追加の情報については、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　V

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。これは内部エラーです。–d スイッチを指定して再実行し、動作を確認してください。
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)。

コード	意味
0	エラーなし。
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。
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。

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。注記: 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も呼ばれます。ハッシュ (またはハッシュ値) 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、元のデータより大幅にサイズが小さく、同じハッシュ値がほかのデータで統計的に生成されることのない方法で生成されます。

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。

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

F

G

H

K

M

P

R

S

X

あ

ま

や

ら