SSH.COM

Tectia® Client 6.6

ユーザ・マニュアル

2024年12月04日

Tectia® Client 6.6: ユーザ・マニュアル

2024年12月04日

SSH Communications Security Corporation
Karvaamokuja 2D, FI-00380 Helsinki, Finland

目次

第1章 本マニュアルについて

本マニュアルでは、Tectia Client のインストールと使用について説明します。本マニュアルは、Tectia Client ソフトウェアのインストールと設定を行うユーザ及び管理者を対象としています。

本マニュアルには以下の情報が記載されています。

- Tectia Client のインストール
- 使用開始
- 認証
- ファイルの転送
- トンネリング
- トラブルシューティング
- コマンドライン・ツール、GUI、監査メッセージ参照を含む付録

接続ブローカーは Tectia Client のあらゆる暗号操作と認証関連のタスクを処理します。また Tectia Client の設定は、XML ファイル、または A.1 で説明されている Tectia コネクション設定　GUI で行われた接続ブローカーの設定を介して行います。

Tectia Client の一般的な情報やその機能については、『Tectia Client/Server Product Description』を参照してください。

1.1. 本マニュアル中で使用される規則

Tectia のマニュアルでは、以下の表記規則が使用されています。

表1.1 マニュアルの規則表

規則	用途	例
太字	ツール、メニュー、GUI エレメント及びコマンド、コマンドライン・ツール、強調	**[適用]** または **[OK]** をクリックします。

規則	用途	例
→	連続するメニューの選択	**[ファイル]** → **[保存]** の順に選択します。
#	コマンドの前に付いた # は、そのコマンドが特権ユーザ (root) として実行されることを示します。	`# rpm --install package.rpm`
$	コマンドの前に付いた $ は、そのコマンドが特権のないユーザとして実行されることを示します。	`$ sshg3 user@host`
\	コマンド行の末尾にある \ は、その行を 1 行では表示できないため、次の行に続くことを示します。	`$ ssh-keygen-g3 -t rsa \` `  -F -c mykey`

注意

「注意」は、本文中の重要なポイントを強調または補足するための一般的な、または役に立つ情報を示します。またメモリの制限、装置の設定、またはプログラムの特定のバージョンなど、特殊な場合にのみ適用される情報を示します。

警告

「警告」はユーザに対して、特定の操作を行う、または避けることができなかった場合にデータが消失する可能性があることを示します。

1.1.1. オペレーティング・システムの名称

情報がオペレーティング・システムの複数のバージョンに適用される場合は、次の命名規則を使用します。

- 「**Unix**」は、サポートされる以下のオペレーティング・システムを指しています。
  - HP-UX
  - IBM AIX
  - Red Hat Linux、SUSE Linux
  - Linux on IBM System z
  - Solaris
  - IBM z/OS (Tectia Server for IBM z/OS が USS で動作中であり、Unix タイプのツールを使用している場合に該当。)

- 情報が IBM z/OS バージョンに直接関係する場合は、IBM z/OS に対して「**z/OS**」が使用されます。
- 「**Windows**」は、サポートされるすべての Windows バージョンを指しています。

1.1.2. ディレクトリ・パス

本マニュアルでは、ディレクトリ・パスを示すために以下の規則を使用しています。

`$HOME`

Unix の環境変数。ユーザのホーム・ディレクトリへのパスを示します。

`%APPDATA%`

Windows の環境変数。ユーザ固有の [Application Data] フォルダへのパスを示します。デフォルトでは以下のように展開されます。

"`C:\Users\<username>\AppData\Roaming`"

`%USERPROFILE%`

Windows の環境変数。ユーザ固有のプロファイル・フォルダへのパスを示します。デフォルトでは以下のように展開されます。

"`C:\Users\<username>`"

`<INSTALLDIR>`

Windows のデフォルトのインストール・ディレクトリを示します。

"`C:\Program Files (x86)\SSH Communications Security\SSH Tectia`" (64 ビット Windows バージョンの場合)

1.2. カスタマー・サポート

Tectia の製品マニュアルはすべて、 https://www.ssh.com/manuals/ で入手できます。

Tectia の全製品の使用方法をまとめた FAQ は http://answers.ssh.com/ でご覧いただけます。

メンテナンス契約をご購入された場合、SSH Communications Security からテクニカル・サポートを受けることができます。契約書で具体的な条件をご確認のうえ、https://support.ssh.com/ にログインしてください。

サポート・リクエスト、機能リクエスト、またはバグ・レポートの送信、及びオンライン・リソースへのアクセスに関する情報は、 https://support.ssh.com/ でご確認いただけます。

1.3. 関連用語

以下の用語が本マニュアル中で使用されます。

クライアント・コンピュータ
: Secure Shell 接続を開始するコンピュータ。

接続ブローカー
: 接続ブローカー は Tectia Client、Tectia ConnectSecure、及び Tectia Server for IBM z/OS クライアント・ツールに含まれているコンポーネントです。接続ブローカー はすべての暗号化動作及び認証関連のタスクを処理します。

FTP-SFTP変換
: Tectia ConnectSecure はクライアント上の FTP 接続を自動的にキャプチャして SFTP に変換し、Tectia Server、Tectia Server for IBM z/OS、または別のベンダの Secure Shell サーバ・ソフトウェアが動作する SFTP サーバに接続をダイレクトすることができます。

ホスト鍵ペア
: Secure Shell サーバを識別するための公開鍵ペア。秘密鍵ファイルにアクセスできるのはサーバのみです。公開鍵ファイルは、サーバに接続するユーザに配布されます。

リモート・ホスト
: 接続の対向側。見る側の視点によってクライアント・コンピュータまたはサーバ・コンピュータのいずれかを指します。

Secure Shell クライアント
: Secure Shell プロトコル・バージョン 2 を使用するクライアント側アプリケーション (Tectia Client の **sshg3**、**sftpg3**、**scpg3** など)。

Secure Shell サーバ
: Secure Shell プロトコル・バージョン 2 を使用するサーバ側アプリケーション。

サーバ・コンピュータ
: Secure Shell サービスが稼動し、Secure Shell クライアントが接続するコンピュータ。

SFTP サーバ
: Secure Shell サーバのサブシステムとして、セキュアなファイル転送サービスを提供するサーバ側アプリケーション。

Tectia Client
: ワークステーションにインストールされるソフトウェア・コンポーネント。Tectia Client は、セキュアな対話的ファイル転送とターミナル・クライアント機能を、Tectia Server またはその他の Secure Shell プロトコルを使用したアプリケーションが稼動しているサーバへのアクセスをリモート・ユーザやサーバの管理を行うシステム管理者に提供します。また、(非透過的) 静的トンネリングもサポートします。

Tectia client/server ソリューション
: Tectia client/server ソリューションはTectia Client、Tectia ConnectSecure、Tectia Server、及びTectia Server for IBM z/OS (Tectia Server for IBM z/OS クライアント・ツールを含む) で構成されています。

Tectia コネクション設定　GUI
: Tectia Client 及び ConnectSecure には、リモート・サーバへの接続設定を行うためのグラフィカル・ユーザ・インターフェイス (GUI) があります。この GUI は Windows と Linux に対応しています。

Tectia ConnectSecure
: サーバ・ホストにインストールされるソフトウェア・コンポーネント。ただしこれは、Secure Shell クライアントとして機能します。Tectia ConnectSecure は FTP の置き換え用に設計されており、FTP-SFTP変換、透過的FTPトンネリング、透過的TCPトンネリング、及び高速ファイル転送サービスを提供します。Tectia ConnectSecure は標準の Secure Shell サーバに接続することができます。

Tectia セキュア・ファイル転送 GUI
: Windows 上の Tectia Client 及び ConnectSecure には、ファイル転送を対話的に処理及び実行するための専用のグラフィカル・ユーザ・インターフェイス (GUI) があります。

Tectia SFTP API
: Tectia ConnectSecure には、C 及び Java 用の専用のアプリケーション・プログラミング・インターフェイス (API) があります。開発者はこれらの API を使用して、セキュアなファイル転送アプリケーションを開発したり、Tectia 製品を他のシステムに統合したりできます。

Tectia Server
: Tectia Server は Secure Shell クライアントの接続先であるサーバ側コンポーネントです。Linux、Unix、及び Windows プラットフォーム用の Tectia Server、Tectia Server for Linux on IBM System z、及び Tectia Server for IBM z/OS という 3 つのバージョンの Tectia Server 製品を利用できます。

Tectia Server for IBM z/OS
: Tectia Server for IBM z/OS は一般的な Secure Shell 接続を提供し、IBM メインフレーム上の高機能ファイル転送 (EFT) 機能と透過的TCPトンネリングをサポートします。

Tectia Server for Linux on IBM System z
: Tectia Server for Linux on IBM System z は、IBM System z プラットフォームで動作する Linux 上で Secure Shell 接続を提供します。

Tectia Server 設定ツール
: Tectia Server には、設定ファイルを編集する代わりに、サーバの設定に使用できるグラフィカル・ユーザ・インターフェイスがあります。この GUI は Windows に対応しています。

透過的FTPトンネリング
: Secure Shell トンネルによって透過的に暗号化され、セキュアになっている FTP 接続。

透過的TCPトンネリング
: Secure Shell トンネルによって透過的に暗号化され、セキュアになっている TCP アプリケーション接続。

トンネルされたアプリケーション
: Secure Shell 接続によってセキュアになっている TCP アプリケーション。

ユーザ鍵ペア
: Secure Shell ユーザを識別するための公開鍵ペア。秘密鍵ファイルにアクセスできるのはユーザのみです。公開鍵ファイルは、ユーザが接続するサーバにコピーされます。

第2章 Tectia Client のインストール

本章では、Tectia Client のインストール (及び削除) 手順についてサポートされているプラットフォームごとに説明するとともに、Tectia のファイルの場所についてまとめています。

2.1. インストールの準備

本項では、サポートされているプラットフォームと、Tectia Client のインストールに必要な前提条件についてまとめています。

2.1.1. システム要件

Tectia Client のプラットフォームとしてサポートされているオペレーティング・システムは以下の表の通りです。

表2.1 Tectia Client 及び Server がサポートするオペレーティング・システム

オペレーティング・システム	Client	Server
HP-UX (PA-RISC) [a]	11i v3	11i v3
HP-UX (IA-64) [a]	11i v3	11i v3
IBM AIX (POWER)	7.1, 7.2, 7.3	7.1, 7.2, 7.3
Oracle Solaris (SPARC)	10、11	10、11
Oracle Solaris (x86-64)	10、11	10、11
Red Hat Enterprise Linux (x86-64)	7, 8, 9	7, 8, 9
Rocky Linux (x86-64)	8, 9	8, 9
Ubuntu (x86-64)	18.04, 20.04, 22.04	18.04, 20.04, 22.04
Debian GNU/Linux (x86-64)	11, 12	11, 12
SUSE LINUX Enterprise Desktop (x86-64)	15	15
SUSE LINUX Enterprise Server (x86-64)	12、15	12、15
Microsoft Windows (x86-64)	10, 11, Server 2016, Server 2019, Server 2022	10, 11, Server 2016, Server 2019, Server 2022

[a] このプラットフォームでは PQC はサポートされません。

注意

オペレーティング・システム・ベンダの推奨に従って、常にオペレーティング・システムにパッチを完全に適用してください。

サポートされているオペレーティング・システムには、以下に示す、またはそれに取って代わるパッチまたはメンテナンス・レベルがインストールされている必要があります。Tectia のソリューションは、以下のパッチ及びメンテナンス・レベルでテストされています。

HP-UX patches

OS バージョンに応じた最新の **HP-required patch bundle** をインストールするのが一般的です。また、Tectia ソフトウェアが正しく機能するためには、libc、pthread、及びリンカ・ツールに対する最新の **HP recommended patches**が必要です。さらに、特定の問題を解決するために個別のパッチが必要な場合もあります。そのようなパッチは別途記載します。

注意

ここにリストアップされているものより新しいパッチが存在するかどうか、HP 社のウェブ・サイトで確認してください。HP 社が推奨する最新バージョンをインストールすることをお勧めします。

- HP-UX 11i v3 on PA-RISC 及び IA-64 (Itanium):
  - PHCO_36551 pthread ライブラリ累積パッチ (2007 年 5 月)
  - PHSS_37202 リンカ及び fdp 累積パッチ (2007 年 10 月)
  - Kerberos Client D.1.6.2 (2007 年 12 月)

2.1.2. ハードウェア及び必要なディスク空き領域

Tectia Client には特別なハードウェアの要件はありません。ここに記載されているオペレーティング・システムの最新バージョンが動作し、ネットワーク接続が機能するものであれば、どのようなコンピュータでも使用できます。

Tectia Client のインストールには約 100 メガバイトのディスク空き容量が必要です。

Tectia Client では、個々のユーザの設定は、そのユーザの個人ディレクトリに保存されます。

2.1.3. ライセンス

Tectia Client が機能するためには、ライセンスが必要です。 ライセンス・ファイル名は `stc66.dat` です。

ライセンスに関する以下の点を、どのプラットフォーム用の Tectia Client を購入したかに応じて考慮してください。

- オンライン・インストール・パッケージでは、ライセンス・ファイルは、リリース・ノート (`.txt`) ファイルや PDF 形式のマニュアルとともに圧縮ファイル (`.zip`/`.tar`) に含まれています。
- Tectia の評価版パッケージにライセンス・ファイルは含まれていません。評価版は、ライセンス・ファイルなしで 45 日間使用できます。 Unix 及び Windows マシンの場合、 ユーザにライセンスの期限切れまでの日数を示すバナー・メッセージが表示されます。
- 評価版または標準商用バージョンから Tectia Quantum Safe Edition へのアップグレードは、ライセンスファイルをライセンス・ディレクトリにコピーし、 Tectia Client ソフトウェアを再起動するだけです。

2.1.4. インストール・パッケージ

Tectia Client のインストール・パッケージはインストール・バンドルに圧縮されています。サポートされているオペレーティング・システムごとに、 Tectia Quantum Safe Edition 製品版 (`-comm-pqc`) 、製品版 (`-comm`) およびアップグレード及び評価版 (`-upgrd-eval`) の 3 つのバンドルがあります。すでに適切なライセンスを所有している場合、評価版はアップグレード・パッケージとして使用できます。

適切な Tectia Client のバンドルを選択してください。

- AIX プラットフォーム:

```
tectia-client-<version>-aix-6-7-powerpc-comm-pqc.tar
tectia-client-<version>-aix-6-7-powerpc-comm.tar
tectia-client-<version>-aix-6-7-powerpc-upgrd-eval.tar
```

- HP-UX PA-RISC プラットフォーム:

```
tectia-client-<version>-hpux-11iv2-3-hppa-comm-pqc.tar
tectia-client-<version>-hpux-11iv2-3-hppa-comm.tar
tectia-client-<version>-hpux-11iv2-3-hppa-upgrd-eval.tar
```

- HP-UX Itanium プラットフォーム:

```
tectia-client-<version>-hpux-11i-ia64-comm.tar
tectia-client-<version>-hpux-11i-ia64-upgrd-eval.tar
```

- Linux 64 ビット・プラットフォーム (Red Hat Enterprise Linux、 Rocky Linux 及び SUSE Linux):

```
tectia-client-<version>-linux-x86_64-comm-pqc.tar
tectia-client-<version>-linux-x86_64-comm.tar
tectia-client-<version>-linux-x86_64-upgrd-eval.tar
```

- inux 64 ビット・プラットフォーム (Ubuntu 及び Debian GNU/Linux):

```
tectia-client-<version>-linux-ubuntu-x86_64-comm-pqc.tar
tectia-client-<version>-linux-ubuntu-x86_64-comm.tar
tectia-client-<version>-linux-ubuntu-x86_64-upgrd-eval.tar
```

- Solaris SPARC プラットフォーム (Solaris 10 用と 11 用に別々のパッケージがあります):

```
tectia-client-<version>-solaris-10-sparc-comm-pqc.tar
tectia-client-<version>-solaris-10-sparc-comm.tar
tectia-client-<version>-solaris-10-sparc-upgrd-eval.tar
tectia-client-<version>-solaris-11-sparc-comm-pqc.tar
tectia-client-<version>-solaris-11-sparc-comm.tar
tectia-client-<version>-solaris-11-sparc-upgrd-eval.tar
```

- Solaris x86-64 プラットフォーム (Solaris 10 用と 11 用に別々のパッケージがあります):

```
tectia-client-<version>-solaris-10-x86_64-comm-pqc.tar
tectia-client-<version>-solaris-10-x86_64-comm.tar
tectia-client-<version>-solaris-10-x86_64-upgrd-eval.tar
tectia-client-<version>-solaris-11-x86_64-comm-pqc.tar
tectia-client-<version>-solaris-11-x86_64-comm.tar
tectia-client-<version>-solaris-11-x86_64-upgrd-eval.tar
```

- Windows プラットフォーム:

```
tectia-client-<version>-windows-comm-pqc.zip
tectia-client-<version>-windows-comm.zip
tectia-client-<version>-windows-upgrd-eval.zip
```

`<version>` は、製品のリリース番号及び現在の Build 番号を表します (`6.6.5.123` など)。

Tectia Client の実際のインストール・パッケージはインストール・バンドルに含まれています。ご使用の環境で使用する製品機能に応じて、インストールするパッケージを選択してください。

Unix 及び Linux プラットフォームでは、Tectia Client は 3 つのインストール・パッケージで提供されます。

- `ssh-tectia-common` パッケージには Tectia Client 及び Server の共通コンポーネントが含まれています。
- `ssh-tectia-client` パッケージには Tectia Client 固有のコンポーネントが含まれています。
- オプションの `ssh-tectia-guisupport` パッケージには、Linux プラットフォームで使用可能な GUI に必要なコンポーネントが含まれています。

Windows では、Tectia Client は 1 つの MSI インストール・パッケージで提供されます。インストール・ウィザードのガイドに従ってインストールするコンポーネントを選択してください。

2.1.5. 以前にインストールした Tectia Client ソフトウェアのアップグレード

注意

アップグレードを開始する前に、変更を加えたすべての設定ファイルのバックアップを取ってください。A.3 の手順を参照してください。

同じマシン上で Tectia Client と Tectia Server の両方を実行している場合、共通コンポーネント間に依存関係があるため、各 Tectia 製品の同じリリースをインストールしてください。

古いバージョンの Tectia 製品や OpenSSH サーバまたはクライアントなどの Secure Shell ソフトウェアが、Tectia の新しいバージョンをインストールする予定のマシン上で動作しているかどうか確認してください。

Unix プラットフォームに Tectia Server をインストールする前に、ポート番号 22 で動作している OpenSSH サーバを停止するか、またはリスナ・ポートを変更してください。OpenSSH ソフトウェアをアンインストールする必要はありません。

SuSE でアップグレードする場合は、前提条件となるパッケージもインストールして下さい。

```
# zypper install insserv-compat
```

どの Tectia バージョンが Tectia Client 6.6 にアップグレードする前にアンインストールする必要があるのかを以下の表に示します。「上書きアップグレード」と示されているバージョンをアップグレードする場合、以前のバージョンはアップグレード中に自動的に削除されます。

表2.2 アップグレードの方法

Tectia のバージョン	AIX	HP-UX	Linux	Solaris	Windows
4.x	削除	削除	削除	削除	削除
5.x ～ 6.0	上書きアップグレード	上書きアップグレード	上書きアップグレード	削除	削除
6.1 ～ 6.6	上書きアップグレード	上書きアップグレード	上書きアップグレード	削除	上書きアップグレード、または透過的 TCP トンネリングがインストールされている場合は削除

設定ファイルのフォーマットとファイルの場所が Tectia Client 5.0 から、また Unix 版で DTD ディレクトリがバージョン 6.2 から変更になりました。 このため、4.x から、5.x ～ 6.x から、および 6.2 とそれ以降からのアップグレードで設定ファイルの動作が異なります。

- 6.2 ～ 6.x の設定ファイルは 6.6 にそのまま移行され、自動的に同じ場所で使用されます。

注意

暗号、MAC、鍵交換など、すべての明示的な変更はアップグレードしても保持されます。 これにより、鍵交換やホスト鍵または公開鍵署名アルゴリズムの SHA-1 など安全ではないアルゴリズムが混入することがあります。 さらに、例えば

Tectia Quantum Safe Edition ライセンスを必要とする Post Quantum Cryptography（PQC）ハイブリッド鍵交換アルゴリズムは、 Tectia バージョン 6.5 およびそれ以前からアップグレードした時に明示的な鍵交換設定が必要になります。 その代わりに、すべての鍵交換などの明示的な設定を削除することで 6.6 にデフォルトまたは PQC ハイブリッド鍵交換を強制することができます。

- 5.x ～ 6.1 の設定ファイルは 6.6 にそのまま移行され、Windows プラットフォームでは自動的に同じ場所で使用されます。

注意

暗号、MAC、鍵交換など、すべての明示的な変更はアップグレードしても保持されます。 これにより、安全ではないアルゴリズムが混入することがあります。 Tectia 6.1 およびそれ以前の Unix 版では、デフォルトの補助データ・ディレクトリ `auxdata` は `/etc/ssh2/ssh-tectia/` にありました。 Tectia Server 設定ファイル (`ssh-server-config.xml`) または Tectia Client 設定ファイル (`ssh-broker-config.xml`) が Tectia バージョン 6.1 またはそれ以前で作成された場合、 DOCTYPE 宣言をアップデートして現在のサーバ設定ファイル DTD ディレクトリ `/opt/tectia/share/auxdata/ssh-server-ng/` または 接続ブローカー 設定ファイル DTD ディレクトリ `/opt/tectia/share/auxdata/ssh-broker-ng/` へのパスが含まれるようにして下さい。

- 4.x の設定ファイルは 6.6 に移行されず、デフォルトの 6.6 の設定が使用されます。ただし Windows プラットフォームでは、接続プロファイルが 4.x から 6.6 に移行されます。

必要に応じて、Tectia コネクション設定 GUI を使用するか、アスキー・テキスト・エディタまたは XML エディタで XML 設定ファイルを手動で編集して、設定ファイルを変更できます。 Tectia Server には `ssh-server-config-example.xml` を、 Tectia Client には `ssh-broker-config-example.xml` のサンプル・ファイルを参照して下さい。

透過的 TCP トンネリングのオプションをインストールしている場合は、6.6 Tectia Client にアップグレードする前に、以前のバージョンのクライアントをアンインストールし、アンインストールの完了後にコンピュータを再起動してください。2.3.6 を参照してください。

注意

Tectia Client のバージョン 6.2.5 以降、透過的 TCP トンネリングのオプションは Windows のインストール・パッケージに含まれなくなりました。透過的 TCP トンネリングは Tectia ConnectSecure で使用できます。

Windows では、以前の Tectia Client の設定ファイルのバックアップ・コピーが自動的に作成され、以下のユーザ固有のディレクトリに保存されます。

"`%APPDATA%\SSH\backup-<version>-<date>`"

ここで `<version>` は Tectia のリリース・バージョン、`<date>` はアップグレードの日付です。

2.1.6. Tectia リリースのダウンロード

すべてのリリースに、インストール・パッケージと一緒に提供される商用ライセンスが必要です。

SSH のカスタマ・ダウンロード・センタから Tectia ソフトウェアをダウンロードするには、以下の手順に従ってください。

1. 次の URL からカスタマ・ダウンロード・センタにログインします。https://my.ssh.com
2. SSH ダウンロードから [**Tectia Client**] を選択し、適切なバージョンを選択します。Tectia 製品はメジャー・リリース、マイナ・リリース、及びメンテナンス・リリースで公開されています。
   - メジャー・リリースは 1 桁目で表されます (例: 6)。メジャー・リリースでは、以前のバージョンの修正に加えて、新製品や既存製品に対する重要な新機能が公開されます。
   - マイナ・リリースはリリース番号の 2 桁目で表されます (例: 6.6)。マイナ・リリースでは新機能や以前のバージョンの修正が公開されます。
   - メンテナンス・リリースはバージョン番号の 3 桁目です (例: 6.6.5)。メンテナンス・リリースでは以前のバージョンの修正が提供され、新機能は提供されません。メンテナンス・リリースは、保守サポート契約を締結しているお客様に提供されます。
3. 適切な製品バージョンとプラットフォームのリンクをクリックすると、圧縮されたインストール・パッケージが、お使いのマシンのデフォルトのダウンロード・フォルダにダウンロードされます。
4. インストールに進みます。後述する Tectia Client のプラットフォーム別インストール手順を参照してください。

2.2. Tectia Client ソフトウェアのインストール

本項では、サポートされているオペレーティング・システムで、Tectia Client をローカルにインストールする手順について説明します。

2.2.1. AIX でのインストール

ダウンロードしたインストール・パッケージには、圧縮されたインストール・ファイルが含まれています。

パッケージは、Tectia Client 及び Server の共通コンポーネントと Tectia Client の固有コンポーネントの 2 つが必要です。

AIX で Tectia Client をインストールするには、以下の手順に従ってください。

1. ダウンロードした `tar` パッケージを解凍します。
2. 以下のコマンドでインストール・パッケージを解凍します。

```
$ uncompress ssh-tectia-common-<version>-aix-6-7-powerpc.bff.Z
```

```
$ uncompress ssh-tectia-client-<version>-aix-6-7-powerpc.bff.Z
```

上記のコマンドの <version> は、Tectia Client の現在のパッケージ・バージョンです (6.6.5.123 など)。

3. root 権限で以下のコマンドを実行して、パッケージをインストールします。

```
# installp -d ssh-tectia-common-<version>-aix-6-7-powerpc.bff SSHTectia.Common
# installp -d ssh-tectia-client-<version>-aix-6-7-powerpc.bff SSHTectia.Client
```

4. ライセンス・ファイルを/etc/ssh2/licenses のディレクトリにコピーします。(「3 桁目」のメンテナンス・アップデートではこの操作は必要ありません。)2.1.3 も参照してください。

2.2.2. HP-UX でのインストール

オペレーティング・システムのパッチが完全に適用されていることを確認してください。Hewlett-Packard 社のウェブ・サイトで、最新のパッチ情報を確認してください。HP-UX プラットフォームで PAM/Kerberos を使用する場合は、Kerberos に関連する最新パッチもインストールしてください。

ダウンロードしたインストール・パッケージには、圧縮されたインストール・ファイルが含まれています。

パッケージは、Tectia Client 及び Server の共通コンポーネントと Tectia Client の固有コンポーネントの 2 つが必要です。

HP-UX で Tectia Client をインストールするには、以下の手順に従ってください。

1. ダウンロードした tar パッケージを解凍します。

2. HP-UX のバージョンに応じてインストール・パッケージを選択します。

   PA-RISC アーキテクチャで動作する 11i v3 (11.31) にインストールする場合、以下の名前のパッケージを使用します。

```
ssh-tectia-common-<version>-hpux-11iv2-3-hppa.depot.Z
ssh-tectia-client-<version>-hpux-11iv2-3-hppa.depot.Z
```

   Itanium アーキテクチャで動作する 11i v3 にインストールする場合、以下の名前のパッケージを使用します。

```
ssh-tectia-common-<version>-hpux-11i-ia64.depot.Z
ssh-tectia-client-<version>-hpux-11i-ia64.depot.Z
```

   上記のコマンドの <version> は、製品のリリース番号及び現在の Build 番号を表します (6.6.5.123 など)。

3. uncompress でパッケージを解凍します。インストールするためには、作成されたパッケージは正しい長いファイル名である必要があります。以下のコマンド例では Itanium バージョンを使用しています。

```
$ uncompress ssh-tectia-common-<version>-hpux-11i-ia64.depot.Z
$ uncompress ssh-tectia-client-<version>-hpux-11i-ia64.depot.Z
```

4. root 権限で以下のコマンドを実行して、パッケージをインストールします。

```
# swinstall -s <path>/ssh-tectia-common-<version>-hpux-11i-ia64.depot SSHG3common
# swinstall -s <path>/ssh-tectia-client-<version>-hpux-11i-ia64.depot SSHG3client
```

注意

上記のコマンドの <path> は、インストール・パッケージのフル・パスです。HP-UX では、同じディレクトリでコマンドを実行する場合でもフル・パスが必要です。

5. ライセンス・ファイルを/etc/ssh2/licenses のディレクトリにコピーします。(「3 桁目」のメンテナンス・アップデートではこの操作は必要ありません。)

2.2.3. Linux でのインストール (RPM)

Linux プラットフォーム用の Tectia Client は、x86-64 プラットフォーム・アーキテクチャで動作する Red Hat Enterprise Linux、 Rocky Linux 及び SUSE Linux 用の RPM (Red Hat Package Manager) バイナリ・パッケージで提供されます。

ダウンロードしたインストール・パッケージには、RPM のインストール・ファイルが含まれています。パッケージは常に、Tectia Client 及び Server の共通コンポーネント用と Tectia Client の固有コンポーネント用の 2 つが必要です。製品をグラフィカル・ユーザ・インターフェイス (GUI) で使用する場合は、オプションの GUI サポート・パッケージもインストールしてください。

Linux で Tectia Client をインストールするには、以下の手順に従ってください。

1. ダウンロードした tar パッケージを解凍します。

2. Linux のアーキテクチャに応じてインストール・パッケージを選択します。

   64 ビット x86-64 アーキテクチャで動作する Red Hat Enterprise Linux、 Rocky Linux 及び SUSE Linux バージョンにインストールする場合、以下の名前のパッケージを使用します。

   ```
   ssh-tectia-common-<version>-linux-x86_64.rpm
   ssh-tectia-client-<version>-linux-x86_64.rpm
   ssh-tectia-guisupport-<version>-linux-x86_64.rpm
   ```

   上記のコマンドの <version> は、製品のリリース番号及び現在の Build 番号を表します (6.6.5.123 など)。

3. root 権限でパッケージをインストールします。

   ```
   # rpm -ivh ssh-tectia-common-<version>-linux-x86_64.rpm
   # rpm -ivh ssh-tectia-client-<version>-linux-x86_64.rpm
   ```

```
# rpm -ivh ssh-tectia-guisupport-<version>-linux-x86_64.rpm
```

または、既に古いバージョンの Tectia Client がインストールされている場合はパッケージをアップグレードします。

```
# rpm -Uvh ssh-tectia-common-<version>-linux-x86_64.rpm
# rpm -Uvh ssh-tectia-client-<version>-linux-x86_64.rpm
# rpm -Uvh ssh-tectia-guisupport-<version>-linux-x86_64.rpm
```

4. ライセンス・ファイルを `/etc/ssh2/licenses` ディレクトリにコピーします。(「3 桁目」のメンテナンス・アップデートではこの操作は必要ありません。)2.1.3 も参照してください。

2.2.4. Linux (DEB) でのインストール

64ビット x86-64 アーキテクチャで動作する Ubuntu 及び Denian のための Debian GNU/Linux 用 Tectia Client は Debian (DEB) バイナリ・パッケージで提供されます。

Tectia インストール・バンドルには DEB インストール・ファイルとライセンス・ファイルが含まれています。

Debian Linux で Tectia Client をインストールするには、以下の手順に従ってください。

1. ライセンス・タイプに応じてインストール・バンドルをダウンロードしてください。

   - 商用 Tectia Quantum Safe Edition ライセンス:

     ```
     tectia-client-<version>-linux-ubuntu-x86_64-comm-pqc.tar
     ```

   - 商用ライセンス:

     ```
     tectia-client-<version>-linux-ubuntu-x86_64-comm.tar
     ```

   - 評価:

     ```
     tectia-client-<version>-linux-ubuntu-x86_64-upgrd-eval.tar
     ```

   パッケージ名の `<version>` は、例えば `6.6.5.123-1` のようにリリース・バージョンとビルド番号に対応します。

2. ダウンロードした `tar` パッケージを解凍します。

3. インストールするパッケージを選択します。共通パッケージの Tectia Client 及び Server と 固有コンポーネントの Tectia Client の2つは常に必須です。

   ```
   ssh-tectia-common-<version>_linux-x86_64.deb
   ssh-tectia-client-<version>_linux-x86_64.deb
   ```

4. root 特権でパッケージをインストールします。

   ```
   # dpkg -i ssh-tectia-common-<version>_linux-x86_64.deb
   # dpkg -i ssh-tectia-client-<version>_linux-x86_64.deb
   ```

5. ライセンス・ファイルを `/etc/ssh2/licenses` のディレクトリにコピーします。(「3 桁目」のメンテナンス・アップデートではこの操作は必要ありません。)

2.2.5. Solaris でのインストール

ダウンロードしたインストール・パッケージには、圧縮されたインストール・ファイルが含まれています。

パッケージは、Tectia Client 及び Server の共通コンポーネントと Tectia Client の固有コンポーネントの 2 つが必要です。

Tectia Client は Solaris 10 及び 11 上のゾーンをサポートしています。Tectia のソフトウェアはグローバル・ゾーンとローカル・ゾーンにインストールできます。Tectia のソフトウェアをグローバル・ゾーンにインストールすると、既存のローカル・ゾーンにも自動的にインストールされます。ただし、ローカル・ゾーンを後からシステムに追加する場合は、別途ローカル・ゾーンに Tectia Client をインストールする必要があります。

Tectia Client をスパース・ゾーンにインストールする場合、インストール・プロセスはシンボリックリンクの作成に失敗したことを報告することに注意してください。実際のインストールは正常に終了していますが、パス設定に `/opt/tectia/bin` を手動で追加する必要があります。

Solaris ゾーンの詳細については、Oracle のマニュアル『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照してください。

Solaris で Tectia Client をインストールするには、以下の手順に従ってください。

1. ダウンロードした `tar` パッケージを解凍します。

2. Solaris のバージョンに応じてインストール・パッケージを選択します。

   SPARC アーキテクチャで動作する Solaris バージョン 10 にインストールする場合、以下の名前のパッケージを使用します。

   ```
   ssh-tectia-common-<version>-solaris-10-sparc.pkg.Z
   ssh-tectia-client-<version>-solaris-10-sparc.pkg.Z
   ```

   SPARC アーキテクチャで動作する Solaris バージョン 11 にインストールする場合、以下の名前のパッケージを使用します。

   ```
   ssh-tectia-common-<version>-solaris-11-sparc.pkg.Z
   ssh-tectia-client-<version>-solaris-11-sparc.pkg.Z
   ```

   x86-64 アーキテクチャで動作する Solaris バージョン 10 または 11 にインストールする場合、以下の名前のパッケージを使用します。

   ```
   ssh-tectia-common-<version>-solaris-<solaris-version>-x86_64.pkg.Z
   ssh-tectia-client-<version>-solaris-<solaris-version>-x86_64.pkg.Z
   ```

   上記のコマンドの `<version>` は、製品のリリース番号及び現在の Build 番号を表します (`6.6.5.123` など)。 `<solaris-version>` は、x86-64 アーキテクチャにインストールする場合の Solaris のバージョン番号 (10 または 11) です。

3. インストール・パッケージを適当な場所に解凍します。Solaris 環境では `/var/spool/pkg` が標準の場所です。以下のコマンド例では Solaris 10 x86-64 を使用します。

```
$ uncompress ssh-tectia-common-<version>-solaris-10-x86_64.pkg.Z
$ uncompress ssh-tectia-client-<version>-solaris-10-x86_64.pkg.Z
```

4. root 権限で `pkgadd` ツールを使用して、パッケージをインストールします。

```
# pkgadd -d ssh-tectia-common-<version>-solaris-10-x86_64.pkg all
# pkgadd -d ssh-tectia-client-<version>-solaris-10-x86_64.pkg all
```

5. ライセンス・ファイルを`/etc/ssh2/licenses` のディレクトリにコピーします。(「3 桁目」のメンテナンス・アップデートではこの操作は必要ありません。)

2.2.6. Windows でのインストール

Windows のインストール・パッケージは 64 ビット (x86-64) プラットフォーム・アーキテクチャで動作する Microsoft Windows バージョン用に、 MSI (Windows インストーラ) 形式で提供されます。

ダウンロードしたインストール・パッケージは、ライセンス・ファイルと実行可能な MSI (Windows インストーラ) パッケージが含まれている zip ファイルです。

インストールは標準的なインストール・ウィザードによって実行されます。ウィザードによってユーザは情報の入力を求められ、プログラム・ファイルがコピーされ、クライアントが設定されます。

注意

Tectia Client と Tectia Server の両方を同じマシンにインストールする場合は、Tectia Server のインストーラ `ssh-tectia-server-<version>-windows-<platform>.msi` を使用して両方の製品をインストールする必要があります。ここで、`<version>` は Tectia Client/Server のリリース・バージョンと Build 番号 (`6.6.5.123` など)、`<platform>` はプラットフォーム・アーキテクチャ (64 ビット Windows バージョンでは `x86_64`) を表します。

以前インストールした Tectia Client をアップグレードする場合は、まず2.1.5 を参照してください。

Windows で Tectia Client をインストールするには、以下の手順に従ってください。

1. インストール用の zip ファイルの内容を一時的な場所に解凍します。

2. 適切な Windows インストーラ・ファイル `ssh-tectia-client-<version>-windows-<platform>.msi` を特定します。ここで

   - `<version>` は `6.6.5.123` など、Tectia Client/Server のリリース・バージョン及び Build 番号を表します。

- `<platform>` はプラットフォーム・アーキテクチャを表し、 `x86_64` は 64 ビット Windows バージョン用です。

3. インストール・ファイルをダブルクリックすると、インストール・ウィザードが起動します。

注意

`.msi` インストーラを実行する前に `.zip` パッケージの内容を解凍すると、ライセンス・ファイルが自動的にインポートされます。

`.msi` インストーラを `.zip` パッケージから直接実行する場合は、インストールが完了した後でライセンス・ファイルの `stc66.dat` を手動でインストールする必要があります。インストール・ウィザードに、ライセンス・ファイルが見つからないというエラー・メッセージが表示され (下図を参照)、Tectia Client を起動しようとすると、ライセンスを次の正しいディレクトリに手動でインストールするよう要求されます。

- "`C:\Program Files (x86)\SSH Communications Security\SSH Tectia\SSH Tectia AUX \licenses`" (64 ビット Windows バージョンの場合)

図2.1 ライセンス・ファイルが見つからない場合の警告

Windows 10 では、ブラウザ経由でダウンロードした Tectia のパッケージによって「Windows によって PC が保護されました」の警告メッセージが表示されることがあります。そのような場合は、「**詳細情報**」及び「**実行**」をクリックしてインストールを進めてください。

4. インストールの各手順でウィザードの指示に従い、必要な情報を入力してください。

5. Tectia Client の [**標準**] インストールには、 **sshg3.exe**、 **scpg3.exe**、及び **sftpg3.exe** のコマンドライン・ツールと、ターミナルとファイル転送用のグラフィカル・ユーザ・インターフェイスが含まれています。

すべてのコンポーネントをインストールするには、ウィザードでセットアップの種類を選ぶときに、[**すべて**] を選択します。

インストールするコンポーネントを選ぶ場合は、ウィザードでセットアップの種類を選ぶときに、[**カスタム**] を選択します。続いて表示されるダイアログボックスで、一部のコンポーネントをインストールの対象外に指定できます。図 2.2 を参照してください。

図2.2 Tectia Client のインストール・オプション

6. インストールが終了したら、[**完了**] をクリックしてウィザードを終了します。

7. Tectia Client をインストールした後は、コンピュータを再起動する必要があります。[**はい**] をクリックして再起動します。

デフォルトのインストール・ディレクトリは以下の通りです。

- "`C:\Program Files (x86)\SSH Communications Security\SSH Tectia`" (64 ビット Windows バージョンの場合)

サイレント・インストール

Tectia Client は、ワークステーションにサイレントにインストールすることもできます。サイレント (非対話型) インストールとは、インストール時にユーザ・インターフェイスが表示されず、ユーザに質問することもないインストール方式です。このオプションではリモート操作による自動インストールができるので、特にシステム管理者にとって便利です。

サイレント・モードでは、Tectia Client はデフォルトの設定で、追加機能なしでインストールされます。

Tectia Client をサイレントにインストールするには、以下のコマンドを使用します。

```
msiexec /q /i ssh-tectia-client-<version>-windows-<platform>.msi INSTALLDIR="<path>"
```

コマンドの中の、

- `<version>` は、Tectia Client の現在のバージョンを表します (`6.6.5.123` など)。
- `<platform>` はプラットフォーム・アーキテクチャを表し、`x86_64` は 64 ビット Windows バージョン用です。
- `<path>` は、目的のインストール・ディレクトリへのパスです。`INSTALLDIR` 変数を省略すると、Tectia Client はデフォルトの場所にインストールされます。

デスクトップ・アイコン

インストール中に、Tectia のアイコンがデスクトップに追加されます。Tectia の SSH ターミナル・ウィンドウとセキュア・ファイル転送ウィンドウには、それぞれ別のプログラム・アイコンがあります。どちらのアイコンも同じアプリケーション (**ssh-client-g3.exe**) を起動します。前者のアイコンはターミナル・ウィンドウを起動し、後者のアイコンはファイル転送ウィンドウを起動します。

図2.3 Tectia SSHターミナル GUI のアイコン

図2.4 Tectia セキュア・ファイル転送 GUI のアイコン

2.3. Tectia Client ソフトウェアの削除

本項では、サポートされているオペレーティング・システムから Tectia Client を削除する手順について説明します。

注意

このアンインストール手順は、ソフトウェアのインストール時に作成されたファイルだけが削除されます。設定ファイルは手動で削除する必要があります。

2.3.1. AIX からの削除

AIX 環境から Tectia Client を削除するには、以下の手順に従ってください。

1. root 権限で以下のコマンドを実行して、インストールを削除します。

```
# installp -u SSHTectia.Client
```

2. Tectia Server の共通コンポーネントも削除する場合は、以下のコマンドを実行します。

```
# installp -u SSHTectia.Common
```

共通コンポーネントを削除すると、同じホストにインストールされている Tectia Server が無効になることに注意してください。

2.3.2. HP-UX からの削除

HP-UX 環境から Tectia Client を削除するには、以下の手順に従ってください。

1. root 権限で以下のコマンドを実行して、インストールを削除します。

```
# swremove SSHG3client
```

2. Tectia Server の共通コンポーネントも削除する場合は、以下のコマンドを実行します。

```
# swremove SSHG3common
```

共通コンポーネントを削除すると、同じホストにインストールされている Tectia Server が無効になることに注意してください。

2.3.3. Linux からの削除 (RPM)

Linux 環境で RPM からインストールした Tectia Client を削除するには、以下の手順に従ってください。

1. root 権限で以下のコマンドを実行して、インストールを削除します。

```
# rpm -e ssh-tectia-client
```

2. Tectia Server の共通コンポーネントも削除する場合は、以下のコマンドを実行します。

```
# rpm -e ssh-tectia-common
```

3. GUI コンポーネントを削除するには、root 権限で以下のコマンドを実行します。

```
# rpm -e ssh-tectia-guisupport
```

2.3.4. Linux からの削除 (DEB)

Debian Linux 環境から Tectia Client を削除するには、以下の手順に従ってください。

1. root 権限で以下のコマンドを実行して、インストールを削除します。

```
# dpkg -P ssh-tectia-client
```

2. Tectia Server の共通コンポーネントも削除する場合は、以下のコマンドを実行します。

```
# dpkg -P ssh-tectia-common
```

2.3.5. Solaris からの削除

Solaris 環境から Tectia Client を削除するには、以下の手順に従ってください。

1. root 権限で以下のコマンドを実行して、インストールを削除します。

```
# pkgrm SSHG3clnt
```

2. Tectia Server の共通コンポーネントも削除する場合は、以下のコマンドを実行します。

```
# pkgrm SSHG3cmmn
```

共通コンポーネントを削除すると、同じホストにインストールされている Tectia Server が無効になることに注意してください。

2.3.6. Windows からの削除

Windows から Tectia Client のインストールを削除するには、いくつかの方法があります。以下のいずれかの手順に従ってください。

Windows のコントロール・パネルのツールを使用する

1. Windows の [**スタート**] メニューから、[**コントロール パネル**] を開き、[**プログラムと機能**] をクリックします。
2. インストール済みプログラムの一覧から [**Tectia Client**] を選択し、[**アンインストール**] をクリックします。
3. [**はい**] をクリックして確認します。

Windows インストーラーを使用する

1. Windows インストーラ・ファイル `ssh-tectia-client-<version>-windows-<platform>.msi` を特定します。ここで
   - `<version>` は `6.6.5.123` など、Tectia Client/Server のリリース・バージョン及び Build 番号を表します。
   - `<platform>` はプラットフォーム・アーキテクチャを表し、 `x86_64` は 64 ビット Windows バージョン用です。

   一部の Windows バージョンでは、インストーラ・ファイルのファイル・タイプ (`.msi`) が表示されないことがあります。
2. インストーラ・ファイルをダブルクリックすると、Windows インストーラが起動します。

3. [**削除**] を選択して、アンインストールを開始します。

4. 削除が完了したら、[**完了**] をクリックします。

サイレント・コマンドライン・ツールを使用する

以下のコマンドを実行することで、Tectia Client をサイレントにアンインストールすることもできます。

```
msiexec /q /x ssh-tectia-client-<version>-windows-<platform>.msi
```

このコマンドの `<version>` は、削除する Tectia Client のバージョン (`6.6.5.123` など) であり、`<platform>` はプラットフォーム・アーキテクチャ (64 ビット Windows バージョンでは `x86_64`) を表します。

2.4. Tectia Client に関連するファイル

本項では、インストール・プロセスが保存する Tectia Client の実行ファイル、設定ファイル、ライセンス・ファイル、及びユーザ固有の設定ファイルのデフォルトの場所を一覧で説明します。

2.4.1. Unix の場合のファイルの場所

Unix プラットフォームでは、Tectia Client のファイルは以下のディレクトリにあります。

- `/etc/ssh2`
  - `/etc/ssh2/ssh-broker-config.xml`: 接続ブローカーのグローバル設定ファイル (ssh-broker-config(5) を参照)
  - `/etc/ssh2/ssh-broker-config-example.xml`: 接続ブローカーの設定例を含むサンプル・ファイル
  - `/etc/ssh2/licenses`: ライセンス・ファイルのディレクトリ(2.1.3 を参照)
  - `/etc/ssh2/hostkeys`: 既知のリモート・ホスト鍵のためのグローバル・ディレクトリ
- `/opt/tectia/share/auxdata/ssh-broker-ng`: 接続ブローカー設定ファイルの DTD ディレクトリ
  - `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd`: 接続ブローカーの設定ファイルで使用される文書型定義 (DTD)。 このファイルは**編集しないでください**!
  - `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`: この設定ファイルが最初に読み込まれ、工場出荷時の設定値が保持されます。このファイルを**編集しないでください** 。ただし、デフォルト設定を確認するために使用することはできます。接続ブローカーが起動するためには、このファイルは使用可能な状態であり、正しくフォーマットが維持されている必要があります。設定オプションについては、ssh-broker-config(5) を参照してください。

注意

Unix 上の Tectia Client 6.1 以前では、デフォルトの補助データ・ディレクトリ `auxdata` は `/etc/ssh2/ssh-tectia/` にありました。`ssh-broker-config.xml` ファイルが Tectia Client のバージョン 6.1 以前で作成されている場合は、接続ブローカー設定ファイルの DTD ディレクトリへの現在のパスが含まれるように、DOCTYPE 宣言を `/opt/tectia/share/auxdata/ssh-broker-ng/` に更新してください。

- `/opt/tectia/bin`: **sshg3** や **ssh-broker-g3** などのユーザ・バイナリ
- `/opt/tectia/man`: マニュアル・ページ
- `/opt/tectia/libexec`: ライブラリ・バイナリ
- `/opt/tectia/lib/sshsecsh`: ライブラリ・バイナリ

ユーザ固有の設定は以下のディレクトリに保存されます。

- `$HOME/.ssh2/ssh-broker-config.xml`: ユーザ固有の接続ブローカー設定ファイル
- `$HOME/.ssh2`: ユーザ鍵のデフォルト・ディレクトリ
  - `$HOME/.ssh2/random_seed`: 乱数発生器のシード・ファイル
  - `$HOME/.ssh2/hostkeys`: 既知のリモート・ホスト鍵のためのユーザ固有のディレクトリ
  - `$HOME/.ssh2/identification`: (オプション) 公開鍵認証で使用される identification ファイル

2.4.2. Windows の場合のファイルの場所

Windows では、Tectia 製品のデフォルトのインストール先 (以下 `<INSTALLDIR>`) は以下の通りです。

- "`C:\Program Files (x86)\SSH Communications Security\SSH Tectia`" (64 ビット Windows バージョンの場合)

Windows では、Tectia Client のファイルは以下のディレクトリにあります。

- "`<INSTALLDIR>\SSH Tectia Client`": Tectia Client のバイナリ
- "`<INSTALLDIR>\SSH Tectia Broker`": 接続ブローカーのバイナリ及び設定ファイルのサンプル
  - "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml`": 接続ブローカーのグローバル設定ファイル (man ページを参照: ssh-broker-config(5))
  - "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example.xml`": 接続ブローカーの設定例を含むサンプル・ファイル
-

"`<INSTALLDIR>\SSH Tectia AUX`": 補助ファイル及び **ssh-keygen-g3.exe** などのバイナリ

- "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng`": 接続ブローカー設定ファイルの DTD ディレクトリ
  - "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml`": この設定ファイルが最初に読み込まれ、工場出荷時の設定値が保持されます。このファイルを**編集しないでください**。ただし、デフォルト設定を確認するために使用することはできます。接続ブローカーが起動するためには、このファイルは使用可能な状態であり、正しくフォーマットが維持されている必要があります。 設定オプションについては、ssh-broker-config(5) を参照してください。
  - "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-ng-config-1.dtd`": 接続ブローカー設定ファイルで使用される文書型定義 (DTD)。 このファイルは**編集しないでください**!
- "`<INSTALLDIR>\SSH Tectia AUX\licenses`": ライセンス・ファイル・ディレクトリ(2.1.3 を参照)
- "`<INSTALLDIR>\SSH Tectia AUX\documents`": 使用許諾契約
- "`C:\ProgramData\SSH\hostkeys`": 既知のホスト鍵のグローバル・ディレクトリ

図 2.5 は、同じマシンに複数の Tectia 製品がインストールされている場合に Windows のスタート・メニューに表示される、Tectia のディレクトリ構造です。

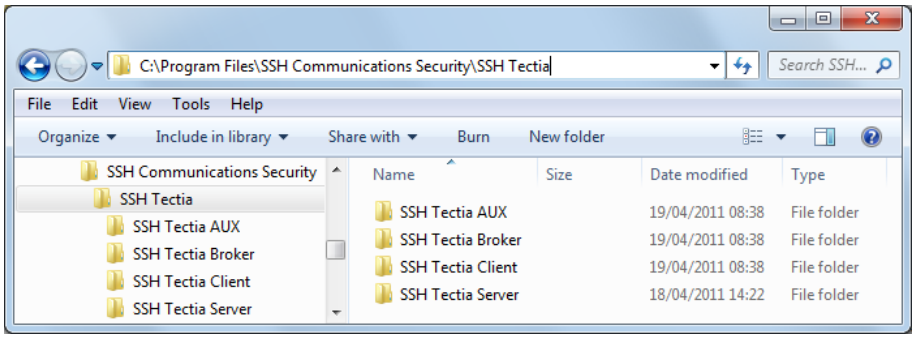


図2.5 Windows の Tectia のディレクトリ構造

ユーザ固有の設定は以下のディレクトリに保存されます。

- `%APPDATA%\SSH\ssh-broker-config.xml`: ユーザ固有の接続ブローカー設定ファイル
- `%APPDATA%\SSH\global.dat`: Tectia SSHターミナル GUI 設定ファイル
- `%APPDATA%\SSH\*.ssh2`: Tectia SSHターミナル GUI プロファイル設定ファイル
- `%APPDATA%\SSH\random_seed`: 乱数発生器のシード・ファイル
- `%APPDATA%\SSH\HostKeys`: 既知のリモート・ホスト鍵のためのユーザ固有のディレクトリ

- `%APPDATA%\SSH\UserKeys`: ユーザ公開鍵ペアのデフォルト・ディレクトリ
- `%APPDATA%\SSH\UserCertificates`: ユーザ証明書の鍵ペアのデフォルト・ディレクトリ
- `%APPDATA%\SSH\identification`: (オプション) 公開鍵認証で使用される identification ファイル

注意

ユーザ固有の `%APPDATA%` ディレクトリはデフォルトで非表示になっています。非表示のディレクトリを表示するには、Windows エクスプローラの設定を変更します。たとえば、メニューで [**整理**] → [**フォルダーと検索のオプション**] を選択します。[**表示**] タブの [**ファイルとフォルダーの表示**] で、[**隠しファイル、隠しフォルダー、および隠しドライブを表示する**] を選択します。

2.4.3. Windows でのレジストリ・キー

Windows では、Tectia Client をインストールすると以下のレジストリ・キーが作成されます。

- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker`
- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker GUI`
- `HKLM\SOFTWARE\SSH Communications Security\SSH Tectia`
- `HKLM\SOFTWARE\SSH Communications Security\SSH Tectia Client`
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia` (x64 アーキテクチャのみ)
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia Client` (x64 アーキテクチャのみ)

2.5. ssh/scp/sftp と sshg3/scpg3/sftpg3 間のシンボリックリンク (Unix の場合)

Tectia Client ではデフォルトで、コマンドライン・クライアント **sshg3**、**scpg3** 及び **sftpg3** と、それらの以前のバージョンである **ssh**、**scp** 及び **sftp** 間のシンボリックリンクは作成されません。

ssh/scp/sftp クライアントの代わりに **sshg3/scpg3/sftpg3** クライアントが常に使用されるように (ユーザが `ssh/scp/sftp` と入力した場合であっても) する場合は、インストール後いつでも以下のスクリプトを実行することで、両者の間にシンボリックリンクを作成できます。

```
# /opt/tectia/libexec/ssh-create-4.x-compat-symlinks
```

これら 2 つのバージョンのクライアントは異なるディレクトリに配置されているため、シンボリックリンクが必要です。

sshg3/scpg3/sftpg3

`/opt/tectia/bin/sshg3` にあります

ssh/scp/sftp

`/usr/local/bin/ssh` にあります

第3章 Tectia Client の使用開始

本章では、Tectia Client のソフトウェアが正常にインストールされた後に、同ソフトウェアの使用を開始する方法について説明します。

3.1. 製品コンポーネント

Tectia Client は以下のコンポーネントで構成されています。

- 接続ブローカー: **ssh-broker-g3**、 **ssh-broker-ctl**
- Secure Shell コマンドライン・ツール: **sshg3**、 **scpg3**、 **sftpg3**
- 補助的なコマンドライン・ツール: **ssh-keygen-g3**、 **ssh-cmpclient-g3**、 **ssh-scepclient-g3**、 **ssh-certview-g3**、 **ssh-ekview-g3**
- Tectia SSHターミナル GUI (Windows) (3.2.1 を参照)
- Tectia セキュア・ファイル転送 GUI (Windows) (5.2 を参照)
- Linux 及び Windows プラットフォームの Tectia 接続ステータス GUI (A.6.1 を参照)
- Linux 及び Windows プラットフォームの Tectia コネクション設定　GUI (A.1 を参照)

3.2. リモート・ホストへの最初のログイン

本項では、Tectia Client からデフォルト設定で Secure Shell サーバにログインする基本的な手順について説明します。Tectia Client 及び Tectia Server のデフォルト設定では、パスワード、公開鍵、及び GSSAPI でログインできます。

Windows で Tectia SSHターミナル GUI を使用してリモート・サーバ・ホストに接続する方法 (3.2.1 を参照) と、Windows 及び Unix で `sshg3` をコマンドラインで使用する方法 (3.2.2 を参照) について、別々に説明します。

Tectia　Client　には、接続設定を行うときや、接続ステータスや認証鍵を確認するときに役立つショートカット・メニューがあります。Tectia　のショートカット・メニューについては、A.6 を参照してください。

3.2.1. Tectia SSHターミナル GUI でのログイン (Windows の場合)

Tectia Client では、新しいリモート・ホスト・コンピュータとの接続を確立し、各ホストに必要な設定を管理することが簡単にできます。クイック接続オプションを使うと、新しい接続を素早く開くことができるので、各接続の設定に関連する作業を最小限に抑えることができます。また、新しいホストについてプロファイルを定義し、それぞれに適した設定を保存することも簡単にできます。

Windows では、Tectia SSHターミナル GUI を次のように使用して、リモート・ホストに接続することができます。

1. デスクトップ上の Tectia SSH ターミナルのアイコンをクリックして開きます。

図3.1 Tectia SSH ターミナルのアイコン

2. Tectia SSHターミナル GUI では、いくつかの方法で Secure Shell 接続を開くことができます。

   - セッションがすでに開いている場合は、**[クイック接続]** コマンド (ツールバーまたは **[ファイル]** メニュー) をクリックします。新しいリモート・ホスト・コンピュータに接続し、別のホストへの古い接続も開いたままにしておくことができます。

   - 前のセッションをすでに閉じている場合は、まだ接続していない状態でターミナル・ウィンドウがアクティブになっているときに、キーボードの **Enter** キーまたは **Space** キーを押すことで、新しいセッションを開くこともできます。

     - **Enter** キーを押すと、前のセッションと同じリモート・サーバに直接接続します。

     - **Space** キーを押すと **[サーバへ接続]** ダイアログボックスが開くので、接続するサーバを定義できます。

   - 接続プロファイルが定義済みの場合は、メニューで **[ファイル]** → **[プロファイル]** → **<プロファイル名>** をクリックするか、ツールバーの **[プロファイル]** ボタンをクリックしてからプロファイル名をクリックすることによって接続することもできます。

     この場合は、プロファイルで定義されている設定 (ホスト名、ポート、ユーザ名など) が自動的に接続に使用され、**[サーバへの接続]** ダイアログボックスは表示されません。接続プロファイルの作成と編集の手順については、A.1.3 を参照してください。

3. これにより **[サーバへ接続]** ダイアログボックスが開き、そこで接続するホストを定義できます。

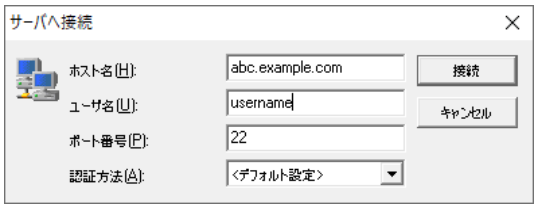


図3.2 [サーバへ接続] ダイアログボックス

以下の情報を定義して、**[接続]** をクリックします。

- **[ホスト名]** – FQDN、短いホスト名、またはリモート・ホストの IP アドレス。
- **[ユーザ名]** – リモート・ホストでのユーザ名
- **[ポート番号]** – 22 がデフォルトの Secure Shell リスナー・ポートです。
- **[認証方法]** – 利用可能なユーザ認証方法 (パスワード、公開鍵、キーボード・インタラクティブ、及び GSSAPI) のいずれかを指定しない限り、デフォルトの設定が使用されます。認証方法の詳細については、第4章 を参照してください。

同じ非接続時ターミナル・ウィンドウ内からのこれ以降のセッションでは、前の接続で使用された値があらかじめ入力されています。

4. サーバの認証が始まります。リモート・サーバ・ホストがローカル・コンピュータにホスト公開鍵を提供します。ホスト鍵によってサーバ・ホストが識別されます。

   Tectia Client により、この鍵についての情報がすでにユーザのホスト鍵ディレクトリに保存されているかどうか確認されます。保存されていない場合は、次に、コンピュータ上のすべてのユーザに共有のホスト鍵ディレクトリが確認されます。このホスト鍵についての情報が見つからない場合は、新しい鍵を検証するよう要求されます。

   サーバの認証に公開鍵認証が使用される場合、最初の接続が非常に重要です。Tectia Client が新しいサーバ・ホスト鍵を受信すると、ホスト識別メッセージが表示されます。

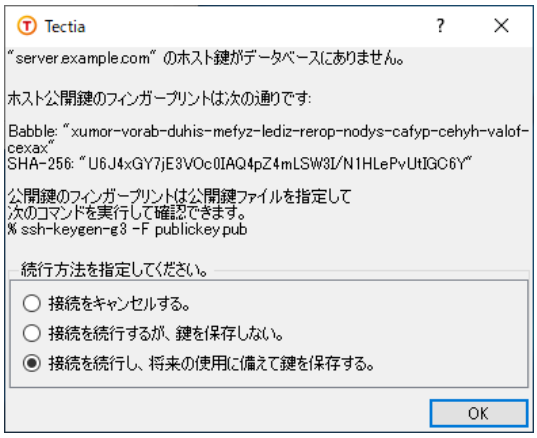


図3.3 ホスト識別ダイアログ – リモート・ホストへの最初の接続

メッセージは、ホストの公開鍵のフィンガープリントが SSH Babble 形式で表示されます。この形式は、ダッシュで区切られた英小文字 5 文字からなる読み上げ可能な一連の単語で構成されています。

5. フィンガープリントの有効性を確認してください。可能であればリモート・ホスト・コンピュータの管理者に電話で問い合わせてください。フィンガープリントを検証したら、将来使用するためにホスト鍵に関する情報を保存しておくと安全です。また、接続をキャンセルすることも、ホスト公開鍵の情報を保存せずにこの接続を続行することもできます。

警告

ホスト公開鍵の信頼性を検証せずに保存することは絶対に避けてください。

6. **[OK]** をクリックしてホスト識別ダイアログを閉じます。

サーバ公開鍵の情報は、クライアントが後でこの鍵を検証できるように、クライアント側のマシンに保存されます。Tectia Client では、 公開鍵情報は「`%APPDATA%\SSH\HostKeys`」ディレクトリに保存されます。

- "`C:\Users\<username>\AppData\Roaming`"

最初の接続の後は、ローカルに保存されたサーバ公開鍵に関する情報のみがサーバ認証に使用されます。

サーバ認証の詳細については、4.2 を参照してください。

7. ユーザの認証が始まります。 **[サーバへの接続]** ダイアログで選択した認証方法、またはデフォルトではパスワードまたは秘密鍵のパスフレーズを使って、サーバへの認証を受けるよう要求されます。必要な認証方法はサーバ設定によって異なります。

サーバによって認証されると、サーバへの Secure Shell 接続が確立されます。

8. (オプション) 接続時に接続プロファイルを使用しなかった場合、[**プロファイル追加**] ダイアログボックスが 5 秒間表示され、接続時に提示した情報を使用して新しい接続プロファイルを作成できるようになります。新しいプロファイルの名前を [**Profile Name**] フィールドに入力します。その後、[**プロファイルへ追加**] をクリックします。Tectia コネクション設定　GUI が開き、新しいプロファイルの情報が表示されます。[**適用**] をクリックして、新しい接続プロファイルを保存します。接続プロファイルの編集の詳細な手順については、A.1.3 を参照してください。

3.2.2. コマンドライン sshg3 でのログイン

以下のようにコマンドラインで **sshg3** を使用すると、リモート・ホストに接続できます。

1. 以下の構文で、 **sshg3** コマンドを入力します。

```
$ sshg3 <hostname>
```

例:

```
$ sshg3 abc.example.com
```

基本構文は以下の通りです。

```
$ sshg3 user@host#port
```

ここで、

- `user` - リモート・ホストで有効なユーザ名を入力します。`user@` 属性はオプションです。ユーザ名を指定しない場合は、ローカル・ユーザ名が使用されます。
- `host` - リモート・ホストの名前を IP アドレス、FQDN (完全修飾ドメイン名)、または短いホスト名で入力します。リモート・ホストでは、Secure Shell バージョン 2 サーバが動作している必要があります。
- `port` - リモート・サーバの Secure Shell リスナ・ポートの番号を入力します。`#port` 属性はオプションです。ポートを指定しない場合、デフォルトの Secure Shell ポート 22 が使用されます。

`ssh-broker-config.xml` ファイルで接続プロファイルを定義している場合は、以下のように接続プロファイルの名前を使って接続することもできます。

```
$ sshg3 profile1
```

この場合、プロファイルで定義されている設定 (ホスト名、ポート、ユーザ名など) が接続に使用されます。接続プロファイルの作成と編集の手順については、「`profiles` エレメント」 を参照してください。

sshg3 のコマンド及びオプションの詳細については、sshg3(1) を参照してください。

2. サーバ認証が始まります。サーバは検証のために、その公開鍵をクライアントに送信します (サーバ公開鍵認証が使用されている場合)。

Tectia Client は、この鍵がすでにユーザのホスト鍵ディレクトリに保存されているかどうか確認します。保存されていない場合は、次に、コンピュータ上のすべてのユーザに共有のホスト鍵ディレクトリが確認されます。

ホスト鍵が見つからない場合は、その鍵を検証するよう要求されます。

Tectia Client が 新しいホスト公開鍵を受信すると、ホスト識別メッセージが表示されます。例:

```
$ sshg3 user@host
Host key not found from database.
Key fingerprint:
xecic-fifub-kivyh-kohag-zedyn-logum-pycuz-besug-galoh-gupah-xaxby
You can get a public key's fingerprint by running
% ssh-keygen-g3 -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)?
```

メッセージは、ホストの公開鍵のフィンガープリントを SSH Babble 形式で表示します。この形式は、ダッシュで区切られた小文字 5 文字からなる読み上げ可能な一連の単語で構成されています。

3. フィンガープリントの有効性を確認してください。可能であればリモート・ホスト・コンピュータの管理者に電話で問い合わせてください。

   フィンガープリントが検証されて正しいことがわかったら、安全に鍵を保存して接続を続行できます。また、接続をキャンセルすることも、鍵を保存せずに接続を続行することもできます。

   サーバ公開鍵を保存する場合、鍵に関する関連情報はクライアント・ホストの `$HOME/.ssh2/hostkeys` ディレクトリ (Unix の場合) または `%APPDATA%\SSH\HostKeys` ディレクトリ (Windows の場合) に保存されます。最初の接続後は、ローカルに保存されているサーバ公開鍵の情報がサーバ認証に使用されます。

   サーバ認証の詳細については、4.2 を参照してください。

4. ユーザの認証が始まります。サーバへの認証を行うよう要求されます。これにはパスワード、または秘密鍵のパスフレーズ (公開鍵がすでにサーバにアップロードされている場合) を使用します。必要な認証方法はサーバ設定によって異なります。

   サーバによって認証されると、サーバへの Secure Shell 接続が確立されます。

3.3. 公開鍵認証の使用

公開鍵認証はデジタル署名の使用に基づいています。公開鍵認証を使用するには、まずクライアント上で鍵ペアを作成し、公開鍵をサーバにアップロードする必要があります。手順については、4.5 を参照してください。

接続確立の段階で、サーバは Tectia Client にチャレンジを送信します。このチャレンジには秘密鍵で署名します。サーバによるユーザ認証が正常に完了すると、サーバへの Secure Shell 接続が確立されます。

接続ブローカーは自動的に認証エージェントとして動作します。デジタル証明書やスマート・カードも簡単に利用できます。認証転送機能を使えば、複数の Secure Shell 接続で公開鍵認証の転送を行えます。接続ブローカーは、Tectia Client を起動すると自動的に起動します。

3.4. Tectia Client の設定

Tectia Client には、すぐに使い始められるようにデフォルト設定が含まれています。既存の設定を編集することで、使用環境のニーズに応じて Tectia Client の動作を調整できます。

Tectia Client では SSH 接続の暗号処理や認証関連のタスクはすべて、接続ブローカーと呼ばれるコンポーネントが処理するため、関連する設定もすべて接続ブローカーで行います。

Linux と Windows では、接続ブローカーの設定を扱うためのグラフィカル・ユーザ・インターフェイスが Tectia Client にあります。その他のプラットフォームでは、設定ファイル (XML 形式) で直接設定を編集できます。接続ブローカーの設定は、ステータス・バーや Tectia SSHターミナル GUI にある アイコンをクリックすると表示される、Tectia コネクション設定 GUI で編集できます。

Windows では、[**ユーザインターフェースの設定**] ツール (Tectia SSHターミナル GUI のツールバーにある アイコンをクリックすると表示される) を使って、Tectia SSHターミナル GUI や Tectia セキュア・ファイル転送 GUI のレイアウト、色、動作を好みに応じて任意に変更できます。ユーザ・インターフェイスのレイアウトを設定する手順については、付録B を参照してください。

3.4.1. 接続ブローカーの設定

Tectia Client のユーザにとって、接続ブローカーの設定に最も重要で、通常は最も必要になる項目は、接続プロファイルの設定です。その他の設定は通常、システム管理者が行います。

繰り返し接続する必要があるサーバには、接続プロファイルを作成することをお勧めします。プロファイルにはサーバ ID、そのサーバでのユーザ名、及び使用する認証方法に関する情報が含まれています。

一般に接続ブローカーでは、以下のような点を設定できます。

セキュアな接続の詳細

これらの設定では、Tectia Client がリモート・サーバへのセキュアな接続を確立する方法を定義します。たとえば、どのようなタイプの接続を開くか、どのような認証方法を使用するか、プロキシを使用するか、トンネルを許可するかなどです。

ユーザおよびサーバの認証方法

ユーザ認証の設定では、ユーザ認証データをリモート・サーバに送信する際に Tectia Client が使用する方法を定義します。Tectia コネクション設定 GUI には、公開鍵の作成とサーバへのアップロードに役立つ公開鍵ウィザード (Linux 及び Windows の場合) が含まれています。

サーバ認証の設定では、リモート・サーバがどのように Tectia Client によって認証されるかを定義します。

接続のトンネル

トンネルは、すべてまたは一部の TCP アプリケーション及び FTP 接続を保護するために定義できます。また、あるリモート・サーバから別のリモート・サーバへの X11 セッションや SSH 接続の転送を許可することもできます。

ヒント

最初に行うのは、ユーザ認証の設定 (ユーザの公開鍵の作成とリモート・サーバへのアップロード) と、繰り返し接続する必要があるサーバの接続プロファイルの作成です。

認証設定を定義する手順については 第4章 を、設定ファイル内の認証関連オプションについては **authentication-methods** を参照してください。

GUI で接続プロファイルを作成する手順については A.1.3 を、設定ファイルに直接接続プロファイルを追加する方法については 「`profiles` エレメント」 を参照してください。

接続ブローカーの**設定オプション**の詳細については、付録A を参照してください。

3.4.2. 接続ブローカーの設定ファイル

接続ブローカーの設定は `ssh-broker-config.xml` という名前の XML ファイルに保存されます。この設定ファイルは、任意の XML エディタやテキスト・エディタで編集できますが、`ssh-broker-config.xml` は妥当な XML ファイルのままでなければなりません。接続ブローカーの設定オプションについて、詳しくは ssh-broker-config(5) を参照してください。

接続ブローカーの設定を変更する場合は通常、Unix では `$HOME/.ssh2`、Windows では `%APPDATA%\SSH\` に保存されている設定ファイルのユーザ固有のコピーを編集します。まず、ユーザ固有の設定ファイルを作成する必要があります。

Windows と Linux では、Tectia コネクション設定 GUI はユーザ固有の設定ファイルへの書き込みも自動的に行います。

関連する設定ファイルの一覧とその場所については、以下の項を参照してください。

- Unix の場合は2.4.1 を参照してください。

- Windows の場合は 2.4.2 を参照してください。

3.4.3. コマンドライン・ツール

Tectia Client にはコマンドライン・ツールの **sshg3**、 **scpg3**、及び **sftpg3** が含まれており、Tectia SSHターミナル GUI や Tectia セキュア・ファイル転送 GUI と同様にセキュアな接続を開き、安全にファイルを転送するために使用できます。

これらのツールはスクリプトの中でもリアルタイムでも使用でき、その動作は一連のオプションで詳細に設定できます。コマンドラインで指定されたオプションは、設定ファイルの指定を上書きします。

各コマンドライン・ツールのオプションは man ページ sshg3(1)、scpg3(1)、及び sftpg3(1) に記載されています。

3.5. 接続プロファイルの作成

Windows 及び Linux 上の Tectia Client では、接続する各 Secure Shell サーバに対して別々の接続設定を行えます。また、同じサーバに対して、たとえば異なるユーザ・アカウントで複数のプロファイルを作成することもできます。

Windows では、以下のビューで接続プロファイルを追加できます。

- **Tectia SSHターミナル GUI** を起動し、**[プロファイル]** ボタンをクリックします。下図に示すように、ドロップダウン・メニューから **[プロファイルの追加]** を選択します。

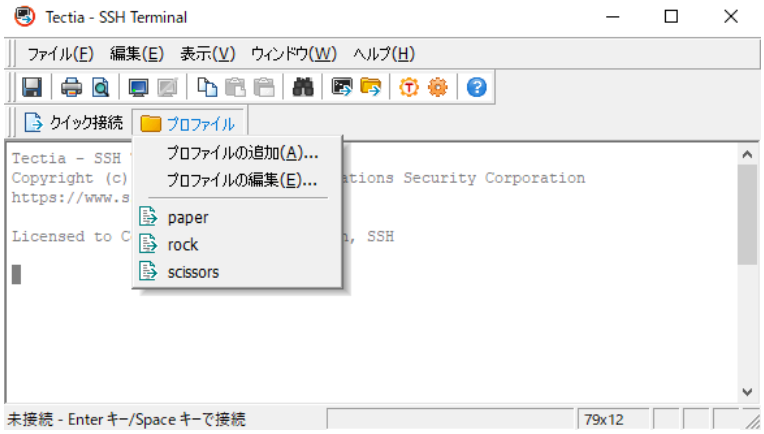


図3.4 接続プロファイルの追加

- Tectia SSHターミナル GUI を起動し、ツールバーの Tectia アイコン をクリックして **Tectia コネクション設定　GUI** を開きます。

  (Windows タスクバーの通知領域にある Tectia アイコン を右クリックし、ショートカット・メニューから **[設定]** を選択しても、Tectia コネクション設定　GUI を開くことができます。)

Linux では、**Tectia コネクション設定　GUI** を開きます。

1. 以下を入力して、`/opt/tectia/bin` ディレクトリに移動します。

```
$ cd /opt/tectia/bin/
```

2. 以下のコマンドで **Tectia コネクション設定　GUI** を起動します。

```
$ ssh-tectia-configuration
```

Tectia コネクション設定　GUI で **[接続プロファイル]** ページに移動し (下図を参照)、**[プロファイルを追加]** をクリックします。

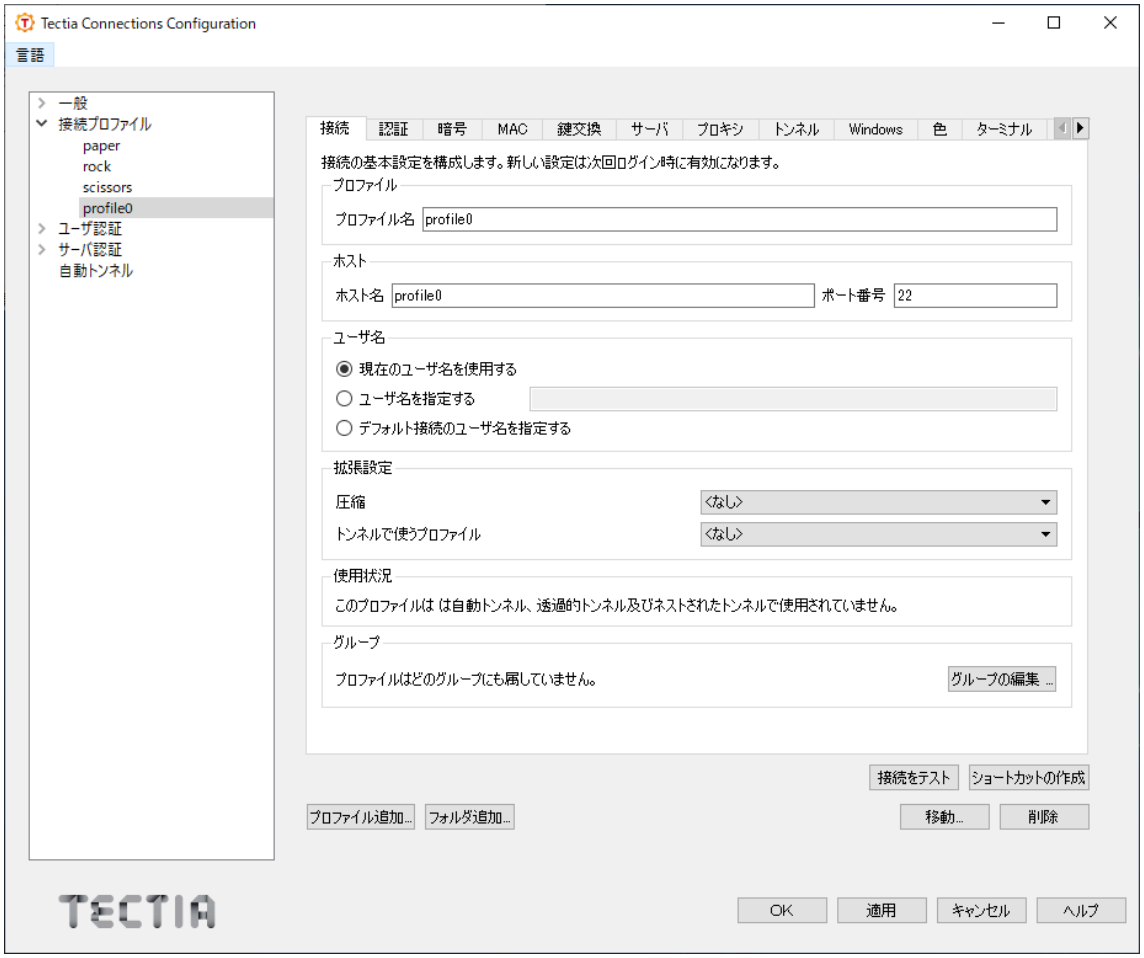


図3.5 接続プロファイルの追加

新しく作成された接続プロファイルは、**[一般]** → **[デフォルト接続]** ページで定義された認証、暗号、MAC、鍵交換、トンネリング、及び詳細なサーバ設定のデフォルト値を継承します。これらの値は、プロファイル固有のタブを使ったページでカスタマイズできます。図 3.6 を参照してください。

接続プロファイルの名前を変更するには、**[接続プロファイル]** リストでプロファイル名を右クリックし、**[名前の変更]** をクリックします。新しい名前を入力します。

接続プロファイルを削除するには、プロファイルを選択し、**[削除]** をクリックします。確認を求められます。**[はい]** をクリックして削除を実行します。

3.5.1. 接続プロファイル設定の定義

[接続プロファイル] ページの **[接続]** タブでは、接続で使用するプロトコル設定を定義できます。変更された接続設定は次回のログイン時に有効になります。

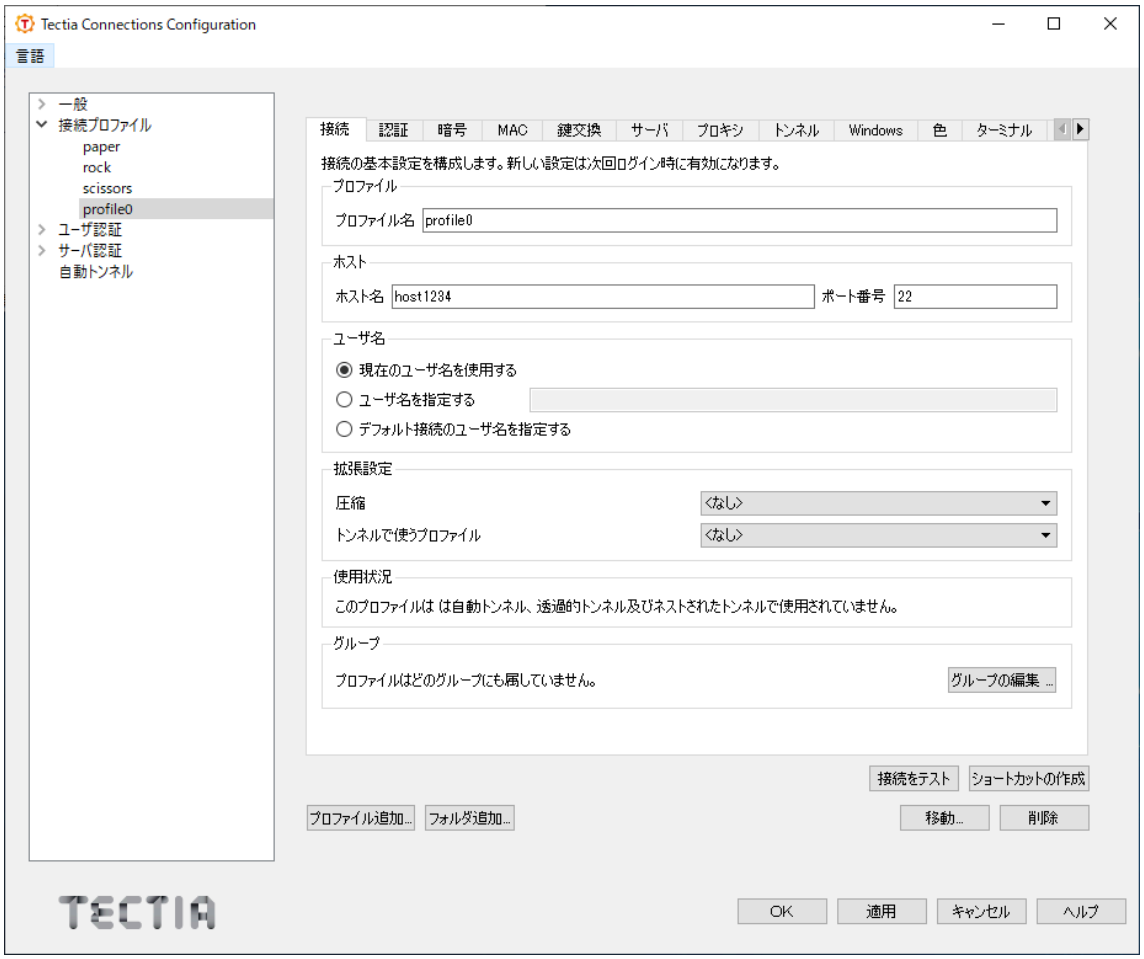


図3.6 接続プロファイルの設定

[プロファイル]

[プロファイル名] にプロファイルの名前を入力します。

[ホスト]

[ホスト名] には、このプロファイルで接続するリモート・ホスト・コンピュータの名前を入力します。

[ポート番号] には、Secure Shell 接続に使用するポート番号を入力します。デフォルトのポートは 22 です。

注意

Secure Shell サーバ・プログラムは、リモート・ホスト・コンピュータ上の指定されたポートをリッスンしている必要があり、そうでない場合は接続試行に失敗します。リモート・ホスト・コンピュータがリッスンしているポートが不明な場合は、リモート・ホストのシステム管理者に問い合わせてください。

[ユーザ名]

現在ログインしている Windows または Unix ユーザ名を使用して常に接続を確立する場合は、**[現在のユーザ名を使用する]** を選択します。この設定は、ユーザ名として %USERNAME% (パーセント記号に注意) を定義するのと同じような働きをします。

リモート・ホスト・コンピュータへの接続時に使用するユーザ名を定義する場合は、**[ユーザ名を指定する]** を選択してユーザ名を入力します。ユーザ名に %USERNAME% (パーセント記号に注意) を指定すると、接続時に現在の Windows または Unix ユーザ・アカウントの名前に置き換えられます。

[拡張設定]

現在は不要: **[圧縮]** では、使用する圧縮設定をドロップダウン・メニューから選択します。**[zlib]** または **[なし]** を選択できます。圧縮はデフォルトでは無効です。

現在は不要: **[トンネルで使うプロファイル]** では、使用する接続プロファイルをドロップダウン・メニューから選択します。ネストされたトンネルはすべて、プロファイルを使って作成されます。トンネリング機能については、『第6章』を参照してください。

3.6. FIPS 140-2 モードの有効化

Tectia Client は FIPS モードで動作させることができます。FIPS モードでは、すべての暗号化処理が FIPS 140-2 規格に従って実行されます。

FIPS モードでは、すべての暗号操作に OpenSSL 暗号化ライブラリが使用されます (3.6.3 を参照)。標準モードでは、すべての暗号操作に Tectia 独自の暗号化ライブラリが使用されます。

注意

FIPS モードでは、暗号化されていない秘密鍵を FIPS モジュールからエクスポートすることが FIPS 規定において禁じられているため、パスフレーズなしでユーザ鍵を生成することはできません。

3.6.1. 設定 GUI を使用した FIPS モードの有効化

Windows で FIPS モードを有効にするには、以下の手順に従ってください。

1. Tectia コネクション設定 GUI を開きます (A.1.1 を参照)。

2. ツリー・ビューで **[一般]** を選択して、一般設定に移動します。

3. **[暗号化ライブラリ]** で **[FIPS モード]** を選択します。

4. デフォルトの接続設定または任意の接続プロファイルに定義されている暗号化アルゴリズムが、FIPS モードと互換性があることを確認します。FIPS モードで許可されないアルゴリズムの場合は通知されます。FIPS 互換のアルゴリズムについては、付録F を参照してください。

5. **[適用]** をクリックします。

6. Tectia ショートカット・メニュー (A.6 を参照してください) から **[ブローカーを 停止]** をクリックします。 .

7. 新しいクライアントか接続を開始すると新しい 接続ブローカー が FIPS モードで起動します。

注意

Windows の場合は、 `SSH Tectia AUX` フォルダに `FIPSMODE` という名前のファイルを作成することで、全ての Tectia 製品 を FIPS モードに変更できます。 `FIPSMODE` ファイルが存在すると、全ての Tectia 製品は設定に関係なく 次回の再起動から FIPS モードで動作することに注意してください。

Windows の場合、 Tectia Server が Tectia Client と同じマシンにインストールされている場合、 Tectia Server の設定GUIから FIPS モードを変更し設定を適用するとこのファイルは自動的に作成及び削除されます。

3.6.2. 設定ファイルを使用した FIPS モードの有効化

Unix で FIPS モードを有効にする手順:

1. 設定を変更するために 接続ブローカー の設定ファイル `ssh-broker-config.xml` を開きます (「接続ブローカーのファイル」 を参照してください)。

2. `general` エレメント下の `crypto-lib` エレメントの値を `fips` に変更します。

3. 設定ファイルの `default-settings` エレメント及び `profiles` エレメントに定義されている暗号化アルゴリズムが FIPS モード互換であることを確認します。FIPS 互換のアルゴリズムについては、付録F を参照してください。

4. 設定ファイルを保存し、動作している場合は 接続ブローカー を停止します:

```
$ ssh-broker-ctl stop
```

5. 新しい接続を開始します。新しい 接続ブローカー が FIPS モードで動作していることは以下のコマンドで確認できます:

```
$ ssh-broker-ctl status
```

注意

Linux の場合は、 `/etc/ssh2/FIPSMODE` という名前のファイルを作成することで、全ての Tectia 製品 を FIPS モードに変更できます。 `FIPSMODE` ファイルが存在すると、全ての Tectia 製品は設定に関係なく 次回の再起動から FIPS モードで動作することに注意してください。

Linux および　Solaris では以下のコマンドを実行することで `FIPSMODE` を有効または無効にできます:

```
# /opt/tectia/sbin/ssh-modeset fips-mode on
```

```
# /opt/tectia/sbin/ssh-modeset fips-mode off
```

現在の FIPS モードは以下のコマンドで確認できます:

```
# /opt/tectia/sbin/ssh-modeset fips-mode-check
```

3.6.3. FIPS 認証済みの暗号化ライブラリ

連邦情報処理標準 (FIPS) 140-2 に準拠して認証されたバージョンの暗号化ライブラリを使用すると、Tectia Client、ConnectSecure および Server を FIPS モードで動作させることができます。

OpenSSL 暗号化ライブラリ一式は Tectia Client と一緒に配布されます。 この OpenSSL FIPS 認証の暗号化ライブラリは、以下の表に示す関数のクラスを提供するために使用されます。

Linux、Windows 及び Soralis で使用される OpenSSL 3.0.8 7 Feb 2023 (FIPS provider: 3.0.8) の関数のリストを 表 3.1 に示します。

表3.1 OpenSSL 暗号化ライブラリ・バージョン 3.0.8 で使用される API

API	説明	OpenSSL の関数
乱数	NIST SP800-90A に基づく AES/CTR DRBG を OpenSSL ライブラリから使用します。	RAND_bytes, RAND_add
暗号	aes-ecb、 aes-cbc、 aes-ofb、 aes-ctx、 aes-gcm 3des-(ecb、cbc、cfb、ofb)	EVP_CIPHER_CTX_*、 EVP_Cipher*
数学ライブラリ	OpenSSL が使用する Bignum 数学ライブラリ	BN_*
Diffie Hellman	DH、 ECDH、 curve25519、 curve448	EVP_PKEY_*、 DH_*
ハッシュ関数	Variants: sha1[検証のみ]、 sha224、 sha256、 sha384、 sha512	EVP_MD_*、 EVP_sha*, EVP_Digest*
公開鍵	Variants: RSA、 DSA、 ECDSA、 Ed25519	EVP_PKEY_*、 i2d_DSA_SIG、

API	説明	OpenSSL の関数
		d2i_DSA_SIG、 i2d_ECDSA_SIG、 d2i_ECDSA_SIG、 EVP_MD_*、 ECDSA_SIG_*、 DSA_SIG_*、 EC_GROUP_*、 EC_POINT_*
Misc		ERR_error_string_n、 ERR_get_error、 OpenSSL_version OSSL_PARAM_*、 OSSL_PROVIDER_*、 CRYPTO_free、 CONF_modules_load_file_ex、 EVP_default_properties_enable_fips

OpenSSL ライブラリの証明書関数は使用されていません。 Tectia には独自の証明書ライブラリがあります。

第4章 認証

Tectia client/server ソリューション が使用する Secure Shell プロトコルでは相互認証 – クライアントがサーバを認証し、サーバがクライアント・ユーザを認証する – が行われます。この認証方法では、互いに相手の身元を確かめます。

リモートの Secure Shell サーバ・ホストは、従来の公開鍵認証または証明書認証のいずれかを使用して自らを証明できます。

Secure Shell クライアント・ユーザの認証には、さまざまな方法があります。求める機能やセキュリティのレベルに応じて、異なる認証方法を組み合わせることも、別々に使用することもできます。

図4.1 ユーザ認証の方法

Tectia Client がデフォルトで使用するユーザ認証方法は、公開鍵認証、パスワード認証、キーボード・インタラクティブ認証、及び GSSAPI 認証です。公開鍵認証と証明書認証は、公開鍵認証方法にまとめられています。

複数の対話型認証方法が許可されるように定義されていて、前の方法が失敗した場合、Tectia Client は、それらの方法を順番にサーバに提示します。これによって、ユーザごとに異なる認証方法を定義できるようになり、同じサーバ構成でさまざまな認証方法に対応できます。

4.1. 対応しているユーザ認証方法

Tectia client/server ソリューション は以下のユーザ認証方法に対応しています。

表4.1 Tectia client/server ソリューション がサポートするユーザ認証方法

認証方法	Tectia Server		Tectia Client 及び ConnectSecure	
	Unix	Windows	Unix	Windows
パスワード[a]	x	x	x	x
公開鍵	x	x	x	x
証明書	x	x	x	x
ホストベース	x	x	x	
キーボード・インタラクティブ	x	x	x	x
PAM[b]	x		x	x
RSA SecurID[b]	x	x	x	x
RADIUS[b]	x	x	x	x
GSSAPI/ Kerberos	x	x	x	x

[a] SELinux が有効なサーバ側でシステムでは、パスワード認証は内部で PAM を使用します。

[b] キーボード・インタラクティブによる

4.1.1. OpenSSH 鍵との互換性

デフォルトでは、Tectia client/server ソリューション は IETF 標準の Secure Shell v2 フォーマットで保存された秘密鍵及び公開鍵を使用します。ただし Tectia Client 及び Server では、レガシーな OpenSSH フォーマットの鍵と関連ファイルも使用できます。

サポートされている OpenSSH フォーマットの鍵は以下の通りです。

- サーバ・ホスト鍵ペア
- クライアントがサーバの認証に使用する、信頼できるサーバ・ホスト公開鍵
- ユーザ秘密鍵 (クライアントがサーバの認証に使用する)
- 認証済みのユーザ公開鍵 (サーバがユーザの認証に使用する)、公開鍵オプションも含む

4.2. 公開鍵によるサーバ認証

サーバは、RSA、DSA、ECDSA、または Ed25519 の公開鍵アルゴリズムに基づくデジタル署名で認証されます。接続の最初に、サーバはその公開鍵をクライアントに送り、検証を受けます。

サーバ認証は Diffie-Hellman 鍵交換時に 1 回の公開鍵操作で行われます。サーバの認証に公開鍵認証が使用される場合、最初の接続が非常に重要です。最初の接続時に、クライアントは図 4.2 に示すようなメッセージを表示します。

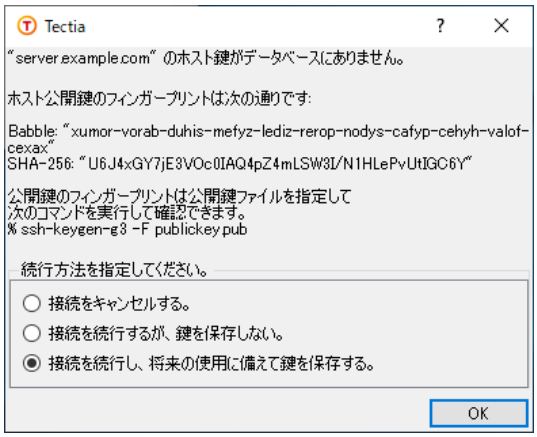


図4.2 Windows 上の Tectia Client – リモート・ホストとの最初の接続

警告

ホスト公開鍵の信頼性を検証せずに保存することは絶対に避けてください。

サーバ・ホストの身元を確認しやすいように、メッセージにはホストの公開鍵のフィンガープリントが表示されます。フィンガープリントは SSH Babble 形式で表示されます。この形式は、ダッシュで区切られた小文字 5 文字からなる読み上げ可能な一連の単語で構成されています。

フィンガープリントの有効性を検証するには、たとえば、リモート・ホスト・コンピュータの管理者に (できれば電話で) 連絡し、鍵のフィンガープリントが正しいか検証するよう管理者に依頼してください。フィンガープリントが検証されない場合、接続先のサーバが意図したサーバではない可能性があります (これを中間者攻撃といいます)。

フィンガープリントが検証できれば、接続は安全に続行できます。その後、サーバの公開鍵に関する関連情報がクライアント側のマシンに保存されます。Unix 上の Tectia Client では `$HOME/.ssh2/hostkeys` ディレクトリに保存されます。Windows 上の Tectia Client では `%APPDATA%\SSH\HostKeys` ディレクトリに保存されます。

保存されたホスト鍵の情報は、その後それらのリモート・ホストへの接続に使用されます。Tectia Client は、特定のサーバに対してどのタイプのホスト鍵 (DSA、RSA、ECDSA、または Ed25519) を保有しているかを確認し、それに応じてクライアントとサーバ間の接続に使用する鍵交換アルゴリズムを自動で選択します。これにより、1 種類のホスト鍵しか保存されていないホストへ素早く接続できます。

`auth-server-publickey` が `strict` (デフォルト設定) 以外のポリシーに 設定されている場合、接続ブローカーに対してログが有効になっていると、Tectia Client は、変更されたホスト公開鍵や新規のホスト公開鍵に関する情報をそのフィンガープリントとともに syslog (Unix の場合) またはイベント・ビューア (Windows の場合) に記録します。

4.2.1. ホスト鍵の保存フォーマット

リモート・ホストとの最初の接続時にホスト鍵を受け取り (またはホスト鍵が変更されており)、その鍵を保存することを選択すると、そのファイル名はハッシュ化された「keys_hhh...」というフォーマットで保存されます。この場合の hhh はホスト・ポート及び名前のハッシュです。保存されたファイルには、ホストの公開鍵のハッシュが含まれています。ハッシュの計算にはソルトが含まれています。ソルトの値は、ホスト鍵と同じディレクトリの salt ファイルに保存されます (Unix では $HOME/.ssh2/hostkeys、Windows では %APPDATA%\SSH\HostKeys)。ハッシュ化されたホスト鍵フォーマットは、ホスト上でのアドレス・ハーベスティングを困難にするためのセキュリティ機能です。

プレーン (従来の) フォーマットでは、ホスト鍵ファイルの名前にホストの名前とポートが含まれており (例: key_22_host.example.com.pub)、ホストの公開鍵がプレーンテキスト・フォーマットでファイルに格納されています。

保存フォーマットは、設定ファイル ssh-broker-config.xml の known-hosts エレメントの filename-format 属性で制御できます。属性値は plain または hash (デフォルト) でなければなりません。

```
<known-hosts path="$HOME/.ssh2/hostkeys" filename-format="plain" />
```

鍵を手動で追加する場合は、鍵の名前は key_<port>_<host>.pub のパターンにする必要があります。この場合の <port> は、Secure Shell サーバが動作しているポート、<host> はサーバへの接続時に使用するホスト名 (例: key_22_alpha.example.com.pub) です。

ハッシュ化されたフォーマットとプレーンテキスト・フォーマットの鍵が両方とも存在する場合、ハッシュ化されたフォーマットが優先されます。

クライアントが接続しようとするホスト名とポートによって、ホスト識別が異なることに注意してください。ホスト名には 4 つの異なるフォーマットがあります。

- 完全修飾ドメイン名 (FQDN)
- 短いホスト名
- IPv4 アドレス
- IPv6 アドレス

これらのフォーマットには互換性がないので、ホスト名のフォーマットに合わせてホスト鍵は別々に保存する必要があります。

ホスト鍵は、ログイン時に使用するホスト名フォーマットで保存されます。たとえば、alpha という名前のリモート・ホストに接続するときに、すべてのホスト名フォーマットを使用する場合、まず以下のコマンドでホストに接続し、4 つの名前すべてでホスト鍵を保存します。

- sshg3 user@alpha

  短いホスト名の鍵を生成します (プレーン・フォーマットでは key_22_alpha.pub)

- sshg3 user@alpha.example.com

FQDN の鍵を生成します (プレーン・フォーマットでは `key_22_alpha.example.com.pub`)

- `sshg3 user@10.1.101.10`

  IPv4 アドレスの鍵を生成します (プレーン・フォーマットでは `key_22_10.1.101.10.pub`)

- `sshg3 user@fd00:10:1:103::1:2f69`

  IPv6 アドレスの鍵を生成します (プレーン・フォーマットでは `key_22_fd00010000101030000000000012f69.pub`)

IPv6 アドレスを使用してサーバに接続すると、Tectia Client に与えられた IPv6 アドレスはコロンなしで正規化され、その正規化されたフォーマットが既知のホスト鍵ファイル名に使用されます。たとえば、 `::1#10022` のプレーン・フォーマットのホスト鍵ファイルは `key_10022_00000000000000000000000000000001.pub` になります。この正規化されたフォーマットは、ハッシュ化されたホスト鍵の保存と読み出しのプロセスでも使用されます。

また、同じホストで異なるポートに接続する必要がある場合、クライアントには、その目的のために別のホスト鍵 (例: `key_22_alpha.pub` と `key_222_alpha.example.com.pub`) が必要になります。

2回目の接続からは、ローカルに保存されているサーバ公開鍵の情報がサーバ認証に使用されます。

4.2.2. システムワイドなホスト鍵ストレージの使用

ホスト鍵がユーザ固有のホスト鍵ディレクトリに見つからない場合、次に Unix では `/etc/ssh2/hostkeys` ディレクトリ、 Windows では `C:\ProgramData\SSH\HostKeys` ディレクトリが検索されます。 ホスト鍵ファイルは自動的にシステムワイドなディレクトリに置かれるわけではなく、システム管理者 (`root`) が手動で更新する必要があります。

ここからは、ホスト鍵の配布を手動で行う手順について説明します。この手順では Unix のファイル・パスを使用していますが、Windows にも応用できます。Unix のパスを、対応する Windows のパスに置き換えてください。

ハッシュ化されたフォーマットでの鍵の保存

ハッシュ化されたリモート・ホスト鍵を取得し、システムワイドな場所に保存するには、以下の手順に従ってください。

1. クライアント側ユーザを選択します。このユーザの `$HOME/.ssh2/hostkeys` がシステムワイドな `/etc/ssh2/hostkeys` の素となります。システムワイドな場所に鍵を配置するために管理者権限が必要なので、このユーザには管理者権限が必要です。

   また、あるサーバのホスト鍵が変更された場合に、その後もシステムワイドな `/etc/ssh2/hostkeys` を維持するために同じユーザ・アカウントを使用する必要があります。このプロセスでは、ユーザのホスト鍵を `$HOME/.ssh2/hostkeys` ディレクトリで保管し、加えられた変更をシステムワイドな `/etc/ssh2/hostkeys` ディレクトリに複製します。

2. 初めて鍵を取得したときは `$HOME/.ssh2/hostkeys` ディレクトリが空であること、または保存されているホスト鍵が意図的なものであることを確認します。

   後で新しい鍵を取得する必要がある場合は、同じ `$HOME/.ssh2/hostkeys/salt` ファイルを使用する必要があります。

3. Tectia Client でリモート・サーバに接続し、フィンガープリントを検証し、鍵を保存します。

   この手順をリモート・サーバの台数分、繰り返します。完了しなければならないのは Secure Shell 接続の鍵交換部分のみで、ユーザ認証を完了する必要はありません。

4. システムワイドな場所に保管するホスト鍵をすべて取得したら、以下のコマンドを実行するなどして、鍵をシステムワイドな場所に配置します。

   ```
   # mkdir /etc/ssh2/hostkeys
   # cp -p $HOME/.ssh2/hostkeys/* /etc/ssh2/hostkeys
   ```

   Tectia Client がハッシュ化されたホスト鍵を識別できるように、ソルト・ファイル (`$HOME/.ssh2/hostkeys/salt`) もコピーする必要があります。また、複数のユーザがシステムワイドな `/etc/ssh2/hostkeys` ディレクトリの管理に関係する場合、該当するユーザ間で同じ `salt` ファイルを共有する必要があります。

ホスト鍵のためのシステムワイドな場所を作成したら、 **ssh-keygen-g3** ツールを使ってその場所を管理できます。

以下に挙げるコピーの例は、ホスト鍵ストレージの管理において最も頻繁に必要とされるコマンドです。これらのコマンドはユーザ固有のホスト鍵ストレージ (`$HOME/.ssh2/hostkeys` ファイル、及び場合によっては `$HOME/.ssh/known_hosts` ファイル) をソースとして使用します。異なるソースから鍵をコピーする場合は、適切な `--hostkeys-directory` または `--hostkey-file` オプションをコマンドに付加する必要があります。

たとえば、「alpha」という新しいホストの鍵を、ユーザ固有のホスト鍵ストレージからシステムワイドなディレクトリにコピーするには、以下のコマンドを入力します。

```
# ssh-keygen-g3 --append=no --overwrite=no \
--copy-host-id alpha /etc/ssh2/hostkeys
```

この場合、 `--overwrite=no` とコマンドしているので、サーバ「alpha」の鍵がすでに存在する場合、このコマンドは失敗し、鍵は更新されません。

既知のホストに鍵を追加するには、以下のコマンドを入力します。

```
# ssh-keygen-g3 --append=yes --copy-host-id alpha /etc/ssh2/hostkeys
```

既知のホストの鍵を更新するには、以下のコマンドを入力します。

```
# ssh-keygen-g3 --append=no --copy-host-id alpha /etc/ssh2/hostkeys
```

既知のホストの一覧からホストを削除するには、以下のコマンドを入力します。

```
# ssh-keygen-g3 --hostkeys-directory /etc/ssh2/hostkeys \
```

```
--delete-host-id alpha
```

ssh-keygen-g3 ツールの詳細については、ssh-keygen-g3(1) を参照してください。

プレーン・フォーマットでの鍵の保存

従来のリモート・ホスト鍵を取得し、システムワイドな場所に保存するには、以下の手順に従ってください。

1. サーバ側ユーザとして、サーバから `/etc/ssh2/hostkey.pub` ファイルを `key_<port>_<hostname>.pub` として、クライアントの `/etc/ssh2/hostkeys/` ディレクトリにコピーします。

   この操作は、サーバ上では非特権ユーザとして実行できますが、クライアント上で実行するには、`root` などの特権ユーザである必要があります。

2. セキュアな手段でファイルを転送するか、 **ssh-keygen-g3** の `-F` オプション (または `--fingerprint` オプション) で転送後にフィンガープリントが一致するかどうかを検証します。例えば、Unix 上の Tectia Server では以下のコマンドを入力します。

   ```
   $ ssh-keygen-g3 -F /etc/ssh2/hostkey.pub
   ```

   クライアントでは以下のコマンドを入力します。

   ```
   # ssh-keygen-g3 -F /etc/ssh2/hostkeys/key_<port>_<hostname>.pub
   ```

   クライアントが接続しようとするホストとポートによって、識別が異なることに注意してください。また、IP を使った接続は、同じホストへの接続でもポートが異なれば別のホストとみなされます。これらすべての異なる名前に、従来の `key_<port>_<hostname>.pub` をコピーできます。

4.2.3. ハッシュ化されたホスト鍵の解決

Tectia Client には、どのハッシュ化されたホスト鍵がどのサーバに属しているかを解決するためのツールがあります。クライアント側ホストには複数のサーバ・ホスト鍵が保存されることがあり、そのファイル名にサーバ名が示されていないことがあるため、あるサーバ公開鍵がクライアント・ホストに保存されているかどうか確認しなければならない場合があります。

Tectia コネクション設定 GUI では、[**ホスト鍵**] ページでこのツールを使用できます。「ホスト鍵の管理」 を参照してください。

コマンドラインでは、コマンドの構文は以下のようになります。

```
ssh-keygen-g3 -F host_name[#port]
```

例:

```
ssh-keygen-g3 -F examplehost#222
```

host_name は完全修飾ドメイン、短いホスト名、またはリモート・ホストの IP アドレスです。コマンドの port の定義は任意です。ポートを指定しない場合、デフォルトの Secure Shell ポート 22 が使用されます。

このツールは、要求されたホストの公開鍵の場所、フィンガープリント (SSH babble フォーマット)、及び種類 (RSA、DSA、ECDSA、または Ed25519) を表示します。例:

```
ssh-keygen-g3 -F examplehost
Fingerprint for key 'examplehost':
  (from location
   /home/user44/.ssh2/hostkeys/keys_bf53882dc47bb767edf161a4f636917f8358d635)
xuvin-zitil-ducid-gevil-vysok-buviz-nynun-pinat-tylev-gusez-dyxix (RSA)
```

指定されたサーバの鍵が見つからない場合、 **ssh-keygen-g3**-F コマンドは、鍵を探した場所を報告し、以下のように表示します。

```
/ No keys found from any key directories or known_hosts files.
```

ホスト鍵の確認対象となるファイルの場所は、複数定義できます。詳細については、4.2.4 を参照してください。

4.2.4. OpenSSH の known_hosts ファイルの使用

Tectia Client は、既知のサーバ・ホストの公開鍵データを含む OpenSSH 形式の known_hosts ファイルにも対応しており、デフォルトの場所にあるユーザ固有のファイル $HOME/.ssh/known_hosts、またはシステムワイドなファイル /etc/ssh/ssh_known_hosts からデフォルトでファイルを読み取ります。ハッシュ化されたフォーマットとプレーン・フォーマットのどちらのホスト鍵にも対応しています。

既知のホスト鍵に他のファイルが使用されるように定義する場合は、 known-hosts エレメントを使用して、接続ブローカーの設定ファイル ssh-broker-config.xmlでファイルを指定できます。既知のホスト鍵の確認には、複数のファイルの場所を定義でき、ssh-broker-config.xml ファイルで定義されている順に接続ブローカーがそれらを読み込みます。設定ファイルの設定はデフォルトの動作を上書きするので、すべての場所が読み込まれるようにするには、OpenSSH 形式の known_hosts ファイルのデフォルトの場所も定義する必要があります。例:

```
<general>
  ...
  <known-hosts path="/home/username/.ssh/known_hosts" />
  <known-hosts path="/etc/ssh/ssh_known_hosts" />
  <known-hosts path="/home/.ssh2/hostkeys" />
  <known-hosts path="/u/username/.ssh2/hostkeys" />
</general>
```

OpenSSH 形式の known_hosts ファイルの処理を無効にするには、空の設定known-hosts path=""を定義します。そのようにすることで、Tectia 関連のホスト鍵ディレクトリのみが使用されます。

OpenSSH 形式の known_hosts ファイルが Tectia Client によって自動的に更新されることはありません。新しいホスト鍵は常に、Tectia の $HOME/.ssh2/hostkeys ディレクトリ、または ssh-

broker-config.xml で最後に設定されたディレクトリに保存されます。詳細については、 known-hosts を参照してください。

4.3. 証明書によるサーバ認証

証明書によるサーバ認証は、公開鍵によるサーバ認証と同様に行われますが、特定のサーバへの最初の接続時に中間者攻撃の可能性が排除される点が異なります。サーバ証明書に付与された認証局の署名により、最初の接続でもサーバ証明書の真正性が保証されます。

ここからは、証明書によるサーバ認証プロセスの簡単な概要について説明します。

1. サーバがその証明書 (公開鍵を含む) をクライアントに送信します。このパケットには、サーバの秘密鍵で署名された、セッションに固有のランダムなデータも含まれています。

2. サーバ証明書は認証局 (CA) の秘密鍵で署名されているので、クライアントは CA 証明書を使用してサーバ証明書の有効性を検証できます。

3. クライアントは、証明書がサーバの名前または IP アドレスと一致するかどうか確認します。接続ブローカーの設定でエンド・ポイント同一性チェックが有効になっている場合 (ssh-broker-config.xml ファイルの cert-validation 属性 end-point-identity-check で、または Tectia コネクション設定 GUI の [CA 証明書] ページの [**エンドポイント同一性チェック**] オプションで) 、クライアントはサーバのホスト名または IP アドレスと、サーバ証明書で指定されたサブジェクト名またはサブジェクト代替名 (DNS アドレス) を比較します。

   接続ブローカーの設定でエンド・ポイント同一性チェックが無効になっている場合、サーバ・ホスト証明書のフィールドは検証されず、有効期間と CRL チェックのみに基づいて証明書が受け入れられます。

   警告

   クライアントでエンド・ポイント同一性チェックを無効にすることはセキュリティ・リスクです。そのようにすると、サーバ・ホスト証明書の発行元と同じ、信頼できる CA から発行された証明書を持っている人は誰でも、サーバに対して中間者攻撃を行えるようになります。

   エンド・ポイント同一性チェックは、サーバのホスト名が証明書のホスト名と一致しない場合に、接続を受け入れるかキャンセルするかを Tectia Client からユーザに確認するように設定することもできます。

4. クライアントは、初期パケットに含まれる署名を確認することで、サーバに有効な秘密鍵があるかどうかを検証します。

システムは認証の際、証明書が失効していないかどうか確認します。この確認は、オンライン証明書状態プロトコル (OCSP) または証明書失効リスト (CRL) を使用して行うことができ、LDAP または HTTP リポジトリで公開できます。

証明書に有効な **認証情報アクセス** 拡張機能が含まれている場合、または OCSP レスポンダが別途設定されている場合、OCSP が自動的に使用されます。OCSP レスポンダが定義されていない場合、または OCSP 接続に失敗した場合は、CRL が使用されます。LDAP を CRL の公開方法として使用する場合、LDAP リポジトリの場所も `ssh-broker-config.xml` ファイルで定義できます。

4.3.1. 設定ファイルによる CA 証明書の管理 (Unix)

クライアントを設定する際、CA 証明書を信頼し、証明書失効リスト (CRL) にアクセスするよう設定する必要があります。

サーバの証明書を信頼するようにクライアントを設定するには、以下の手順に従ってください。

1. CA 証明書をクライアント・マシンにコピーします。X.509 証明書をそのままコピーするか、または CA 証明書を含む PKCS #7 パッケージをコピーします。

   ssh-keygen-g3 で `-7` フラグを指定すると、PKCS #7 パッケージから証明書を抽出できます。

2. `ssh-broker-config.xml` ファイルの `general` エレメントで、ホスト認証に使用する CA 証明書を定義します。

   ```
   <cert-validation end-point-identity-check="yes"
                    http-proxy-url="http://proxy.example.com:800">
     <ldap-server address="ldap://ldap.example.com:389" />
     <ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
     <dod-pki enable="no" />
     <ca-certificate name="ssh_ca1"
                     file="ssh_ca1.crt"
                     disable-crls="no"
                     use-expired-crls="100" />
   </cert-validation>
   ```

   クライアントは、定義された CA によって発行された証明書のみを受け入れるようになります。

   `ca-certificate` エレメントの `disable-crls` 属性を `"yes"` に設定することで、CRL の使用を無効にできます。

   > **注意**
   >
   > CA が一時的な証明書を発行する場合かテスト目的以外では CRL の使用を無効にしないでください。それ以外の場合は、常に CRL を使用することを強くお勧めします。

   また、CRL チェックに使用する LDAP サーバまたは OCSP レスポンダを定義します。CA 証明書に CRL 配布ポイント拡張機能が含まれる場合は、LDAP サーバの定義は必要ありません。

3. CA サービス (OCSP、CRL) がファイアウォールの環境内にある場合、`ssh-broker-config.xml` ファイルに SOCKS サーバも定義します。SOCKS サーバは `cert-validation` の内部で、 `socks-server-url` エレメントで定義されます。

4.3.2. GUI による CA 証明書の管理

Tectia コネクション設定 GUI を使用した CA 証明書の管理については、「CA 証明書の管理」 で説明されています。

4.4. パスワードによるユーザ認証

パスワード認証方法はデフォルトで設定されているので、最も導入しやすい方法です。通信はすべて暗号化されるので、パスワードが盗聴されることはありません。

Unix システムでは、パスワード認証はパスワードの設定方法に応じて `/etc/passwd` または `/etc/shadow` ファイルを使用します。シャドウ・パスワード・ファイルは Linux サーバと Solaris サーバで使用できますが、HP-UX サーバと AIX サーバでは使用できません。

Windows のパスワード認証では、Windows へのログインに使うパスワードを使ってユーザを認証します。 さらに、SSHサーバが許可する場合は、管理者権限を持つユーザはユーザ名の前に `elevated` を付けることでパーミッションを保持できます。例えば:

```
$ sshg3 elevated,Administrator@example.com
```

4.4.1. 設定ファイルによるパスワード認証の定義 (Unix)

クライアントでパスワード認証を有効にするには、以下のように `ssh-broker-config.xml` ファイルの `authentication-methods` エレメントに `auth-password` エレメントを含める必要があります。

```
<authentication-methods>
...
<auth-password />
...
</authentication-methods>
```

その他の認証方法も同様に設定ファイルに記載できます。 その際、対話の必要が最も少ない方法を最初に配置します。

4.4.2. 接続プロファイル内の保存されたパスワードの使用

非対話型接続で使用される接続プロファイルでは、Tectia Client の設定やシステムに保存されたパスワードを使用することもできます。

接続ブローカーの設定ファイル `ssh-broker-config.xml` では、保存されるパスワードは以下のような構文で、 `password` エレメントで設定されます。

```
<profiles>
  <profile>
    <authentication-methods>
      <auth-password />
    </authentication-methods>
  ...
    <password file="path/to/file" />
  </profile>
...
</profiles>
```

`password` エレメントは、パスワード認証の応答としてクライアントが送信するユーザ・パスワードの指定に使用できます。

パスワードは `string` 属性で直接指定することもできますが、より安全な方法としては、パスワードを含むファイルへのパスを `file` 属性で定義するか、または `command` 属性を使用して、パスワードを出力するプログラムまたはスクリプトへのパスを定義します。

`command` 属性を使用してシェル・スクリプトを参照する場合は、そのスクリプトがユーザのシェルを定義し、実際のパスワードを出力することも確認してください。そのようになっていない場合、シェル・スクリプトに使用するシェルを特定できないため、実行されたプログラムは失敗します。たとえば、パスワードの文字列が `my_password.txt` という名前のファイルに定義されていて、bash シェルを使用する場合は、以下の行をスクリプトに含めます。

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

警告

このオプションを使用してパスワードを指定する場合は、意図するユーザ以外は `ssh-broker-config.xml` ファイル、パスワード・ファイル、またはプログラムにアクセスできないようにすることが非常に重要です。

注意

コマンドライン・オプションで指定したパスワードがある場合、この設定を上書きします。

Tectia コネクション設定 GUI で、保存されるパスワードを [**接続プロファイル**] → [**認証**] タブで設定します。[**非インタラクティブ使用のためにパスワードを保存**] を選択し、パスワード、またはパスワード・ファイルやプログラムへのパスを定義します。

警告

保存されたパスワードを使用する場合は、意図するユーザ以外は Tectia Client のホストやパスワード・ファイル、またはプログラムにアクセスできないようにすることが非常に重要です。

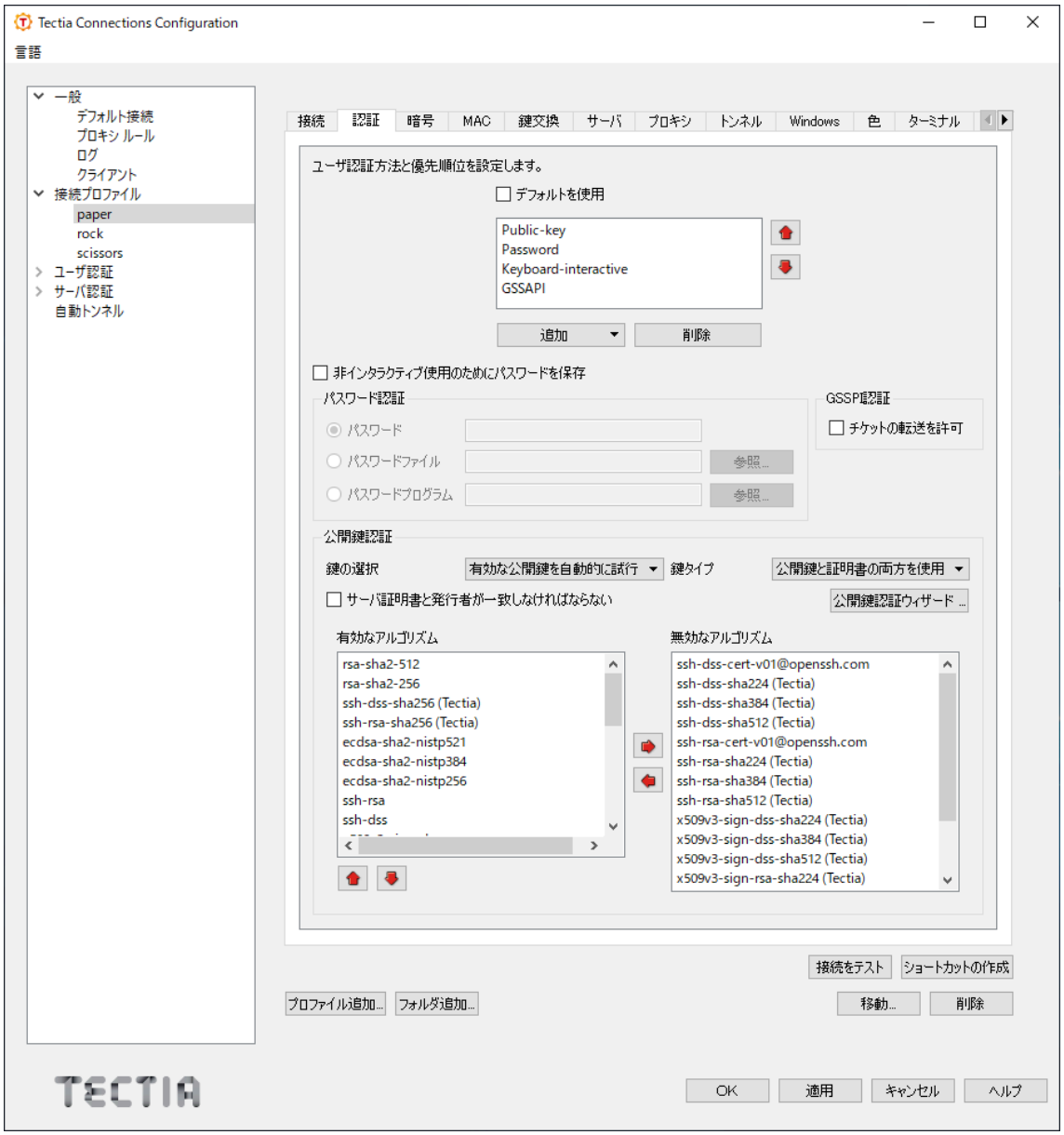


図4.3 プロファイルの認証方法の設定

パスワードをそのまま設定に保存するには、[**パスワード**] フィールドにパスワードを直接入力します。

パスワードが格納されているファイルを使用するには、[**パスワードファイル**] を選択し、そのファイルへのパスをフィールドに入力します。

パスワードを出力するプログラムまたはスクリプトを使用するには、[**パスワードプログラム**] を選択し、そのプログラムへのパスをフィールドに入力します。

注意

パスワード・ファイルやパスワード・プログラムに対して適切なパーミッションを設定することが必要です。ファイルまたはプログラム実行ファイルはユーザ、

ローカル管理者、またはローカル管理者グループのメンバーによって所有されており、ファイルに管理者用の `allow-type` パーミッションが設定されている必要があります。

4.4.3. GUI による認証方法の管理

Tectia コネクション設定　GUI を使用した認証方法の管理については、「認証の定義」 で説明されています。

4.5. 公開鍵によるユーザ認証

公開鍵認証はデジタル署名の使用に基づいています。各ユーザは鍵ファイルのペアを作成します。その鍵ファイルのうち、1 つはユーザの公開鍵、もう 1 つはユーザの秘密鍵です。サーバはユーザの公開鍵を保持しており、秘密鍵はユーザだけが所有しています。

鍵ファイルは、ユーザが　書き込み　権　(及び　読み取り　権) を持ち、かつ他のユーザがアクセスできない場所に保存する必要があります。これらのユーザ固有の権利は `key.pub` ファイル、`authorized_keys` ディレクトリ、及び `authorization` ファイル (使用する場合) について必要です。

ユーザが認証を試みると、クライアントがサーバに署名を送信し、サーバは一致する公開鍵の有無を確認します。鍵がパスフレーズで保護されている場合、クライアントはユーザにパスフレーズの入力を要求します。

秘密鍵ファイルは自分自身の認証に使用されることを忘れないでください。秘密鍵ファイルは安全な場所に保管し、他の誰からもアクセスできないようにしてください。他の誰かがその秘密鍵ファイルにアクセスできた場合は、あなたになりすましてリモート・ホスト・コンピュータへのログインを試行することが可能になります。できる限り、秘密鍵を保護するためのパスフレーズを定義してください。複数のユーザで共有しているマシンでは、他のユーザが秘密鍵にアクセスできないパーミッション設定になっていることを確認してください。

警告

他のユーザがアクセスできる場所には秘密鍵を保存しないでください。

Windows のローミング・プロファイル機能を使用している場合は、個人設定がローミング・プロファイル・サーバに複製されることにも注意してください。秘密鍵をデフォルトの場所 (Windows　ユーザ・アカウントのプロファイル・フォルダ下)　に保存している場合、ネットワーク・トラフィックを盗聴している悪意のあるユーザに秘密鍵が盗聴される可能性があります。そのため、ユーザ設定フォルダが、プロファイルのローミングで使用されるディレクトリであってはなりません。

Tectia Client で公開鍵認証を使用するには、以下の手順を実行してください。

1. 鍵ペアを生成します。Windows で、組み込みの [公開鍵認証ウィザード] を使用して (4.5.3 を参照)、あるいは Unix または Windows のコマンドラインで **ssh-keygen-g3** を使用して (4.5.1 を参照)、あなた自身の鍵ファイルを生成できます。

Tectia コネクション設定　GUI の [**鍵と証明書**] ページで既存の鍵をインポートすることもできます。「鍵と証明書の管理」 を参照してください。

2. 公開鍵をリモート・ホスト・コンピュータにアップロードします。Windows　では、これは自動的に行うことができます　(「公開鍵の自動アップロード」　を参照)。Unix　及び Windows では、公開鍵を手動でコピーすることもできます (4.5.2 を参照)。

以下の項の説明では、

- サーバ は、接続しようとしている Secure Shell サーバを実行しているリモート・ホストです。
- サーバ・ユーザは、ログインに使用する Server 上のユーザ名です。
- クライアント は、Secure Shell クライアント (Tectia Client) を実行しているホストです。
- クライアント・ユーザ は、ServerUser として Server へのログインを許可される Client 上のユーザ名です。

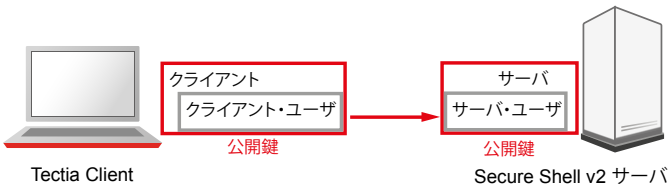


図4.4 公開鍵を使用したユーザ認証

この説明では、クライアント・ユーザ が他の認証方法 (通常はパスワード) を使用して、サーバ・ユーザ として サーバ にログインすることを許可されていることを前提としています。

4.5.1. ssh-keygen-g3 による鍵の作成

公開鍵ペアを作成するには、クライアント 上で以下のように **ssh-keygen-g3** を実行します。

```
$ ssh-keygen-g3
Generating 3072-bit rsa key pair
   9 oOo.oOo.oOo
Key generated.
3072-bit rsa, ClientUser@Client, Mon Aug 15 2022 12:08:07 +0200
Passphrase :
Again :
Private key saved to  /home/ClientUser/.ssh2/id_rsa_3072_a
Public key saved to   /home/ClientUser/.ssh2/id_rsa_3072_a.pub
```

オプションなしで実行すると、 **ssh-keygen-g3** は新しい鍵のためのパスフレーズを要求します。十分な長さ (20 文字程度) の任意の連続する文字 (スペースも可) を入力してください。

注意

FIPS モードでは、暗号化されていない秘密鍵を FIPS モジュールからエクスポートすることが FIPS 規定において禁じられているため、パスフレーズなしでユーザ鍵を生成することはできません。

新しい認証キー・ペアは 2 つの別々のファイルから構成されています。そのうちの 1 つが秘密鍵で、自分以外には決して知られてはならないものです。秘密鍵はパスフレーズと一緒にしか使用できません。

Unix では、鍵ペアはデフォルトで `$HOME/.ssh2` ディレクトリに保存されます (存在しない場合は **ssh-keygen-g3** によって作成されます)。Windows では、鍵ペアはデフォルトで `%APPDATA%\SSH\UserKeys` ディレクトリに保存されます。

上記の例では、秘密鍵ファイルは `id_rsa_3072_a` です。公開鍵ファイルは `id_rsa_3072_a.pub` であり、他のコンピュータに配布できます。

デフォルトでは、 **ssh-keygen-g3** は 3072 ビットの RSA 鍵ペアを作成します。 **ssh-keygen-g3** で `-t` オプションを指定することで、DSA、ECDSA、または Ed25519 の鍵を生成することもできます。鍵の長さは `-b` オプションで指定できます。自動化されたジョブのために、以下のように `-P` オプションを使って、パスフレーズなしで鍵を生成できます。

```
$ ssh-keygen-g3 -t ecdsa -b 384 -P
```

ssh-keygen-g3 のオプションの詳細については、ssh-keygen-g3(1) を参照してください。

4.5.2. 手動による公開鍵のアップロード

本項のコマンドはすべて、Tectia Client を実行しているマシンから **sshg3** 及び **scpg3** を使用して示されています。サーバ側の設定は、リモート・サーバにログインし、ローカルでコマンドを入力することでも行えます。

あなたの鍵ペアによる公開鍵認証を有効にするには、以下の手順に従ってください。

1. 接続ブローカーが探せるディレクトリに鍵を配置します。

   接続ブローカーはデフォルトでは、Unix では `$HOME/.ssh2` ディレクトリ、Windows では `%APPDATA%\SSH\UserKeys` 及び `%APPDATA%\SSH\UserCertificates` ディレクトリで見つかった各鍵を試行します。

   Tectia コネクション設定 GUI の [**鍵と証明書**] ページで、鍵がある他のディレクトリを追加することもできます。「鍵と証明書の管理」 を参照してください。Unix では `ssh-broker-config.xml` ファイルの `general/key-stores/key-store` エレメントを使用できます。「鍵ストアの設定例」t を参照してください。

2. (オプション) identification ファイルを作成します。

   すべての鍵をデフォルトのディレクトリに保存し、すべての鍵を公開鍵認証及び/または証明書認証に使用することを許可する場合、`identification` ファイルを使用する必要はありません。`identification` ファイルが存在しない場合、接続ブローカーはデフォルトのディレク

トリで見つかった各鍵を試行します。identification ファイルが存在する場合、そのファイルに記載されている鍵が最初に試行されます。

identification という名前のファイルを、Unix では $HOME/.ssh2 ディレクトリに、Windows では %APPDATA%\SSH ディレクトリに作成します。

このファイルを任意のテキスト・エディタで編集し、以下の行を追加します（id_rsa_3072_a は秘密鍵のファイル名に置き換えてください)。

```
IdKey       id_rsa_3072_a
```

鍵は identification ファイルと同じディレクトリにあることを前提としていますが、絶対パスや相対パスを指定することもできます。たとえば Windows では、以下のように入力します。

```
IdKey       UserKeys\id_rsa_3072_a
```

identification ファイルには複数のキー ID を含めることができます。identification ファイルの構文の詳細については、 $HOME/.ssh2/identification を参照してください。

3. 他の認証方法で サーバ に接続し、まだ存在しない場合はホーム・ディレクトリの下に .ssh2 (及び .ssh2/authorized_keys) または .ssh ディレクトリを作成します。

   リモート・ホストが実行しているサーバ・バージョンに応じて、以下のいずれかのコマンドを実行します。

   - Unix または z/OS 上の Tectia Server:

     ```
     $ sshg3 ServerUser@tectia_server mkdir .ssh2
     ```

     Tectia Server 5.x 以降の Unix で authorization ファイルを使用しない場合は、以下のように authorized_keys ディレクトリも作成します。

     ```
     $ sshg3 ServerUser@tectia_unix mkdir .ssh2/authorized_keys
     ```

   - Windows 上の Tectia Server:

     ```
     $ sshg3 ServerUser@tectia_win "cmd /c mkdir .ssh2"
     ```

     Tectia Server 5.x 以降の Windows で authorization ファイルを使用しない場合は、以下のように authorized_keys ディレクトリも作成します。

     ```
     $ sshg3 ServerUser@tectia_win "cmd /c mkdir .ssh2\authorized_keys"
     ```

   - Unix または z/OS 上の OpenSSH サーバ:

     ```
     $ sshg3 ServerUser@open_server mkdir .ssh
     ```

4. 公開鍵を サーバ にコピーします。

   リモート・ホストが実行しているサーバ・バージョンに応じて、以下のいずれかの操作を行います。

   - Unix 及び Windows 上の Tectia Server 5.x 以降:

SCP を使用して、公開鍵をサーバの authorized_keys ディレクトリ (Unix サーバではデフォルトで $HOME/.ssh2/authorized_keys、Windows サーバではデフォルトで %USERPROFILE%\.ssh2\authorized_keys) にアップロードします。

```
$ scpg3 id_rsa_3072_a.pub ServerUser@tectia_server:.ssh2/authorized_keys/
```

- Unix 及び Windows 上の Tectia Server 4.x:

  SCP を使用して、サーバ (Unix ではデフォルトで $HOME/.ssh2 ディレクトリ、Windows サーバではデフォルトで %USERPROFILE%\.ssh2 ディレクトリ) に公開鍵をアップロードします。

```
$ scpg3 id_rsa_3072_a.pub ServerUser@tectia4x_server:.ssh2/
```

- Tectia Server for IBM z/OS:

  公開鍵を EBCDIC フォーマットに変換する必要があります。そのためには、ファイル転送コマンドに **scpg3** の dst-site コマンドライン・オプション、または **sftpg3 site** コマンドを含めます。 **site** のパラメータの詳細については、5.3.1 を参照してください。

  SCP を使用して、公開鍵をサーバ (デフォルトでは $HOME/.ssh2 ディレクトリ) にアップロードします。

```
$ scpg3 --dst-site="C=ISO8859-1,D=IBM-1047,X=TEXT" id_rsa_3072_a.pub \
ServerUser@tectia_zos:$HOME/.ssh2/
```

- Unix 上の OpenSSH サーバ:

  SCP を使用して、公開鍵をサーバの $HOME/.ssh ディレクトリにアップロードします。

```
$ scpg3 id_rsa_3072_a.pub ServerUser@open_unix:.ssh/
```

5. authorization または authorized_keys ファイルを サーバ 上で作成します。

   リモート・ホストが実行しているサーバ・バージョンに応じて、以下のいずれかの操作を行います。

   - Unix 及び Windows 上の Tectia Server 5.x 以降では、公開鍵がユーザの authorized_keys ディレクトリに保存されている場合、authorization ファイルは必要ありません。ただし、authorization ファイルをオプションで使用できます。次の項目で説明している、Tectia Server 4.x でのファイルの作成方法を参照してください。

   - Unix 及び Windows 上の Tectia Server 4.x 及び Tectia Server for IBM z/OS では、ユーザの .ssh2 ディレクトリに保存されている authorization ファイルが必要です。ログインに認可された公開鍵は、authorization ファイルによって指定されます。

     authorization ファイルに鍵のエントリを追加します。Unix または z/OS サーバでは以下のように入力します。

```
$ sshg3 ServerUser@tectia_server "echo Key id_rsa_3072_a.pub >> \
.ssh2/authorization"
```

Windows サーバでは以下のように入力します。

```
$ sshg3 ServerUser@tectia4x_win "cmd /c echo Key id_rsa_3072_a.pub >> \
.ssh2\authorization"
```

認証ファイルの例を以下に示します (Unix 及び z/OS サーバではデフォルトで `$HOME/.ssh2/authorization`、Windows サーバではデフォルトで `%USERPROFILE%\.ssh2\authorization`)。

```
Key     id_rsa_3072_a.pub
```

この指示によって Tectia Server は、ログインを認証する際に有効な公開鍵として `id_rsa_3072_a.pub` を使用します。

- OpenSSH サーバでは、公開鍵を OpenSSH 公開鍵ファイル・フォーマットに変換して、ユーザの `.ssh` ディレクトリの `authorized_keys` ファイルに保存する必要があります。

  サーバ上で公開鍵を OpenSSH 公開鍵ファイル・フォーマットに変換し、`$HOME/.ssh/authorized_keys` ファイルに加えます。これは、以下のようにリモート・コマンドを使って OpenSSH サーバ上で行えます。

  ```
  $ sshg3 ServerUser@open_server "ssh-keygen -i -f id_rsa_3072_a.pub >> \
  .ssh/authorized_keys"
  ```

6. `ssh-broker-config.xml` ファイルで公開鍵認証が有効になっていることを確認してください (デフォルトで有効になっています)。

   ```
   <authentication-methods>
     <auth-publickey />
   ...
   </authentication-methods>
   ```

   その他の認証方法も同様に設定ファイルに記載できます。その際、対話の必要が最も少ない方法を最初に配置します。

サーバ 側で、アカウントに対して公開鍵認証を許可するように設定されていれば、これで公開鍵認証を使用して クライアント から サーバ にログインできるようになります。

以下のようにログインを試みます。

```
Client$ sshg3 Server
```

秘密鍵のパスフレーズの入力を求められるはずです。パスフレーズを入力すると、Secure Shell 接続が確立されます。

4.5.3. [公開鍵認証ウィザード] での鍵の作成

Windows と Linux では、Tectia の [**公開鍵認証ウィザード**] で鍵ペアを生成し、公開鍵をホストにアップロードできます。「公開鍵の生成」 及び 「公開鍵の自動アップロード」 を参照してください。このウィザードは、秘密鍵と公開鍵の 2 つの鍵ファイルを生成します。

新しい秘密鍵と公開鍵はローカル・コンピュータの `%APPDATA%\SSH\UserKeys` ディレクトリ (Windows の場合) または `$HOME/.ssh2/` ディレクトリ (Linux の場合) に保存されます。秘密鍵ファイルにはファイル拡張子がありません。公開鍵は秘密鍵と同じベース・ファイル名ですが、ファイル拡張子は `.pub` になります。

接続ブローカーの設定、デフォルト設定、及び関連する接続プロファイルで公開鍵認証が許可されていることを確認してください (デフォルトでは許可されています)。デフォルト設定については 「認証の定義」 を、接続プロファイルについては 「認証の定義」 を参照してください。

公開鍵認証に鍵ペアを使用するには、公開鍵をリモート・ホスト・コンピュータにアップロードする必要があります。リモート・ホストで SFTP サーバが実行されている場合、ウィザードを使用して新しい公開鍵のコピーをサーバに自動的にアップロードできます。鍵の自動アップロードについては、「公開鍵の自動アップロード」 を参照してください。手動による鍵のアップロードについては、4.5.2 を参照してください。

公開鍵の生成

新しい鍵は Tectia コネクション設定 GUI で生成されます。**[ユーザ認証]** で **[鍵と証明書]** ページを選択し、**[鍵生成...]** をクリックして **[公開鍵認証ウィザード]** を開始します。

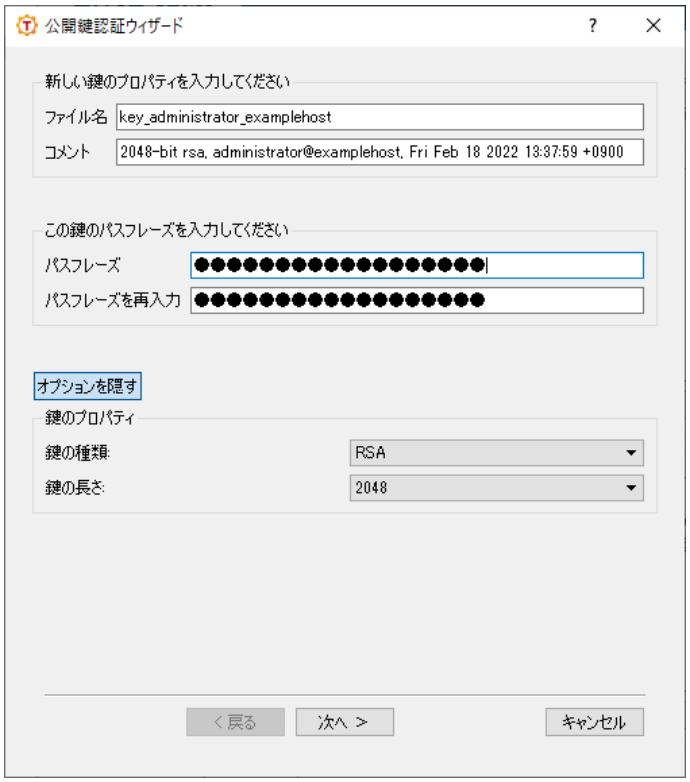


図4.5 [公開鍵認証ウィザード]

鍵のプロパティと、秘密鍵を保護するために必要なパスフレーズを定義します。鍵を使用して自分自身を認証する際には、常にこのパスフレーズの入力が要求されます。

[ファイル名]

鍵ファイルに付ける一意の名前を入力します。Tectia Client では、ユーザ名とホスト名を組み合わせた名前が提案されます。

[コメント]

このフィールドには、鍵ペアについて説明する、短いコメントを記入します。たとえば、その鍵が使用される接続について説明できます。このフィールドは必須ではありませんが、後で鍵を特定するのに役立ちます。

[パスフレーズ]

鍵を使用するときに入力しなければならないフレーズを入力します。このパスフレーズには、パスワードに似た機能があり、秘密鍵をある程度保護します。

注意

FIPS モードでは、暗号化されていない秘密鍵を FIPS モジュールからエクスポートすることが FIPS 規定において禁じられているため、パスフレーズなしでユーザ鍵を生成することはできません。

パスフレーズは推測されにくいものにしてください。理想はアルファベットと数字の両方を使用した 20 文字以上です。句読文字も使用できます。 パスフレーズも秘密鍵もネットワークに送信されることはありませんが、ローカルでアクセスできる場合は辞書攻撃が使用できます。使いやすさのためには、パスフレーズを空白にするのではなく認証エージェントの使用を推奨します。ssh-groker-g3 はデフォルトで認証エージェントとして機能します。

パスフレーズはしっかりと記憶し、書き留めないようにしてください。

[パスフレーズを再入力]

パスフレーズを再入力します。これにより、入力ミスがないことが確認できます。

生成する鍵の種類と、デフォルトとは異なる鍵の長さを定義するには、**[詳細オプション]** をクリックします。デフォルトでは、Tectia Client は 3072 ビットの RSA 鍵のペアを生成します。

[鍵のプロパティ] フィールドでは、以下の選択を行います。

[鍵の種類]

生成する鍵の種類を選択します。利用可能なオプションは Ed25519、 RSA (デフォルト)、 ECDSA 及び DSA です。

注意

FIPSモードでは、(FIPS 186-5 に準拠し) RSA、 ECDSA 及び Ed25519 はサポートされます。DSA は非推奨です。

[鍵の長さ]

生成する鍵の長さ (複雑さ) を選択します。以下のオプションから選択できます。

- DSA/RSA 鍵: 2048、 3072、 4096、 5120、 6144、 7168、 8192 ビット
- ECDSA 鍵: 256、384、521 ビット
- Ed25519 鍵: 256 ビット

同じ種類の鍵でも、大きいほどよりセキュアになりますが、それとともに生成時間が長くなります。256 ビットの ECDSA 鍵と 3072 ビットの RSA 鍵は同等のセキュリティを提供します。

[次へ] をクリックして、「公開鍵の自動アップロード」 の説明に従って、鍵のアップロードに進みます。

公開鍵の自動アップロード

SFTP　サブシステムが有効になっているサーバには、公開鍵を自動的にアップロードできます。**[公開鍵認証ウィザード]** では、選択したリモート・ホストに新規公開鍵がそれぞれ自動的にアップロードされます。このウィザードでは、既存のすべての鍵が一覧表示され、いつでも鍵を選択して他のリモート・サーバにアップロードすることもできます。

[公開鍵認証ウィザード] にアクセスするには、ツリー・ビューで **[ユーザ認証]** → **[鍵と証明書]** の順にクリックします。

[鍵と証明書のリスト] から鍵を選択し、**[アップロード]** をクリックします。

ウィザードの **[公開鍵をアップロード]** ビューで、鍵をアップロードするリモート・ホストを定義します。

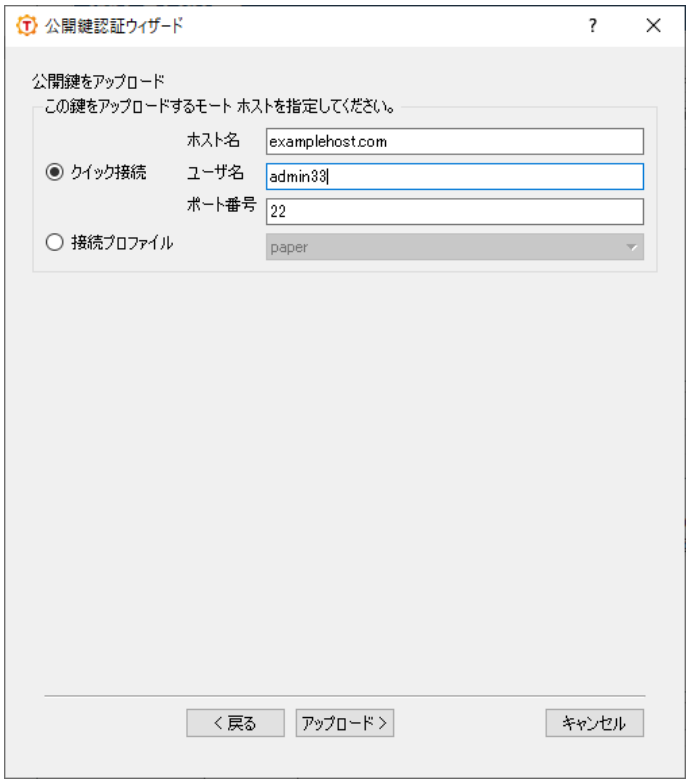


図4.6 鍵のアップロード

[クイック接続]

このオプションを選択して、リモートの **[ホスト名]** と、そこで使用する自分の **[ユーザ名]** を定義します。デフォルトの Secure Shell ポートは 22 です。

[接続プロファイル]

目的のリモート・ホストとユーザ名を規定する接続プロファイルをドロップダウン・リストから選択します。

[アップロード] をクリックして、選択したサーバに鍵をアップロードします。すでにリモート・サーバ・ホストに接続されている場合は、鍵のアップロードがすぐに開始されます。接続されていない場合は、サーバ上での認証が求められます (デフォルトではパスワードが必要です)。

公開鍵はデフォルトのユーザ・ホーム・ディレクトリ (Windows では `%USERPROFILE%\.ssh2`、Unix では `$HOME/.ssh2`) にアップロードされます。

注意

鍵ユーザは、サーバ上の鍵ディレクトリへの 書き込み パーミッションを持っている必要があり、そうでない場合は自動アップロードに失敗します。リモート・ホスト・コンピュータの管理者がユーザ・アクセスを制限している場合、サーバ設定で公開鍵認証が許可されていても、ユーザ自身が公開鍵認証を設定できないようになっていることがあります。

自動アップロードに成功しても、サーバ管理者がユーザのホーム・ディレクトリ以外の場所に鍵を保存するようにシステムを設定していることがあります。そのような場合は、追加する鍵と authorization ファイルを適切なディレクトリに手動で移動する必要があります。

自動アップロード機能を使用しない場合は、4.5.2 を参照してください。

4.5.4. OpenSSH で生成した鍵の使用

Tectia Client は、OpenSSH で生成したユーザ鍵ペアにも対応しています。OpenSSH 鍵は、`ssh-broker-config.xml` ファイルで `key-stores` エレメントを使用して指定できます。以下に設定例を示します。

```
<key-stores>
  <key-store type="software"
             init="key_files(/home/exa/keys/id_rsa.pub,/home/exa/keys/id_rsa)" />
  <key-store type="software"
             init="directory(path(/home/exa/.ssh))" />
</key-stores>
```

この例では、`id_rsa` という名前の鍵と、ユーザのデフォルトの OpenSSH 鍵ディレクトリ (ユーザのホーム・ディレクトリ下の `.ssh`) にあるすべての鍵が追加されます。

OpenSSH 鍵とディレクトリは Tectia コネクション設定ツールの [**鍵と証明書**] ページで追加できます。「鍵と証明書の管理」 を参照してください。

公開鍵は、標準的な SSH2 鍵と同じようにサーバにアップロードできます。4.5.2 及び 「公開鍵の自動アップロード」 を参照してください。

4.5.5. Windows サーバに関する特別な検討事項

Tectia Server 5.1 以前で Windows ドメイン・ユーザ・アカウントに公開鍵認証を使用してログオンする場合、ログオンを試みる際にユーザ名として `DOMAIN\user` を指定する必要があります。Unix のコマンドラインでは、以下の例のようにバックスラッシュをエスケープする必要があります。

```
$ sshg3 DOMAIN\\user@win-server
```

Tectia Server 5.2 以降ではこの操作は必要ありません。Tectia Server 5.2 以降が動作し、Windows ドメインに属しているマシンにログオンするときは、ユーザ・アカウントはデフォルトでデフォルト・ドメインのドメイン・アカウントであるとみなされます。代わりに同じ名前のローカル・アカウントでログオンする場合は、マシン名を「ドメイン」として指定する必要があります。たとえば、Windows のコマンドラインでは以下のように指定します。

```
> sshg3 MACHINE\user@machine
```

4.6. 証明書によるユーザ認証

技術的には、証明書認証は公開鍵認証方法の一部です。秘密鍵による署名の作成と、公開鍵 (証明書認証を行う場合は X.509 証明書に含まれる) を使用した署名の検証は、従来の公開鍵でも証明書でも同様に行われます。大きな違いは、特定のユーザが特定の公開鍵または証明書でログインすることを許可するかどうか決定する点です。従来の公開鍵では、すべてのサーバがすべてのユーザの公開鍵を持っていなければなりませんが、それに対して証明書を使う場合は、ユーザの公開鍵をサーバに配布する必要はなく、CA の公開鍵 (自己署名証明書) を配布するだけで十分です。

簡単に説明すると、証明書認証は以下のような仕組みになっています。

1. クライアントがユーザ証明書 (ユーザの公開鍵を含む) をサーバに送信します。このパケットには、ユーザの秘密鍵で署名された、セッションに固有のデータも含まれています。
2. サーバが CA 証明書 (及び必要に応じて外部リソース) を使用して、ユーザの証明書が有効であることを確認します。
3. サーバが、初期パケットに含まれる署名を確認することで、ユーザが有効な秘密鍵を所有しているかどうかを検証します。
4. サーバがユーザ証明書をサーバの設定ファイルのルールと照合し、ログインを許可するかどうかを決定します。

4.6.1. 設定ファイルの使用 (Unix)

X.509　証明書で自身を認証するようにクライアントを設定するには、以下の手順に従ってください。

1. 自分用の証明書を登録します。これを行うには、たとえば **ssh-cmpclient-g3** または **ssh-scepclient-g3** コマンドライン・ツールを使用します。

   例: **ssh-cmpclient-g3** を使用した鍵の生成と登録

   ```
   $ ssh-cmpclient-g3 INITIALIZE
   -P generate://ssh2:passphrase@rsa:3072/user_rsa \
   -o /home/user/.ssh2/user_rsa -p 62154:ssh \
   -s 'C=FI,O=SSH,CN=user;email=user@example.org' \
   -S http://fw.example.com:1080 http://pki.example.com:8080/pkix/ \
   'C=FI, O=SSH, CN=Test CA 1'
   ```

2. 鍵と証明書を、接続ブローカーが探せるディレクトリに配置します。

   接続ブローカーはデフォルトで、Unix では `$HOME/.ssh2` ディレクトリ、Windows では `%APPDATA%\SSH\UserKeys` 及び `%APPDATA%\SSH\UserCertificates` ディレクトリで見つかった各鍵を試行します。

   Tectia コネクション設定 ツールの [**鍵と証明書**] ページで、鍵がある他のディレクトリを追加することもできます。「鍵と証明書の管理」 を参照してください。Unix では `ssh-broker-config.xml` ファイルの `general/key-stores/key-store` エレメントを使用できます。「鍵ストアの設定例」t を参照してください。

3. (オプション) identification ファイルを作成します。

   すべての鍵をデフォルトのディレクトリに保存し、すべての鍵を公開鍵認証及び/または証明書認証に使用することを許可する場合、`identification` ファイルを使用する必要はありません。`identification` ファイルが存在しない場合、接続ブローカーはデフォルトのディレクトリで見つかった各鍵を試行します。`identification` ファイルが存在する場合、そのファイルに記載されている鍵が最初に試されます。

   ソフトウェア証明書の秘密鍵を `$HOME/.ssh2/identification` ファイルで指定します (`CertKey` オプションは `IdKey` オプションと同様に機能します)。

   ```
   CertKey     user_rsa
   ```

   証明書自体は `user_rsa.crt` から読み込まれます。

   identification ファイルの構文の詳細については、 `$HOME/.ssh2/identification` を参照してください。

4. `ssh-broker-config.xml` ファイルで公開鍵認証が有効になっていることを確認します (デフォルトで有効になっています)。

   ```
   <authentication-methods>
     <auth-publickey />
   ...
   </authentication-methods>
   ```

   その他の認証方法も同様に設定ファイルに記載できます。その際、対話の必要が最も少ない方法を最初に配置します。

4.6.2. Windows での証明書によるユーザ認証の設定

Windows では、Tectia コネクション設定　GUI を使用して X.509 証明書を使用したユーザ認証を設定できます。また、証明書によるユーザ認証のために Tectia Server を設定する必要があります。『Tectia Server Administrator Manual』を参照してください。

1. Tectia コネクション設定　GUI を起動します。

   Windows タスクバーの通知領域で を右クリックし、 **[設定]** を選択します。

2. **[一般]** で **[デフォルト接続]** をクリックします。**[認証]** タブを選択します。公開鍵認証が有効になっており、一覧の中で最初または唯一の方法であることを確認します。デフォルトでは有効になっています。

   [公開鍵認証] では、認証に公開鍵、証明書、またはその両方を使用するかどうかを選択できます。

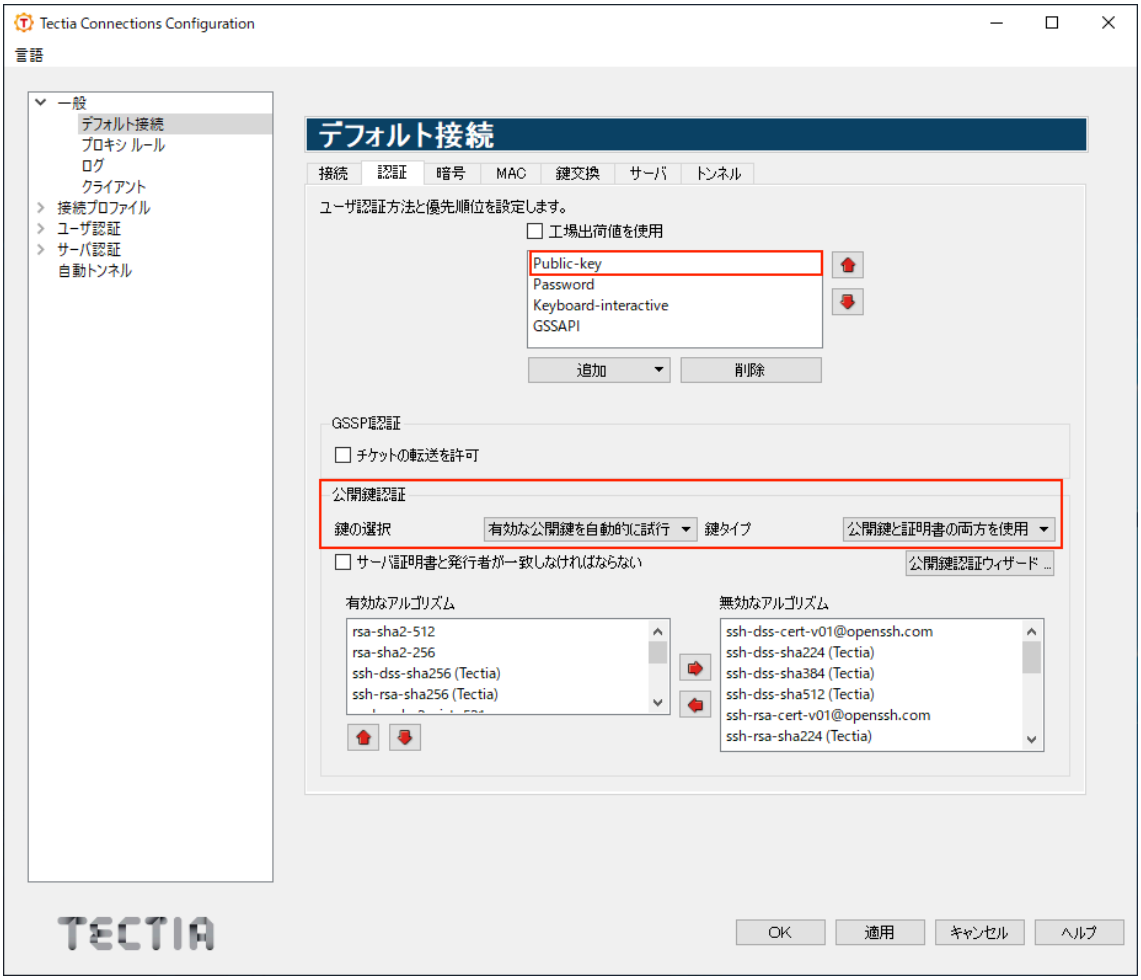


図4.7 公開鍵認証の有効化

3. 接続プロファイルを使用している場合は、**[接続プロファイル]** でプロファイル名を選択します。**[認証]** タブを選択し、公開鍵認証が有効になっていることを確認します。

4. Tectia は、ユーザの個人ストアである Microsoft 証明書ストアに証明書をインストールすることを勧めます。

5. **[ユーザ認証]** で **[鍵プロバイダ]** を選択します。**[マイクロソフト Crypto API を有効にする]** チェックボックスを選択し、**[適用]** をクリックします。

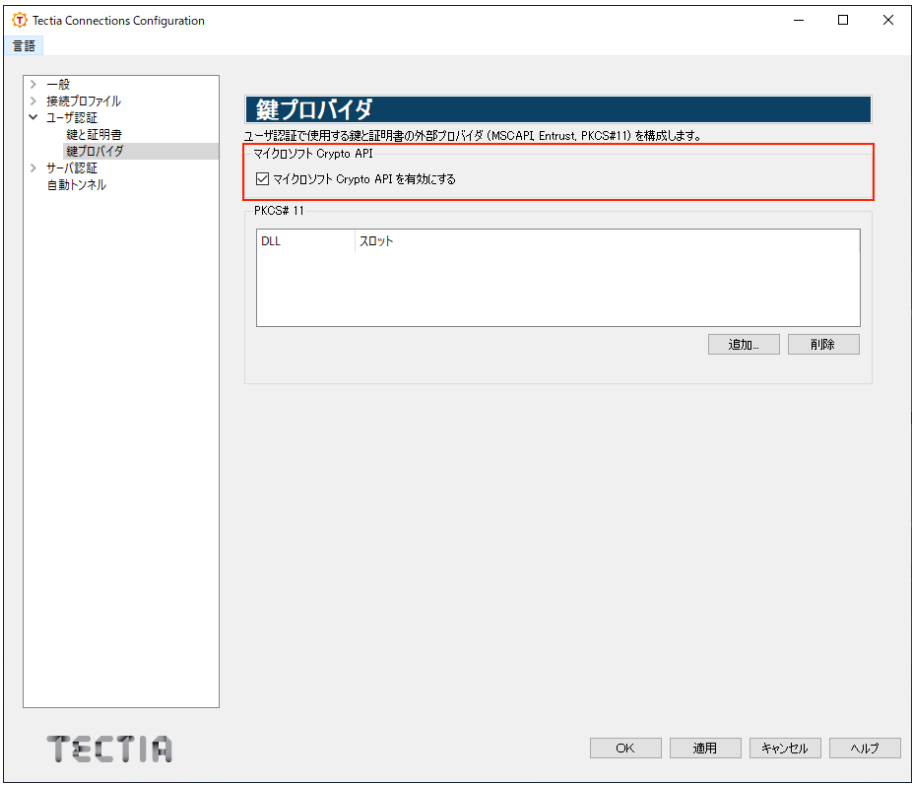


図4.8 証明書プロバイダとしてマイクロソフト Crypto API を有効にする

USB トークンやスマートカードが対応していれば、マイクロソフト Crypto API 経由で証明書情報を読み取ることもできます。また、**[PCKS#11]** を有効にすることで、USB トークンやスマートカードを使用することができます。

6. これで証明書が自動的にクライアントに読み込まれます。**[ユーザ認証]** で **[鍵と証明書]** を選択します。使用できる証明書は **[鍵と証明書のリスト]** で確認できます。

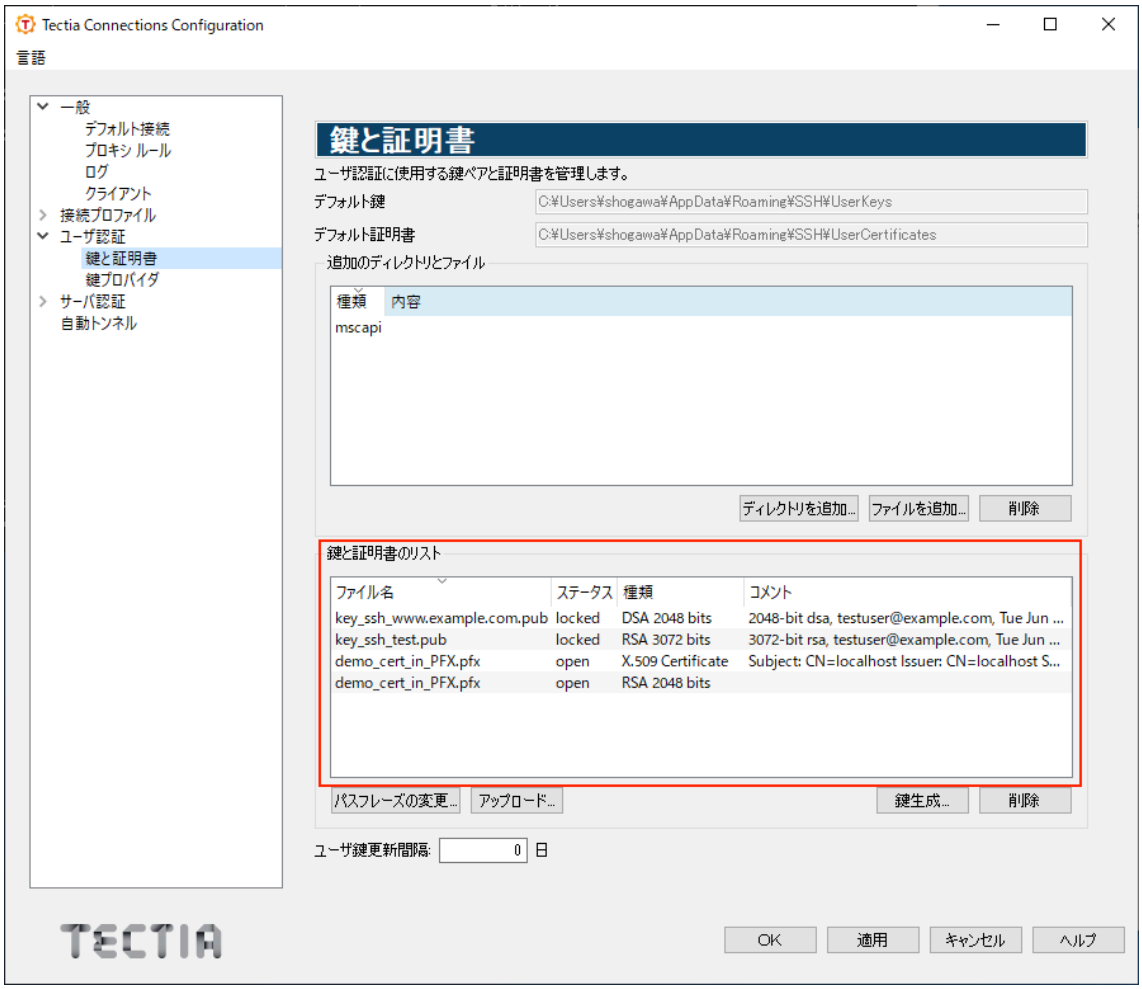


図4.9 使用できる証明書の確認

Tectia Client はファイル・システムから鍵や証明書の情報を読み取ることもできます。それらは **[追加のディレクトリとファイル]** で定義できます。

注意

クライアント証明書がクライアント認証専用に設定されていることを確認してください。そうすることで、複数の証明書のトラブルシューティングが容易になります。たとえば、サーバ認証の証明書をユーザ証明書としては使用することはできないからです。

鍵と証明書の設定について、詳しくは 「鍵と証明書の管理」 を参照してください。

証明書によるユーザ認証のトラブルシューティング

何らかの理由で証明書認証が成功しない場合、Tectia Server をトラブルシューティング・モードで実行し、トラブルシューティング・ログを確認することで、エンドユーザ接続に関する多くの情報を得ることができます。詳細については、『Tectia Server Administrator

Manual』の「Starting Tectia Server in Debug mode on Windows」の項を参照してください。

4.6.3. Tectia コネクション設定　GUI による PKCS 証明書のインポート

Tectia コネクション設定　GUI の [ユーザ認証] にある [**Keys and Certificates**] ページで、既存の PKCS #12、PKCS #7 及び X.509 証明書をインポートできます。「鍵と証明書の管理」 を参照してください。

4.7. ホストベースのユーザ認証 (Unix)

ホストベース認証はクライアント・マシンのホスト公開鍵を使用して、リモート・サーバに対してユーザを認証します。ホストベース認証は Unix 上の Tectia Client で使用できます。リモートの Secure Shell サーバは Unix、Windows、または z/OS サーバのいずれかを使用できます。

通常、ホストベース認証の設定にはサーバの管理者権限　(root)　が必要です。設定方法は『Tectia Server Administrator Manual.』に記載されています。

4.8. キーボード・インタラクティブによるユーザ認証

キーボード・インタラクティブは、さまざまな種類の認証メカニズムを実装するために使用できる、汎用的な認証方法です。現在サポートされている、ユーザの入力のみを必要とする認証方法であれば、キーボード・インタラクティブで実行できます。

サポートされるキーボード・インタラクティブのサブメソッドは、Secure　Shell　サーバによって異なります。一般的にサポートされているサブメソッドには、パスワード、RSA SecurID、RADIUS、PAM 認証などがあります。

注意

サーバがオプションのサブメソッドを複数許可している場合、クライアントは特定のキーボード・インタラクティブ・サブメソッドを要求できません。サブメソッドが提供される順番は、サーバの設定に依存します。しかし、たとえばサーバが SecurID　とパスワードの　2　つのオプションのサブメソッドを許可している場合、サーバから　SecurID　が提供されたときに　Enter　キーを押すことによって、ユーザは　SecurID　をスキップできます。その後、ユーザにはパスワードの入力が求められます。

4.8.1. 設定ファイルによるキーボード・インタラクティブ認証方法の定義 (Unix)

Tectia　Client　でキーボード・インタラクティブ認証を有効にするには、`ssh-broker-config.xml` ファイルに以下の行があることを確認してください。

```
<authentication-methods>
...
  <auth-keyboard-interactive />
...
</authentication-methods>
```

4.8.2. GUI によるキーボード・インタラクティブ認証方法の定義

キーボード・インタラクティブ認証の使用は接続ブローカーの設定です。Tectia コネクション設定 GUI を使用して認証方法を管理する方法については、「認証の定義」 で説明されています。

4.9. GSSAPI によるユーザ認証

GSSAPI (ジェネリック・セキュリティ・サービス・アプリケーション・プログラミング・インターフェイス) は、機構に依存しない方法でアプリケーションにセキュリティ・サービスを提供する関数インターフェイスです。これにより、標準化された 1 つの API で異なるセキュリティ機構を利用できます。多くの場合 GSSAPI は、GSSAPI では最も一般的な Kerberos と関連しています。

Linux プラットフォームでは Kerberos ライブラリがデフォルトでインストールされます。他のほとんどの Unix プラットフォームでも使用できますが、別途インストールする必要があります。

Windows の場合、GSSAPI は Windows 2003 (またはそれ以降) のネットワークで Kerberos による統合認証を提供します。ローカル・アカウントはマシンの境界を越えて移動できないため、この方法ではドメイン・アカウントを使用します。

GSSAPI 認証方法には (設定を除いて) ユーザ・インターフェイスがありません。つまり、ユーザは何も求められません。GSSAPI 交換時に何らかの障害が発生した場合、その原因はクライアントのデバッグ・ログで確認できます。

4.9.1. 設定ファイルによる GSSAPI 認証方法の定義 (Unix)

クライアントで GSSAPI 認証を有効にするには、`ssh-broker-config.xml` ファイルに以下の行があることを確認してください。

```
<authentication-methods>
  <auth-gssapi />
...
</authentication-methods>
```

その他の認証方法も同様に設定ファイルに記載できます。その際、対話の必要が最も少ない方法を最初に配置します。

4.9.2. GUI による GSSAPI 認証方法の定義

Tectia コネクション設定 GUI を使用した認証方法の管理については、「認証の定義」 で説明されています。

第5章 ファイルの転送

Tectia Client 及び Tectia Server には、Secure File Transfer プロトコル (SFTP) を使用したセキュアなファイル転送の基本機能があります。

また既存の FTP ファイル転送のセキュリティを透過的に高めるために、別製品として Tectia ConnectSecure も用意されています。この製品には、基本的な Secure Shell クライアント・サービスに加え、FTP-SFTP 変換や透過的 FTP トンネリング、SFTP アプリケーション・プログラミング・インターフェイス (API) も含まれています。Tectia ConnectSecure の詳細については、『Tectia ConnectSecure Product Description』及び『Administrator Manual』を参照してください。

本章では、SCP 及び SFTP コマンドライン・ツール、及び SFTP グラフィカル・ユーザ・インターフェイス (GUI) を使用したセキュアなファイル転送の手順について説明します。

5.1. scpg3 及び sftpg3 コマンドによるセキュア・ファイル転送

Tectia Client は、セキュアなファイル転送のために **scpg3** (セキュア・コピー) コマンドと **sftpg3** コマンドを提供します。これらのコマンドライン・クライアントは Secure File Transfer プロトコル (SFTP) を利用します。

scpg3 及び **sftpg3** コマンドでファイルをアップロードする際、ファイルには `TRUNCATE` フラグが設定されます。ファイル転送が完了するまで、ファイル・サイズは 0 と表示されます。

これらのセキュアなファイル転送のコマンドは暗号処理と認証作業を接続ブローカーに依存しているため、接続ブローカーがまだ起動していない場合、接続ブローカー(**ssh-broker-g3** プロセス) をオン・デマンド・モードで起動します。

複数のファイル転送コマンドを同時に起動するスクリプトで **scpg3** 及び **sftpg3** コマンドライン・クライアントを使用する場合、接続ブローカーがすでにバックグラウンドで実行されている必要があります。接続ブローカーが起動して動作し始めるまでに数秒かかるので、スクリプトがすぐに開始されないように注意してください。接続ブローカーがまだ起動中の場合、スクリプトが失敗する可能性があります。

接続ブローカーを起動するには、 **ssh-broker-g3** コマンドを実行します。詳細については、ssh-broker-g3(1) を参照してください。

5.1.1. scpg3 の使用

scpg3 (Windows の場合は **scpg3.exe**) は、ネットワーク上でファイルをセキュアにコピーするために使用されます。 **scpg3** は **ssh-broker-g3** を使用して、Secure Shell バージョン 2 プロトコルを使用したセキュア・トランスポートを提供します。リモート・ホストは、 **sftp-server** (または **sft-server-g3**) サブシステムが有効になっている Secure Shell バージョン 2 サーバを実行している必要があります。

scpg3 の基本構文は以下の通りです。

```
scpg3 user@source:/directory/file user@destination:/directory/file
```

scpg3 を使用すると、ローカル・システムからリモート・システムへ、またはその逆方向へファイルをコピーできます。2 つのリモート・ホスト間のコピーも許可されます。ローカル・パスは、`user@system:` のプレフィックスを付けずに指定できます。相対パスも使用でき、ユーザのホーム・ディレクトリを基準として解釈されます。

Windows のパスの先頭にはスラッシュ ("/") を付ける必要があります。たとえば、ローカルのファイルをリモートの Windows サーバにコピーする場合、以下のように指定します。

```
scpg3 localfile user@destination:/C:/directory/file
```

コマンドライン・オプションの詳細については、scpg3(1) を参照してください。

5.1.2. sftpg3 の使用

sftpg3 (Windows の場合は **sftpg3.exe**) は、ネットワーク上でのセキュアなファイル転送に使用できる FTP ライクなクライアントです。 **sftpg3** は **ssh-broker-g3** を使用して、Secure Shell バージョン 2 プロトコルを使用したセキュア・トランスポートを提供します。

sftpg3 は **ftp** のように機能しますが、接続には FTP デーモンや FTP クライアントを使用しません。 **sftpg3** を使用すると、 **sftp-server** (または **sft-server-g3**) サブシステムが有効になっている Secure Shell バージョン 2 サーバを実行している、任意のホストに接続できます。

sftpg3 の基本構文は以下の通りです。

```
sftpg3 user@host
```

sftpg3 にはローカルとリモートの 2 つの接続エンド・ポイントがあり、どちらも Tectia Client のホスト以外のホストに接続できます。デフォルトでは、ローカル・エンド・ポイントは Tectia Client のホストのファイル・システムに接続され、リモート・エンド・ポイントはコマンドラインで定義されたホストに接続されます (コマンドラインでホストが定義されていない場合は未接続のままになります)。

対話的に起動すると、 **sftpg3** は従来の **ftp** プログラムと同様に SFTP コマンドを入力するためのプロンプトを表示します。また、実行するコマンドを記述したバッチ・ファイルを使って、 **sftpg3** を非対話的に起動することもできます。

コマンドライン・オプションとコマンドの詳細については、sftpg3(1) を参照してください。

5.1.3. 拡張ファイル転送機能

Tectia Client のコマンドライン・ツール **scpg3** 及び **sftpg3**では、以下の拡張ファイル転送機能を利用できます。

- 大きな ファイルを転送するためのチェックポイント/リスタート (IETF 対応の SSH サーバで使用)
- 使用する前にファイルが完全に転送されたことを確認するためのプレフィックス (IETF 対応の SSH サーバで使用)
- ファイル転送速度を改善するためのストリーミング (Tectia Server で使用)

コマンドの詳細については、scpg3(1) 及び sftpg3(1) を参照してください。

5.2. セキュアなファイル転送 GUI (Windows)

Windows 上の Tectia Client にはセキュアなファイル転送 GUI があり、リモート・ホスト・コンピュータからローカル・コンピュータへのファイルのダウンロードや、リモート・ホストへのファイルのアップロードが容易に行えます。Tectia セキュア・ファイル転送 GUI は Windows エクスプローラと似た作動をします。

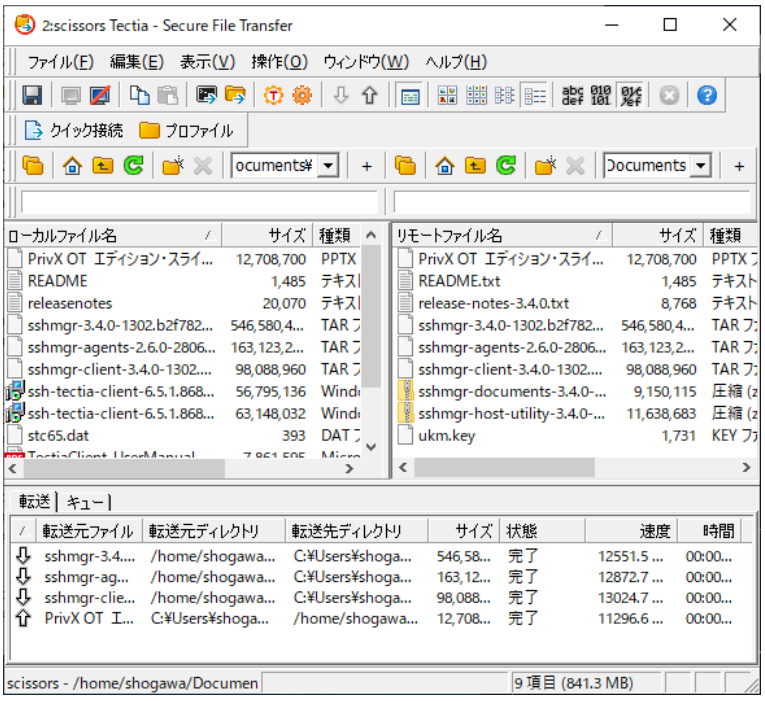


図5.1 Tectia セキュア・ファイル転送 GUI

Tectia セキュア・ファイル転送 GUI では、接続ブローカーの設定で定義された接続プロファイルを使用するか、[**クイック接続**] オプションを使用してリモート・ホストに接続できます。

5.2.1. セキュアなファイル転送 GUI の設定の定義

Tectia セキュア・ファイル転送 GUI のデフォルト設定を行う方法については、B.1.5 で説明しています。

5.2.2. Tectia セキュア・ファイル転送 GUI を使用したファイルのダウンロード

1 つのファイル、または複数のファイルを同時にダウンロードするには、さまざまな方法があります。`Shift` キーや `Control` キーによる複数ファイルの選択は、Windows のエクスプローラと同じように機能します。

ドラッグ・アンド・ドロップ

ドラッグ・アンド・ドロップは、ファイルをダウンロードする最も簡単な方法です。ダウンロードするファイルを選択し、マウスのボタンを押したまま、ダウンロードする場所 (Windows のデスクトップなど) にファイルを移動させて、ボタンを離すだけです。

ダウンロード・ボタン 

選択したファイルをダウンロードするには、ツールバーの [**ダウンロード**] ボタンをクリックします。

ショートカット・メニュー

リモート・ビューでファイルを右クリックすると、ショートカット・メニューが表示されます。すぐに転送を行う場合は [**ダウンロード**] を、そのファイル用に別の転送先フォルダを定義する場合は [**ダウンロード ダイアログ**] を選択します。

[**ダウンロード ダイアログ**] オプションを選択した場合は、[**ダウンロード - フォルダの選択**] ダイアログが表示されます。これは標準の Windows のファイル選択ダイアログで、選択したファイルのダウンロード先の場所を選択できます。適切なフォルダ (または他の場所) の選択選択後、現在のダウンロードのステータスが転送ビューに表示されます。

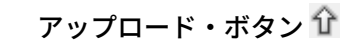
ファイル・ビューの上にある検索バーを使用すると、グロブ・パターンを使って表示するファイルを絞り込むことができます。

5.2.3. Tectia セキュア・ファイル転送 GUI を使用したファイルのアップロード

ファイル転送ウィンドウを使用して、ローカル・コンピュータからリモート・ホスト・コンピュータにファイルをアップロードできます。1 つのファイル、または複数のファイルを同時にアップロードするには、さまざまな方法があります。`Shift` キーや `Control` キーによる複数ファイルの選択は、Windows のエクスプローラと同じように機能します。

ドラッグ・アンド・ドロップ

ドラッグ・アンド・ドロップは、ファイルをアップロードする最も簡単な方法です。アップロードするローカル・ファイル (デスクトップや Windows エクスプローラなどにあるもの) をクリックし、マウスのボタンを押したまま、[**ファイル転送**] ウィンドウのファイル・ビューにファイルを移動し、ボタンを離すだけです。

アップロード・ボタン 

選択したファイルをアップロードするには、ファイル転送ウィンドウにあるツールバーの [**アップロード**] ボタンをクリックします。

ショートカット・メニュー

ローカル・ビューでファイルを右クリックするか、またはリモート・ビューで何もない場所を右クリックすると、ショートカット・メニューが表示されます。すぐに転送を行う場合は [**アップロード**] を、アップロードするファイルを選択する場合は [**アップロード ダイアログ**] を選択します。

[**アップロード ダイアログ**] オプションを選択した場合は、[**アップロード - ファイルの選択**] ダイアログが表示されます。これは標準の Windows のファイル選択ダイアログで、アップロードするファイルを選択できます。ファイルの選択後、現在のアップロードのステータスが転送ビューに表示されます。

ファイル・ビューの上にある検索バーを使用すると、グロブ・パターンを使って表示するファイルを絞り込むことができます。

5.2.4. [転送] タブと [キュー] タブ

Tectia のファイル転送ビューの下部には、[**転送**] 及び [**キュー**] の 2 つのタブがあります。図 5.1 を参照してください。

[**転送**] タブには、ローカル・コンピュータとリモート・コンピュータ間ですでに転送されたファイルのリストが表示されます。

[**キュー**] タブでは、後からアップロードやダウンロードを行うファイルのリストをカスタマイズして作成できます。マウスを使って [キュー] タブにファイルをドラッグ・アンド・ドロップすると、ファイルは別の転送コマンドを待機する状態になります。

リモート・ホストにログインした後で [**キュー**] ページを右クリックすると、以下のオプションのショートカット・メニューが表示されます。

- [**追加**] では、選択したファイルをキューに追加します。[キュー] ページで右クリックして選択します。

  [**転送キューの編集**] ダイアログが表示されます。リスト領域の上にある [**新規作成**] をクリックして、転送する新しいファイルのパスを入力するか、または参照ボタン (...) をクリックしてファイルを参照します。

- キューへ投入済みのファイルの転送先を編集するには、編集するファイルを選択し、[キュー] ページで右クリックして、[**編集**] を選択します。

  [**転送キューの編集**] ダイアログが開いたら、ファイルの新しい転送先ディレクトリを入力します。参照ボタン (...) をクリックすると、転送先ディレクトリを参照できます。

  [**編集**] オプションは複数のファイルに対して同時に使用できますが、転送の方向 (アップロードまたはダウンロード) はすべてのファイルで同じである必要があります。

- キューからファイルを削除するには、ファイルを選択し、ページで右クリックして、[**削除**] を選択します。

- 単一のファイルを転送するには、ファイルを選択し、[キュー] ページで右クリックして、[**転送**] を選択します。

- キューへ投入済みのファイルをすべて転送するには、[キュー] ページで右クリックして、[**すべて転送**] を選択します。

5.2.5. ファイル・プロパティの定義

ローカル・ビューまたはリモート・ビューでファイルを選択し、[**操作**] → [**プロパティ**] (またはショートカット・メニューの [**プロパティ**]) を選択すると、ファイル・プロパティの一部を表示して変更できるダイアログが開きます。

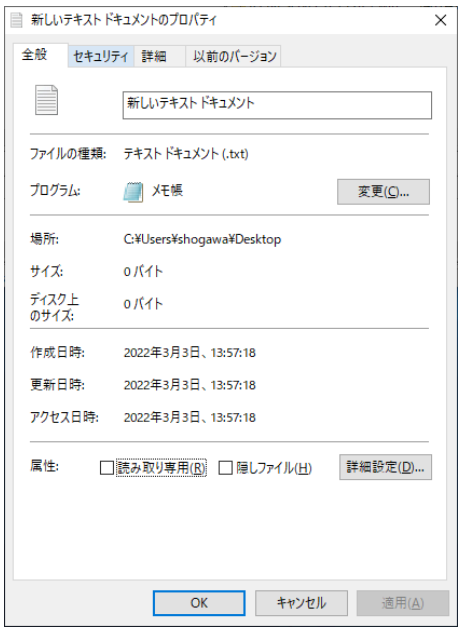


図5.2 ファイルのプロパティ・ページ

プロパティを表示し、必要に応じて定義します。たとえば、ファイルの読み取り/書き込みパーミッションを変更したり、ファイルを隠しファイルとして定義したりできます。

詳細については、オペレーティング・システムのマニュアルを参照してください。

5.2.6. Windows エクスプローラとの違い

ファイル転送ウィンドウの操作は Windows エクスプローラとほとんど同じです。ただし、自分のコンピュータのローカル (Windows エクスプローラなど) でファイルを扱うのと、ホスト・コンピュータのセキュアなリモート接続を介してファイルを扱う (Tectia Client のファイル転送など) のとでは性質が異なるため、操作に多少の違いがあります。

フォルダの削除

空でないリモート・フォルダを削除することはできません。そのフォルダ内のファイルとサブフォルダを先に削除してください。

複数の貼り付け操作

コピーと貼り付けの操作では、ファイルを貼り付ける際にファイル名を変更できません。そのため、Windows エクスプローラのように、1 つの場所に何度もファイルを貼り付けて、貼り付けたファイルのコピーを複数作成することはできません。

注意

転送されるファイルの最大サイズは、ファイル・システムによってのみ制限されます。(多くのシステムで、最大ファイル・サイズは 2 ギガバイトです)

5.3. ファイル転送の制御

現在の Secure File Transfer プロトコル (SFTP) は、転送するファイルに関する情報を一切持たず、ファイルの内容のみをバイト・ストリームとして転送します。Unix 系のファイルでは、送り手と受け手が同じ CCS を使っていれば、これで十分です。しかし、コマンドライン・ツールで **chmod** コマンドを使用することにより、転送されたファイルに対して特定のファイル・パーミッションを設定できます。

MVS データ・セットでは、どの転送フォーマットを使用するのか、どのコード・セットが関係するか、ファイルの特性は何かなど、Tectia にはより多くの情報が必要になります。Tectia は SFTP にいくつかの拡張機能を導入しており、**scpg3** や **sftpg3** の site コマンドを使うことで情報を中継できます。

MVS のファイル転送について、詳しくは『Tectia Server for IBM z/OS User Manual』を参照してください。

5.3.1. site コマンド

コマンドの説明については、 sftpg3(1) の **site** と **lsite** のコマンド、及び scpg3(1) の `--dst-site` と `--src-site` のオプションを参照してください。

コマンドを指定する場合、完全なパラメータ名、またはその省略形のどちらも使用できます。たとえば、以下の 2 つのコマンドの結果は同じになります。

```
sftp> site x=bin
```

```
sftp> site transfer_mode=bin
```

使用できる site のパラメータを以下に挙げ、それぞれについて説明します。

表5.1 site パラメータ

パラメータ	略語	指定できる値
AUTOMOUNT	-	YES|NO|IMMED
[NO]AUTOMOUNT	[NO]AUTOM	-
AUTORECALL	-	YES|NO
[NO]AUTORECALL	[NO]AUTOR	-
BLKSIZE	B, BLOCKSI	size
BLOCKS	BL	-
CONDDISP	CO	CATLG|UNCATLG|KEEP|DELETE
CYLINDERS	CY	-
DATACLAS	DA	class
DATASET_SEQUENCE_NUMBER	SEQNUM	number
DEFER	DE	YES|NO
[NO]DEFER	-	-
DIRECTORY_SIZE	M, DI, DIRSZ	size
EXPIRY_DATE	EXPDT	yyddd|yyyyddd
FILE_STATUS	STATUS	NEW|MOD|SHR|OLD
FILETYPE	FILET	SEQ|JES
FIXRECFM	FI	length
JOB_ID	JESID	ID
JOB_OWNER	JESO	name
JOBNAME	JESJOB	name
KEYLEN	KEYL	length
KEYOFF	KEYO	offset
LABEL_TYPE	LABEL	NL|SL|NSL|SUL|BLP|LTM|AL|AUL
LIKE	-	like
LRECL	R, LR	length
MGMTCLAS	MG	class
NORMDISP	NOR	CATLG|UNCATLG|KEEP|DELETE
PRIMARY_SPACE	PRI	space
PROFILE	P, PROF	profile
RECFM	O, REC	recfm
RECORD_TRUNCATE	U, TRUN	YES|NO
[NO]TRUNCATE	[NO]TRU, [NO]TRUN	-
RETENTION_PERIOD	RET	days
SECONDARY_SPACE	SE, SEC	space
SIZE	L	size

パラメータ	略語	指定できる値
SPACE_RELEASE	RLSE	YES|NO
SPACE_UNIT	SU	BLKS|TRKS|CYLS|AVGRECLEN
SPACE_UNIT_LENGTH	SUL	length
STAGING	S, STAGE	YES|NO
STORCLAS	ST	class
SVC99_TEXT_UNITS	SVC99	string
TRACKS	TR	-
TRAILING_BLANKS	TRAIL	YES|NO
[NO]TRAILINGBLANKS	[NO]TRAI, [NO]TRAIL	-
TRANSFER_CODESET	C, CODESET	codeset
TRANSFER_FILE_CODESET	D, FCODESET	codeset
TRANSFER_FILE_LINE_DELIMITER	J, FLDELIM	UNIX|MVS|MVS-FTP|DOS|MAC|NEL
TRANSFER_FORMAT	F, FORMAT	LINE|STREAM|RECORD
TRANSFER_LINE_DELIMITER	I, LDELIM	UNIX|MVS|MVS-FTP|DOS|MAC|NEL
TRANSFER_MODE	X, MODE	BIN|TEXT
TRANSFER_TRANSLATE_DSN_TEMPLATES	A, XDSNT	templates
TRANSFER_TRANSLATE_TABLE	E, XTBL	table
TYPE	T	PS|PO|PDS|POE|PDSE|GDG| HFS|VSAM|ESDS|KSDS|RRN
UNIT	UN	unit
UNIT_COUNT	UC, UNC	number
UNIT_PARALLEL	UNP	YES|NO
VOLUME_COUNT	VC, VOLCNT	number
VOLUMES	VO, VOL	vol1+vol2+...

AUTOMOUNT=YES|NO|IMMED

YES に設定すると、データ・セットがオンラインでないために通常の割り当てに失敗した場合、Tectia はそれを割り当て、マウントするようシステムに要求します。このためには、ユーザに SSZ.MOUNT ファシリティに対する読み込みパーミッションが付与されている必要があります。

NO に設定すると、オフラインのデータ・セットは自動的にマウントされません。

IMMED に設定すると、Tectia は通常の割り当てを試みず、データ・セットを直ちにマウントするようシステムに要求します。

デフォルト: NO

[NO]AUTOMOUNT|[NO]AUTOM

AUTOMOUNT|AUTOM は AUTOMOUNT=YES と同じです。
NOAUTOMOUNT|NOAUTOM は AUTOMOUNT=NO と同じです。

AUTORECALL=YES|NO

ストレージ・マネージャによって移行されたデータ・セットが自動的に呼び出されるかどうかを定義します。

デフォルト: YES

[NO]AUTORECALL|[NO]AUTOR

AUTORECALL|AUTOR は AUTORECALL=YES と同じです。
NOAUTORECALL|NOAUTOR は AUTORECALL=NO と同じです。

BLKSIZE|B|BLOCKSI= size

最大ブロック・サイズを指定します。

デフォルト: none

BLOCKS|BL

領域割り当て単位がブロックであることを指定します。 SPACE_UNIT=BLKS と同じです。

CONDDISP|CO=CATLG|UNCATLG|KEEP|DELETE

ファイル転送が完了前に終了した場合 (クライアントまたはサーバは稼働しているが、相手側と切断された場合。たとえば、クライアント側で**CTRL+C** を押したとき) の出力ファイルの処理を指定します。

注意

クライアント (ローカルまたはクライアント側に転送する場合) またはサーバ (リモートまたはサーバ側に転送する場合) が停止すると、それらの処理は制御できなくなります。

オプションには、ファイルの種類 (MVS または HFS) に応じて以下の効果があります。

- CATLG: MVS データ・セットが保持され、その名前がカタログ化されます。HFS ファイルは保持されます。
- UNCATLG: MVS データ・セットの名前はカタログから削除されますが、データ・セットは保持されます。HFS ファイルは保持されます。
- KEEP: MVS データ・セットが保持されます (カタログ化されていればカタログ化されたまま、カタログ化されていなければカタログ化されていないままになります)。HFS ファイルは保持されます。
- DELETE: MVS データ・セットの名前はカタログから削除され、データ・セットに割り当てられた領域は解放されます。HFS ファイルは削除されます。

デフォルト: CATLG

CYLINDERS|CY

領域割り当て単位がシリンダであることを指定します。 SPACE_UNIT=CYLS と同じです。

DATACLAS|DA= class

データ・セットのデータ・クラスを指定します。

デフォルト: none

DATASET_SEQUENCE_NUMBER|SEQNUM= number

テープ・ボリューム上のデータ・セットの相対位置を特定します。

デフォルト: システム・デフォルト

DEFER|DE=YES|NO

データ・セットの割り当てを、割り当て段階からデータ・セットを開く時点まで延期するかどうかを指定します。

YES に設定すると、データ・セットが開かれるまでデータ・セットの割り当てが延期されます。

NO に設定すると、データ・セットは割り当て段階で割り当てられます。

デフォルト: NO

[NO]DEFER|DE

DEFER|DE は DEFER=YES と同じです。
NODEFER は DEFER=NO と同じです。

DIRECTORY_SIZE|M|DI|DIRSZ= size

ディレクトリの 256 バイトのレコード数を指定します。

デフォルト: 10

EXPIRY_DATE|EXPDT= yyddd|yyyyddd

新しいデータ・セットの有効期限を指定します。この日付以降、オペレーティング・システムはデータ・セットを削除または上書きできるようになります。

デフォルト: システム・デフォルト

FILE_STATUS|STATUS=NEW|MOD|SHR|OLD

データ・セットのステータスを定義します。値を入力すると、データ・セットの割り当て時にその値が使用されます。この属性は、JCL の DD ステートメントの DISP パラメータの最初の値に対応します。以下の値を指定できます。

- NEW: データ・セットを作成します。

- MOD: 既存のデータ・セットに付加します。データ・セットが存在しない場合は、新しいデータ・セットが作成されます。
- SHR: 読み取り専用のデータ・セットを作成します。
- OLD: 既存のデータ・セットを指定します。

FILETYPE|FILET=SEQ|JES

ファイル・システムまたは z/OS ジョブ入力サブシステム (JES) のどちらとインターフェイスするか指定します。

FILETYPE=JES を使用すると、 **put** 及び **sput** コマンドは転送されたファイルを、実行のために内部リーダのジョブ・キューに投入し、 **get** コマンド及び **sget** コマンドはスプール・データ・セットを取得することが有効になります。JES とのインターフェイスを終了し、通常のファイル・アクセスに戻すには、ファイルの種類をシーケンシャル (SEQ) に戻すか、空の文字列 (つまり FILETYPE=) に設定します。ファイルの種類に空の文字列を入力すると、ファイルの種類がデフォルトに設定されます。

デフォルト: SEQ

FIXRECFM|FI= length

データ・セット編成が FB に設定され、固定レコード長が length に設定されます。

デフォルト: none

JOB_ID|JESID= ID

FILETYPE=JES モードの場合、 JOB_ID は、 **get** などの JES スプールにアクセスするコマンドが、指定した ID に一致するジョブ ID を持つジョブにのみ適用されるように指定します。

get、 **sget** などのコマンドでジョブ ID を指定すると、指定したジョブのスプール・ファイルを取得できます。

JOB_OWNER|JESO= name

FILETYPE=JES モードの場合、 JOB_OWNER は、 **ls** や **get** などの JES スプールにアクセスするコマンドが、指定された name と一致する所有者を持つジョブにのみ適用されることを指定します。

デフォルト: カレント・ユーザ

JOBNAME|JESJOB= name

FILETYPE=JES モードの場合、 JOBNAME は、 **ls** や **get** などの JES スプールにアクセスするコマンドが、指定された name と一致するジョブ名を持つジョブにのみ適用されることを指定します。

KEYLEN|KEYL= length

データ・セットで使用される鍵の長さをバイト単位で指定します。

デフォルト: none

KEYOFF|KEYO= offset

鍵オフセット (指定された VSAM データ・セットのレコードにおける鍵の最初のバイトの位置) を指定します。

デフォルト: none

LABEL_TYPE|LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL

データ・セットのラベルの種類。この属性は、JCL の DD ステートメントの LABEL パラメータの最初の値に対応します。

注意

site では、RACF または同等のセキュリティ製品を使用して、適切なリソースへのアクセスを制限することにより、 BLP 及び NL テープ処理の使用を制御することをお勧めします。

LIKE= like

RECFM、 BLKSIZE、及び LRECL 属性のコピー元となるモデル・データ・セットの名前を指定します。名前はカタログ化されたデータ・セットの完全な DSN でなければならず、その前には 3 つのアンダースコアが必要です。

PS データ・セットを作成し、モデルが PS データ・セットである場合を除き、 LIKE を使用する場合は TYPE 属性を含める必要があります。

デフォルト: none

LRECL|R|LR= length

最大レコード長または固定レコード長。

デフォルト: VSAM の場合は 4096、データ・セット編成が F または FB の場合は 80、それ以外の場合は 1024

MGMTCLAS|MG= class

データ・セットの管理クラスを指定します。

デフォルト: none

NORMDISP|NOR=CATLG|UNCATLG|KEEP|DELETE

ファイル転送が正常に終了した後に使用されるデータ・セットの処理を指定します。この属性は、JCL の DD ステートメントの DISP パラメータの 2 番目の値に対応します。

デフォルト: CATLG

PRIMARY_SPACE|PRI= space

データ・セットの一次領域割り当て。

デフォルト: none

PROFILE|P|PROF= profile

ファイル転送プロファイルは、ファイル転送に使用される名前付きプロファイルを指定します。プロファイル名の大文字と小文字は区別されます。特殊なプロファイル名 P=% ではプロファイルは使用されません。これにより、ファイル名に基づくプロファイルのマッチングも防止します。

デフォルト: none

RECFM|O|REC= recfm

RECFM はデータ・セット編成を指定します。以下の文字の有効な組み合わせであれば、あらゆる値を指定できます。

```
F       Fixed
V       Variable
U       Undefined
B       Blocked
S       Spanned or standard
M       Machine line printer codes
A       ASA line printer codes
```

デフォルト: VB

RECORD_TRUNCATE|U|TRUN=YES|NO

MVS データ・セットの書き込み中にレコードの切り詰めが発生した場合、RECORD_TRUNCATE を YES に設定すると、システムはデータ・セットの書き込みを継続します。RECORD_TRUNCATE を NO に設定したり、省略したりすると、システムは転送を中断します。

転送されたレコード (コード・セットと行区切り記号の変換後) の長さが、データ・セットの最大レコード長より大きい場合、レコードの切り詰めが発生します。切り詰めが発生するのは、TRANSFER_FORMAT を LINE または RECORD に設定しているときだけです。STREAM フォーマットでは転送データにレコードの概念はなく、すべてのレコードが最大長まで書き込まれます。

LINE 転送フォーマットでは、転送されるレコードの長さは、改行文字までの文字数です。

RECORD フォーマットでは、転送されるレコードの長さは、レコードの前にある 4 バイトのバイナリ長フィールドで指定されます。

データ・セット・レコードの最大長は、以下のようにデータ・セット編成によって異なります。

```
F and FB - LRECL
V and VB - LRECL-4
U        - BLKSIZE
VSAM     - MAXRECLEN
```

Tectia Client は、レコードの切り詰めが原因でデータ・セットの書き込みを中断した場合、システムが切り詰めを検知すると書き込み操作を完了させます。ディスクに 1 つ以上のレコードを書き込みますが、そのうちの少なくとも 1 つは切り詰められます。データ・セットはシステム上に残されます。

Tectia Client は一度の書き込み操作で大量のデータ (通常は 32kB) を書き込むことがあります。最後の操作で複数のレコードが書き込まれることがあり、そのうちのいくつかは切り詰められます。小さなファイルはファイルの末尾に書き込まれることがあるため、結果として得られるデータ・セットは、 RECORD_TRUNCATE=YES に設定して書き込まれたものと同じになります。

ファイル転送クライアント・プログラムのなかには、サーバからのエラー・メッセージや警告メッセージを必ずしも表示しないものがあります。詳細モード (--verbose、 -v) を使用すると、サーバからのメッセージがより多く表示されることがあります。

注意

Tectia Client が RECORD_TRUNCATE=YES の設定でデータ・セットを書き込むと、データ損失が発生することがあります。

[NO] TRUNCATE | [NO] TRU | [NO] TRUN

TRUNCATE | TRU | TRUN は RECORD_TRUNCATE=YES と同じです。

[NO] TRUNCATE | [NO] TRU | [NO] TRUN は RECORD_TRUNCATE=NO と同じです。

RETENTION_PERIOD | RET= days

保持期間　　(日数)。この保持期間が過ぎるとデータ・セットは失効し、オペレーティング・システムはデータ・セットを削除または上書きできるようになります。

デフォルト: システム・デフォルト

SECONDARY_SPACE | SE | SEC= space

データ・セットの二次領域割り当て。

デフォルト: none

SIZE | L= size

データ・セット割り当てのサイズ推定 (バイト単位)。

デフォルト: 1000000

SPACE_RELEASE | RLSE=YES | NO

SPACE_RELEASE は、新しいデータ・セットが割り当てられたときに、未使用のディスク領域を解放するかどうかを指定します。 YES に設定すると、新しいデータ・セットの未使用ディスク領域が解放されます。 NO に設定すると、新しいデータ・セットの割り当て済みディスク領域が保持されます。

デフォルト: YES

SPACE_UNIT | SU=BLKS | TRKS | CYLS | AVGRECLEN

データ・セットの領域割り当ての単位。

領域割り当て単位には以下の値を指定できます。

- BLKS: ブロック
- CYLS: シリンダ
- TRKS: トラック
- AVGRECLEN: 平均レコード長

デフォルト: none

SPACE_UNIT_LENGTH|SUL= length

SPACE_UNIT=BLKS または SPACE_UNIT=AVGRECLEN のときに、領域割り当て単位のサイズを指定します。

デフォルト: SPACE_UNIT=AVGRECLEN では 100、 SPACE_UNIT=BLKS では none

STAGING|S|STAGE=YES|NO

ファイルまたはデータ・セットにアクセスするときに、SFTP サーバでステージングを使用するかどうかを指定します。

NO に設定すると、ステージングは使用されません。

YES に設定すると、必要なときにステージングが使用されます。

デフォルト: NO

注意

ステージングを使用する場合、環境変数 _CEE_RUNOPTS の TRAP オプションを OFF に設定しないでください。そのようにすると **sftpg3** は起動に失敗します。 TRAP オプションはデフォルトで ON になっています。

STORCLAS|ST= class

システムが管理するストレージのストレージ・クラスを指定します。

デフォルト: none

SVC99_TEXT_UNITS|SVC99= string

他のファイル転送属性からの引数を上書きする、または引数に追加する動的割り当ての引数。

デフォルト: none

TRACKS|TR

領域割り当て単位がトラックであることを指定します。 SPACE_UNIT=TRKS と同じです。

TRAILING_BLANKS|TRAIL=YES|NO

転送されたデータ・セットの末尾の空白を保持するかどうかを指定します。

YES に設定すると、末尾の空白が転送されます。この設定は、たとえば固定フォーマットのデータ・セットを Unix タイプのファイル・システムに転送する際に、その構造を保持するために使用できます。

NO に設定すると、末尾の空白が取り除かれます。

デフォルト: NO

注意

このオプションは行区切りのターゲット・ファイル (TRANSFER_FORMAT=LINE) にのみ適用され、ユニットレコード・データ・セットには適用されません。

[NO]TRAILINGBLANKS|[NO]TRAI|[NO]TRAIL

TRAILINGBLANKS|TRAI|TRAIL は TRAILING_BLANKS=YES と同じです。
NOTRAILINGBLANKS|NOTRAI|NOTRAIL は TRAILING_BLANKS=NO と同じです。

TRANSFER_CODESET|C|CODESET= codeset

転送中のデータに指定されたコード・セットが設定されます。codeset は、変換を行うシステムの **iconv** 機能が認識しているコード・セット名です。使用可能なコード・セットは、USS プロンプトで -l オプションを使い **iconv** コマンドを呼び出すことで表示できます。

```
> iconv -l
```

デフォルト: none

例: Windows SFTP クライアントが z/OS のデータセットにファイルを送信し z/OS からデータセットを受信する

```
sftp> site C=ISO8859-1 D=IBM-1047 ❶
sftp> sput file.txt //DATASET.TXT ❷
sftp> sget //DATASET.TXT file.txt ❸
```

❶ z/OS サーバに対して、転送中のコード・セットは ISO8859-1 であり、データ・セットは IBM-1047 コード・セットでサーバに保存されることを伝えます。
❷ サーバはデータ受信時にコード・セットを ISO8859-1 から IBM-1047 に変換します。
❸ サーバはデータ送信前にコード・セットを IBM-1047 から ISO8859-1 に変換します。

注意

行区切り記号の情報は常に、変換を実行できるホスト、上記の場合は z/OS ホストに与えられます。

TRANSFER_FILE_CODESET|D|FCODESET= codeset

データ・セット内のデータに指定されたコード・セットが設定されます。codeset は、変換を行うシステムの **iconv** 機能が認識しているコード・セット名です。使用可能なコー

ド・セットは、USS プロンプトで `-l` オプションを使い **iconv** コマンドを呼び出すことで表示できます。

```
> iconv -l
```

デフォルト: none

`TRANSFER_FILE_LINE_DELIMITER|J|FLDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL`

転送ファイルの行区切り記号は、(転送元または転送先) ファイルで使用される改行規則を指定します。以下の値を指定できます。

- `UNIX`: ファイル内で使用される行区切り記号は LF (`\n`、 `0x0A`) です。
- `MVS`: ファイル内で使用される行区切り記号は NL (`\n`、 `0x15`) です。データ・セットに書き込む際、CR (`\r`、 `0x0D`) コードもエンド・オブ・ラインとみなされます。
- `MVS-FTP`: MVS のデータ・セットを読み込む際、データ・セットの各レコードが 1 行として扱われます。転送の行区切り記号はレコードに付加されます。レコード・データ内の制御文字はすべて保持されます。

  プリンタ制御文字を含むデータ・セットを読み込む際、制御文字は出力に保持されます。

  `TRANSFER_TRANSLATE_TABLE|E` によって、または `TRANSFER_CODESET|C` と `TRANSFER_FILE_CODESET|D` によってコード・セット変換が指定されている場合、付加される区切り記号は、`TRANSFER_LINE_DELIMITER|I`、 `TRANSFER_CODESET|C`、または `TRANSFER_TRANSLATE_TABLE|E` で指定されている区切り記号になります。コード・セット変換が要求されない場合、区切り記号はデータ・セットのコード・セットによって定義されます。デフォルトでは EBCDIC です。

  コード・セットは、 `TRANSFER_CODESET` を指定せずに `TRANSFER_FILE_CODESET` を定義することで指定できます。たとえば、Unicode の DOS 区切り記号 (`x'000D000A'`) をレコードに付加するには "`I=DOS,J=MVS-FTP,D=UCS-2`" と設定し、ISO Latin 1 の Unix 区切り記号 (`x'0A'`) を付加するには "`I=UNIX,J=MVS-FTP,D=ISO8859-1`" と設定します。

  この方法は、データ・セットを書き込むときには使用しないでください。
- `DOS`: ファイル内で使用される行区切り記号は CRLF (`\r\n`、 `0x0D 0x0A`) です。
- `MAC`: ファイル内で使用される行区切り記号は CR (`\r`、 `0x0D`) です。
- `NEL`: ファイル内で使用される行区切り記号は Unicode の改行 (`0x85`) です。

デフォルト: none

注意

行区切り記号の情報は、Tectia がインストールされたホストなど、変換を行うことができるホストに与える必要があります。

行区切り記号の変換は 1 バイトのコード・セットに対してのみ実行されます。

行区切り記号を変換するには、 TRANSFER_LINE_DELIMITER|I と TRANSFER_FILE_LINE_DELIMITER|J の両方が指定されている必要があります。

例: z/OS Tectia SFTP クライアントが Windows ホストにデータ・セットを送信し、Windows からファイルをコピーして戻す

この例ではコード・セットも変換されます。

```
sftp> lsite I=dos J=mvs                ❶
sftp> lsite C=IBM-437 D=IBM-1047       ❷
sftp> sput //DATASET.TXT file.txt      ❸
sftp> sget file.txt //DATASET.COPY.TXT ❹
```

❶ 転送の行区切り記号が DOS、転送ファイルの区切り記号が MVS に設定されています。
❷ 転送コード・セットが IBM-437、転送ファイル・コード・セットが IBM-1047 に設定されています。
❸ z/OS クライアントは、各レコードの後に NL (0x15) 文字を挿入します。行区切り記号の変換では、すべての NL:s が CRLF (0x0D 0x0A) 文字に変換され、コード・セット変換では変更されないまま残ります。
❹ CRLF の行区切り記号が LF 文字に変換され、コード・セット変換で NL 文字に変換されます。NL 文字 (データ中に CR 文字があればそれらも) ごとに現在のレコードが書き出され、新しいレコードが開始されます。

TRANSFER_FORMAT|F|FORMAT=LINE|STREAM|RECORD

バイト・ストリームは、SFTP プロトコル・パケットのペイロードとして転送されるバイトで構成されています。バイト・ストリームのフォーマットはLINE、 STREAM、または RECORD のいずれかです。3 つすべてのフォーマットで、テキスト・データ、非テキスト・データ、またはそれらを混ぜたデータを含めることができます。

MVS データ・セットを書き込む場合、 RECORD_TRUNCATE が YES に設定されていない限り、最大レコード長または固定レコード長より長いレコードはエラーの原因となり、その場合はレコードが切り詰められます。固定レコード長のデータ・セットに書き込む場合、レコード転送フォーマットを使用すると短いレコードはバイナリ・ゼロで埋められ、行転送フォーマットを使用すると短いレコードは空白で埋められます。

- LINE: 行転送フォーマットがレコードをベースとします。この場合、レコードの終わりを示すために区切り文字を使用します。区切り文字にはキャリッジ・リターン (CR) またはニューライン (NL) を使用できます。ASA 制御文字を含むデータ・セットに書き込んだり、そのようなデータ・セットから読み込んだりする場合、フォーム・フィード (FF) も区切り記号として扱われます。これらの文字の EBCDIC と アスキーの値は下表の通りです。行転送フォーマットで Tectia Client に送信するデータは、EBCDIC であるか、または転送中に EBCDIC に変換する必要があります。

Delimiter	EBCDIC				ASCII			
	Name	Dec	Oct	Hex	Name	Dec	Oct	Hex
\r Carriage Return	CR	13	015	0x0D	CR	13	015	0x0D

\n Newline	NL	21	025	0x15	LF	10	012	0x0A
\f Form Feed	FF	12	014	0x0C	FF	12	014	0x0C

アスキーには NL 文字がなく、代わりにライン・フィード (LF) を使って行を区切ります。

アスキーのライン・フィード (LF/10/012/0x0A) を EBCDIC のライン・フィード (LF/37/045/0x25) に変換したり、EBCDIC のニューライン (NL/21/025/0x15) をアスキーのネクスト・ライン (NEL/133/0205/0x85) に変換したりすることは避けてください。

Tectia Client に \r\n や \n\r のような二重の区切り記号を送ると、レコードが 2 つになるので注意が必要です。 TRANSFER_LINE_DELIMITER 及び TRANSFER_FILE_LINE_DELIMITER 属性を使用すると、Tectia Client のサーバまたはクライアント・プログラムに行区切り記号の規則を変換させることができます。

Tectia Client は、サーバ初期化 SFTP プロトコル・メッセージの「Server Newline Convention (サーバ改行規則)」として \n を送信します。

ASA ライン・プリンタ制御文字を含む MVS ファイルから行フォーマットのデータを転送する場合、IBM z/OS の『XL C/C++ Programming Guide』、『Using ASA Text Files』の章で説明されているように、Tectia Client は制御文字と行区切り文字を変換します。

ASA コードを変更せずにレコードを転送するには、 STREAM または RECORD の転送フォーマットを使用するか、または DD カードを使用してデータ・セットを定義し、 RECFM=FB または RECFM=VB を指定します。

行転送フォーマットで転送されたデータ・セットと、メインフレームで再作成されたデータ・セットは、必ずしも同一ではありません。

- STREAM: ストリーム転送フォーマットにデータ・セットのデータ・バイトは含まれますが、構造情報は含まれません。固定レコード長のデータ・セットをストリーム・フォーマットで転送し、同じレコード長で再作成すると、レコード構造は保持されます。可変長レコードをストリーム・フォーマットで転送すると、正しく再作成されません。

- RECORD: レコード転送フォーマットがレコードをベースとします。各レコードの前には、4 バイトのビッグエンディアン・バイナリ整数からなる長さフィールドがあり、これはレコード内のデータ・バイト数を示します。 RECFM=V または RECFM=VB のデータ・セットにおけるレコード記述語とフォーマットが異なることに注意してください。

  レコード転送フォーマットで転送されたデータ・セットは、任意のデータ・セット・タイプで再作成できます。

デフォルト: LINE

TRANSFER_LINE_DELIMITER|I|LDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL

転送の行区切り記号は、接続を介して転送されるデータで使用される改行規則を指定します。以下の値を指定できます。

- UNIX: 接続の行区切り記号は LF (\n、 0x0A) です。
- MVS: 接続の行区切り記号は NL (\n、 0x15) です。データを EBCDIC からアスキーに変換した場合、NL は LF (\n、 0x0A) になります。
- MVS-FTP: データ・セットに書き込む際、LF (\n、 0x0A) 制御コードのみがエンド・オブ・ラインとみなされます。CR (\r、 0x0D) コードはレコードのデータとして保持されます。

  ASA プリンタ制御文字を含むデータ・セットを書き込む場合、各行の最初の文字が ASA 文字として使用されます。

  この方法は、データ・セットを読み込むときには使用しないでください。
- DOS: 接続の行区切り記号は CRLF (\r\n、 0x0D 0x0A) です。
- MAC: 接続の行区切り記号は CR (\r、 0x0D) です。
- NEL: ファイル内で使用される行区切り記号は Unicode の改行 (0x85) です。

デフォルト: none

注意

行区切り記号の情報は、Tectia がインストールされたホストなど、変換を行うことができるホストに与える必要があります。

TRANSFER_MODE|X|MODE=BIN|TEXT

転送モードは、コード・セットと行区切り記号の変換が行われるかどうかを指定します。使用できる値は以下の通りです。

- BIN: コード・セットと行区切り記号の変換は行われません。
- TEXT: コード・セットと行区切り記号の変換が行われます。

デフォルト: none

注意

TRANSFER_MODE が指定されていない場合に TRANSFER_CODESET と TRANSFER_FILE_CODESET または TRANSFER_TRANSLATE_TABLE の両方が存在すると、変換が行われます。

TRANSFER_TRANSLATE_DSN_TEMPLATES|A|XDSNT= templates

templates は変換テーブルの検索テンプレートを指定します。変換テーブル名 (前述を参照) を挿入する箇所を示すには、'%T' を記述します。テンプレートはプラス文字で区切ります。データ・セット名のテンプレートにはスラッシュを含めてはならず、その代わりに 2 つまたは 3 つのアンダースコアを前に付ける必要があります。

最初に見つかった変換テーブル・データ・セットがコード変換に使用されます。

注意

変換テーブルは行区切り記号を EBCDIC の NL 文字に変換する必要があります。TRANSFER_FORMAT を参照してください。

デフォルト: none

TRANSFER_TRANSLATE_TABLE|E|XTBL= table

TABLE は、コード・セット変換を指定するテーブルの名前です。この属性を設定すると、転送コード・セット及びファイル・コード・セットの属性が上書きされます。このテーブルは常に通常の方向で適用されます。つまり、最初の文字配列は (行からデータ・セットへの) 入力データに、2 番目の配列は出力データに使用されます。データ・セットにアスキーが含まれていて、EBCDIC として転送する必要がある場合など、逆の変換が必要な場合は、(システム・ユーティリティ CONVXLAT を使用したり、既存の変換データ・セットを編集したりするなどして) 文字配列を逆順にしたテーブル・データ・セットを作成できます (またはシステム・プログラマに対応を依頼してください)。

TYPE|T=PS|PO|PDS|POE|PDSE|GDG|HFS|VSAM|ESDS|KSDS|RRN

データ・セットが作成されるときのデータ・セットの種類を指定します。使用できる値は以下の通りです。

- PS: 作成されるデータ・セットのタイプは PS です。
- PO|PDS: 作成されるデータ・セットのタイプを PDS にします。PDS を作成するためには、DIRECTORY_SIZE パラメータを指定する必要があります。ディレクトリ・サイズを指定しない場合、パーティション化されたデータ・セットではなく、シーケンシャルなデータ・セットが作成されます。
- POE|PDSE: 作成されるデータ・セットのタイプは PDSE です。
- GDG: 作成されるデータ・セットのタイプは GDG です。
- HFS: 作成されるデータ・セットのタイプは HFS です。
- VSAM: 作成されるデータ・セットのタイプは VSAM です。
- ESDS: 作成されるデータ・セットのタイプは VSAM ESDS です。
- KSDS: 作成されるデータ・セットのタイプは VSAM KSDS です。
- RRN: 作成されるデータ・セットのタイプは VSAM RRN です。

デフォルト: データ・セット名にメンバーが含まれる場合は PO。それ以外の場合は PS

UNIT|UN= unit

データ・セットが属する予定の (すでに存在している場合は属している) デバイスまたはデバイス・グループの名前。unit の最大長は 8 文字です。値が最大長を超える場合は、8 文字に切り詰められます。

デバイスのアドレスを指定することもできます。その場合は、4 桁のアドレスの前にアンダースコアを付けます。

デフォルト: none

UNIT_COUNT|UC|UNC= number

データ・セットのデバイス数を指定します。この属性は、JCL の DD ステートメントの UNIT パラメータの 2 番目の値に対応します。

デフォルト: システム・デフォルト

UNIT_PARALLEL|UNP=YES|NO

データ・セットの全ボリュームを並列にマウントするようにシステムに要求します。この属性は、JCL の DD ステートメントの UNIT パラメータの 2 番目の値にある 'P' の文字に対応します。

デフォルト: システム・デフォルト

VOLUME_COUNT|VC|VOLCNT= number

出力データ・セットが必要とするボリュームの最大数を指定します。この属性は、JCL の DD ステートメントの VOLUME パラメータのボリューム・カウントの値に対応します。

デフォルト: システム・デフォルト

VOLUMES|VO|VOL= vol1+vol2+...

データ・セットが属する予定の (すでに存在している場合は属している) ボリュームをプラス記号 (+) で区切ったリストです。

デフォルト: none

第6章 Secure Shell トンネリング

トンネリングとは、Secure Shell を経由して、そのままではセキュアではないアプリケーション・トラフィックを転送する方法です。トンネリングを行うことで、そのままではセキュアではない POP3 や SMTP、HTTP ベースのアプリケーションにセキュアに接続できます。

Secure Shell v2 接続プロトコルには、幅広い用途に使用できるチャネルがあります。そのようなチャネルがすべて、1 つの暗号化されたトンネルで多重化されており、任意の TCP/IP ポートや X11 接続のトンネリング (転送) に使用できます。

トンネルを使用するクライアント/サーバ・アプリケーションは、暗号化トンネルを使用しない場合と同様に、自身の認証手順が存在する場合はそれを実行します。

プロトコルやアプリケーションは、固定のポート番号にしか接続できない場合があります (例: IMAP 143)。それ以外の場合は、使用可能なポートであればトンネリングに選択できます。リモート・トンネルの場合、1024 未満のポート (ウェルノウン・サービス・ポート) は一般ユーザでは使用できず、システム管理者 (root 権限) のみが使用できます。

トンネルにはローカルとリモートの 2 つの基本的な種類があります。それぞれは送信トンネル、受信トンネルとも呼ばれています。X11 転送とエージェント転送はリモート・トンネルの特殊なケースです。トンネリングのさまざまなオプションについては後述の項で取り扱います。

6.1. ローカル・トンネル

harutoyo@fujitsu.com

ローカル (送信) トンネルは、ローカル・ポートで受信したトラフィックを指定されたリモート・ポートに転送します。

コマンドラインで `sshg3` を使用する場合、ローカル・トンネリング・コマンドの構文は以下のようになります。

```
client$ sshg3 -L [protocol/][listen-address:]listen-port:dst-host:dst-port sshserver
```

ここで、

- `[protocol/]` は、トンネルされた接続で使用されるプロトコルを指定します。これは `ftp` または `tcp` です (オプション引数)。デフォルトは `tcp` です。

- `[listen-address:]` は、ローカル・クライアントのどのインターフェイスでリッスンするかを定義します (オプション引数)。 省略された場合は、クライアント側の全てのインターフェースをバインドするために `-g --gateway` オプションが `-L` の前で使用されていない場合を除き、ローカル・インターフェースのみでリッスンされます

- `listen-port` はローカル・クライアントのポート番号で、このポートで受信した接続はサーバにトンネルされます。

- `dst-host:dst-port` は、サーバから接続がトンネルされる接続先ホスト・アドレスとポートを定義します。

- `sshserver` は Secure Shell サーバの IP アドレスまたはホスト名です。

接続先ホスト及び sshserver のホスト名や IP アドレスを、egrep 構文に従った正規表現で定義できますが、ワイルドカードはサポートされていません。

注意

`dst-host` に IP アドレスではなくドメイン名を指定した場合、 `sshserver` のアドレス・ファミリの設定に従って名前が解決されます。たとえば、ドメイン名が AAAA DNS レコード (IPv6) に解決され、サーバのアドレス・ファミリの設定が `inet` (IPV4) の場合、トンネルは機能しません。

ローカル・トンネリングをセットアップすると、ローカル・クライアント・ホスト上のリスナ・ポートが割り当てられます。このリスナへの接続が行われるたびに、接続は Secure Shell を介してリモート・サーバにトンネルされ、指定された接続先ホスト及びポートにサーバからもう一つの接続が行われます。サーバからのその先の接続はセキュアではなく、通常の TCP 接続になります。

注意

ローカル・クライアント・ホストにアクセスできるユーザは誰でもローカル・トンネルを使用できるようになります。

図 6.1 は、ローカル・トンネリング (ポート転送) に関与するさまざまなホストとポートを示しています。

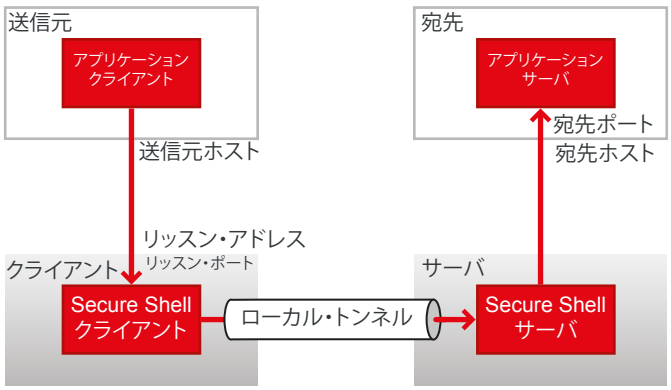


図6.1 ローカル・トンネリングの用語

たとえば、コマンドラインで以下の `sshg3` コマンドを実行すると、クライアント・ホストのポート番号 1234 で受信したトラフィックはすべて、サーバのポート番号 23 に転送されます。

```
client$ sshg3 -L 1234:localhost:23 --abort-on-failing-tunnel username@sshserver
```

コマンドの転送先アドレスはトンネルの (リモート) エンド・ポイントで解決されます。この場合、`localhost` はサーバ・ホスト (`sshserver`) を指します。

この例では `--abort-on-failing-tunnel` オプションも指定されています。これによって、トンネル・リスナの作成に失敗した場合 (ポートがすでに予約されている場合など) にコマンドが中断されます。通常、サーバへの接続は成功したが、リスナの作成に失敗した場合、エラー・メッセージは表示されません。

6.1.1. 非透過的 TCP トンネリング

非透過的 TCP トンネリングを使用する場合、トンネルされるアプリケーションはサーバに直接接続するのではなく、ローカルのリスナ・ポートに接続するように設定されます。Tectia Client はリモート・サーバに接続をセキュアに転送します。

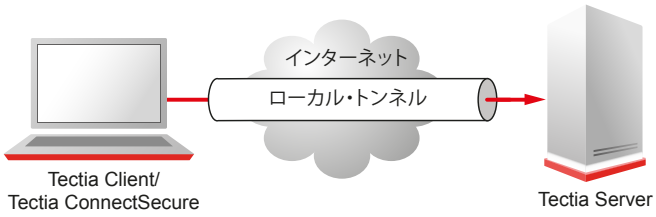


図6.2 シンプルなローカル・トンネル

たとえば、`sshclient`、`sshserver`、及び `imapserver` の 3 つのホストがあり、`sshclient` のポート番号 `143` で受信するトラフィックを `imapserver` のポート番号 `143` に転送する場合、`sshclient`

と sshserver の間の接続だけがセキュアになります。使用するコマンドは、以下のようなものになります。

```
sshclient$ sshg3 -L 143:imapserver:143 username@sshserver
```

図 6.3 は、Secure Shell サーバが DMZ ネットワークに存在する場合の例です。この場合の接続は、Secure Shell クライアントから Secure Shell サーバまでは暗号化されますが、その後、企業ネットワーク内では IMAP サーバまで暗号化されません。

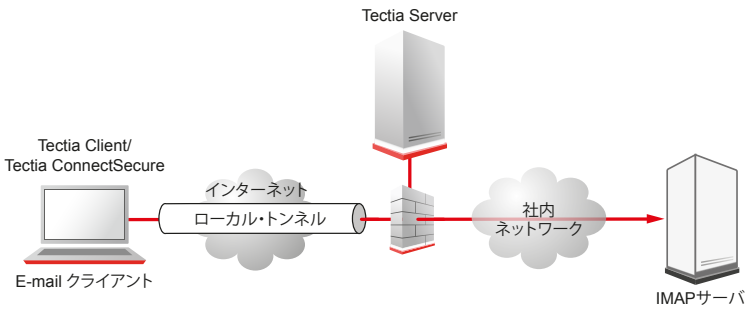


図6.3 IMAP サーバまでのローカル・トンネル

トンネルは、接続ブローカーの設定ファイルで接続プロファイルに定義することもできます。定義されたトンネルは、プロファイルとの接続が行われたときに自動的に開かれます。以下は、ssh-broker-config.xml ファイルの例です。

```
<profile id="id1" host="sshserver.example.com">
...
  <tunnels>
    <local-tunnel type="tcp"
                  listen-port="143"
                  dst-host="imap.example.com"
                  dst-port="143"
                  allow-relay="no" />
  ...
  </tunnels>
</profile>
```

デフォルトでは、クライアント・ホスト自身から送信されるローカル・トンネルのみが許可されます。トンネル・リスナ・ポートへの他のマシンからの接続を許可するには、allow-relay を yes に設定します。

Tectia コネクション設定 GUI でのトンネリングの設定は、プロファイルごとに [**接続プロファイル**] → [**トンネル**] で行います。「トンネリングの定義」 を参照してください。

自動トンネル

自動トンネルは、アプリケーション接続用の非透過的ローカル・トンネルを作成する方法の 1 つです。

自動トンネルは、トンネルの確立に常に接続プロファイルを使用します。接続ブローカーの起動時に自動的に有効になるローカル・トンネルのリスナを作成できます。実際のトンネル

は、リスナ・ポートに初めて接続されたときに形成されます。その時点でサーバへの接続が開かれていない場合は、その接続も自動的に開かれます。

接続ブローカーの設定ファイルでは、以下のような設定を行います。

```
<static-tunnels>
  <tunnel type="tcp"
          listen-port="9874"
          dst-host="st.example.com"
          dst-port="9111"
          allow-relay = "no"
          profile="id1" />
</static-tunnels>
```

自動トンネルは Tectia コネクション設定　GUI の [**自動トンネル**] ページで設定できます。手順については、A.1.6 を参照してください。

ローカル・トンネリングの例

ローカル・ポート転送を使用してセキュアなトンネルを作成するために **sshg3** を使用する場合、トンネルされる TCP アプリケーションは、アプリケーション・サーバ・ポートではなく localhost ポートに接続するように設定します。

例として挙げるアプリケーションの `clientapp1` は、デフォルトで TCP ポート番号 2345 を使用して Unix サーバ `unix.example.com` に接続します。

```
$ clientapp1 --username user1 --server unix.example.com --port 2345
```

この TCP アプリケーションを Secure Shell でセキュアにするためには、以下のコマンドを使用します。

```
$ sshg3 -L 2345:localhost:2345 user1@unix.example.com -S -f &
$ clientapp1 --username user1 --server localhost --port 2345
```

上記の **sshg3** コマンドはリモート Secure Shell サーバ `unix.example.com` に接続し、ポート番号 2345 でローカル・リスナを作成し、受信トラフィックを `localhost:2345` に転送するようリモート Secure Shell サーバに指示し、シングルショットモードでバックグラウンドに移行します。

6.1.2. 非透過的 FTP トンネリング

非透過的 FTP トンネリングは一般的なトンネリング機構を拡張したものです。一般的なトンネリング (ポート転送) 機構とは異なり、非透過的 FTP トンネリングは、FTP 制御チャネルに加えて、転送されたファイルもセキュアにします。FTP　トンネリング・コードは、トンネルされた FTP 制御チャネルをモニタし、要求に応じてデータ・チャネルのための新しいトンネルを動的に作成します。

非透過的 FTP トンネリングを使用すると、ローカル・クライアントのポートからリモート・サーバへのトンネルが作成されます。FTP クライアントは Tectia Client に接続するように設定されており、そこから、Secure Shell サーバが動作しているエンド・ポイントに接続が転送されます。

標準的な使用事例は、Tectia Client が FTP クライアントと同じホストにあり、FTP サーバが Secure Shell サーバと同じホストにある場合です。これ以外の設定もサポートされていますが、Tectia Client と Secure Shell サーバ間の接続だけが暗号化される点に注意が必要です。

非透過的 FTP トンネリングは、コマンドラインで要求することも、接続ブローカーの設定で有効にして定義することもできます。設定された非透過的 FTP トンネリングは、Tectia Client で定義されている接続プロファイルを使用します。

コマンドラインでは、FTP トンネリングはローカル・トンネルとリモート・トンネルの両方に使用されます。非透過的 FTP トンネリングは、以下の構文の `sshg3` コマンドを入力することで開始されます。

```
sshclient$ sshg3 -L ftp/1234:localhost:21 username@sshserver
```

`sshg3` コマンドの詳細については、sshg3(1) の man ページを参照してください。

FTP トンネリングの設定は Tectia コネクション設定 GUI で各プロファイルに対して [**接続プロファイル**] → [**トンネル**] で行えます。「トンネリングの定義」 を参照してください。

FTP トンネルは、接続ブローカーの設定ファイルで接続プロファイルに対して定義することもできます。以下は、接続ブローカーの設定ファイル `ssh-broker-config.xml` の例です。

```
<profiles>
  <profile id="id1" host="sshserver.example.com"
  ...
    <tunnels>
      <local-tunnel type="FTP"
                    listen-port="1234"
                    dst-host="127.0.0.1"
                    dst-port="21"
                    allow-relay="NO" />
      ...
    </tunnels>
  </profile>
</profiles>
```

これで、以下の (例の) コマンドで FTP 接続が行えるようになります。

```
sshclient$ ftp
ftp$ open localhost 1234
```

これで、クライアントのポート番号 1234 への FTP 接続が、Secure Shell サーバのポート番号 21 へトンネルされるようになりました。

FTP トンネリングの代替策として、セキュアなファイル転送のために **sftpg3** または **scpg3** クライアントが使用できます。これらのクライアントはコマンドラインやスクリプトで使用できます。また、Tectia Server にはサブシステムとして **sft-server-g3** がすでにあり、 **sftpg3** 及び **scpg3** クライアントは Tectia Client に含まれているので、FTP トンネリングよりも必要な設定が少なくなります。サーバ・マシンでのリモート・ユーザ制限の管理は、FTP でも行う必要がないため、より簡単になります。

FTP トンネリングがどのように行われるかを正確に理解するためには、FTP プロトコルのアクティブ・モードとパッシブ・モードという 2 つの異なる事例を調べる必要があります。

パッシブ・モードでの FTP のトンネル

パッシブ・モードでは、FTP クライアントが `PASV` コマンドをサーバに送信すると、サーバはデータ・チャネルのリスナ・ポートを開き、リスナの IP アドレスとポート番号をクライアントへの応答として送信することで応答します。応答の形式は以下の通りです。

```
227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)
```

ここで、 `a1.a2.a3.a4` は IP アドレス、`p1*256+p2` はポート番号です。たとえば、IP アドレス 10.1.60.99、ポート番号 1548 に対する応答は`227 Entering Passive Mode (10,1,60,99,6,12)` です。

`PASV` コマンドの応答を受けた接続ブローカーは、応答に記載されている転送先へのローカル・ポート転送を作成します。その後、接続ブローカーは応答の IP アドレスとポート番号を、新しく作成したローカル・ポート転送のリスナ (常に localhost アドレスの 127.0.0.1 に存在する) を指すように書き換え、FTP クライアントにその応答を渡します。FTP クライアントは応答に基づいてデータ・チャネルを開き、Secure Shell 接続を介して、FTP サーバが開いたリスナにデータを効果的にトンネルします。この結果として、Secure Shell サーバから FTP サーバまで (それらが別のマシンにある場合) を除くデータ・チャネル全体が保護されます。この一連の流れは、すべてのデータ・チャネルに対して自動的に行われます。

トンネルは localhost アドレスに対して開かれるため、パッシブ・モードを使用する場合、FTP クライアントは Tectia Client と同じマシン上で実行される必要があります。

アクティブ・モードでの FTP のトンネリング

アクティブ・モードでは、FTP クライアントは FTP サーバから FTP クライアントへのデータ・チャネルのためにローカル・ポート上にリスナを作成し、IP アドレスとポート番号を以下の形式のコマンドで FTP サーバに送信してチャネルを要求します。

```
PORT a1,a2,a3,a4,p1,p2
```

ここで、`a1.a2.a3.a4` は IP アドレス、`p1*256+p2` はポート番号です。接続ブローカーはこのコマンドをインターセプトし、Secure Shell サーバの localhost アドレスから、`PORT` コマンドで指定されたアドレス及びポートへのリモート・ポート転送を作成します。

トンネルを作成した後、接続ブローカーは `PORT` コマンドで指定されたアドレスとポートを、Secure Shell サーバで新しく開いたリモート転送を指すように書き換え、その内容を FTP サーバに送信します。これで FTP サーバは、`PORT` コマンドで指定されたアドレスとポートにデータ・チャネルを開き、Secure Shell 接続を介してデータを効率的に転送するようになります。接続ブローカーは受信したデータを、FTP クライアントが作成したオリジナルのリスナに渡します。この結果として、Tectia Client から FTP クライアントまでを除くデータ・チャネル全体がセキュアになります。この一連の流れは、すべてのデータ・チャネルに対して自動的に行われます。

アクティブ・モードでの FTP トンネリングが機能するためには、FTP サーバが Secure Shell サーバと同じホストで実行されており、FTP クライアントと Tectia Client が同じホストに存在している必要があります。

注意

アクティブ・モードでの FTP トンネリングは、らゆる構成で動作すると保証されているわけではありません。可能であれば、FTP 接続をトンネルするときはパッシブ・モードを使用してください。

6.1.3. SOCKS トンネリング

SOCKS トンネリングは、SOCKS4 または SOCKS5 クライアント・プロトコルをサポートするアプリケーションのトンネリングに使用できる機構です。

ローカル・ホストの特定のポートからリモート・サーバの特定のポートへのトンネリング (別名ポート転送) を設定する代わりに、ユーザのアプリケーションで使用できる SOCKS サーバを指定できます。各アプリケーションは、localhost ポートで SOCKS サーバを使用するように設定されていることを除き、通常の方法で設定されています。Secure Shell クライアント・アプリケーションの Tectia Client は localhost のポートを開き、あらゆる SOCKS クライアント・アプリケーションに対して SOCKS4 及び SOCKS5 サーバを模倣します。

アプリケーションが IMAP4 や POP3、SMTP、HTTP などのサービスに接続する際、SOCKS サーバに必要な情報が提供されますが、実際には、SOCKS サーバを模倣した Tectia Client に提供されることになります。Tectia Client はこの情報を使用して Secure Shell サーバへのトンネルを作成し、トラフィックの双方向の中継をセキュアに行います。

コマンドラインで `sshg3` を使用する場合、SOCKS トンネリング・コマンドの構文は以下のようになります。

```
client$ sshg3 -L socks/[listen-address:]listen-port username@sshserver
```

ここで、

- `[listen-address:]` は、クライアント上のどのインターフェイスでリッスンするのかを定義します (オプションの引数)。
- `listen-port` はクライアント上のポート番号です。
- `sshserver` は Secure Shell サーバの IP アドレスまたはホスト名です。

たとえば以下のコマンドは、クライアントのポート番号 `1234` から `sshserver` へのローカル・トンネルをセットアップします。アプリケーションは、クライアントのポート番号 `1234` で SOCKS サーバを使用するように設定されています。サーバからの接続は保護されることなく、アプリケーションから要求された接続先ホストに転送されます。

```
sshclient$ sshg3 -L socks/1234 username@sshserver
```

SOCKS トンネルは、接続ブローカーの設定ファイルで接続プロファイルに定義することもできます。以下は、`ssh-broker-config.xml` ファイルの例です。

```
<profile id="id1" host="sshserver.example.com">
...
  <tunnels>
```

```
    <local-tunnel type="socks"
                  listen-port="1234"
                  allow-relay="no" />
  ...
  </tunnels>
</profile>
```

6.2. リモート・トンネル

リモート (受信) トンネルは、リモート・ポートで受信したトラフィックを指定されたローカル・ポートに転送します。

コマンドラインで `sshg3` を使用する場合、リモート・トンネリング・コマンドの構文は以下のようになります。

```
client$ sshg3 -R [protocol/][listen-address:]listen-port:dst-host:dst-port \
username@sshserver
```

ここで、

- `[protocol/]` は、トンネルされた接続で使用されるプロトコルを指定します。これは `ftp` または `tcp` です (オプション引数)。デフォルトは `tcp` です。
- `[listen-address:]` は、リモート・サーバのどのインターフェイスでリッスンするかを定義します (オプション引数)。 省略された場合は、サーバ側の全てのインターフェースをバインドするために `-g --gateway` オプションが `-L` の前で使用されていない場合を除き、ローカル・インターフェースのみでリッスンされます
- `listen-port` はリモート・サーバのポート番号で、このポートで受信した接続はクライアントにトンネルされます。
- `dst-host:dst-port` は、クライアントからの接続がトンネルされる接続先ホスト・アドレスとポートを定義します。
- `sshserver` は Secure Shell サーバの IP アドレスまたはホスト名です。

接続先ホストと sshserver の IP アドレス及びホスト名は、egrep 構文に従った正規表現を使って定義できます。ワイルドカードはサポートされていません。

注意

`dst-host` に IP アドレスではなくドメイン名を指定した場合、 `client` のアドレス・ファミリの設定に従って名前が解決されます。たとえば、ドメイン名が AAAA DNS レコード (IPv6) に解決され、クライアントのアドレス・ファミリの設定が `inet` (IPV4) の場合、トンネルは機能しません。

リモート・トンネリングをセットアップすると、リモート・サーバ上のリスナ・ポートが割り当てられます。このリスナへの接続が行われるたびに、接続は Secure Shell を介してローカル・クライアントにトンネルされ、指定された接続先ホスト及びポートにクライアントか

らもう一つの接続が行われます。クライアントからのその先の接続はセキュアではなく、通常の TCP 接続になります。

注意

リモート・サーバ・ホストにアクセスできるユーザは誰でもリモート・トンネルを使用できるようになります。

図 6.4 は、リモート・ポート転送に関与するさまざまなホストとポートを示しています。

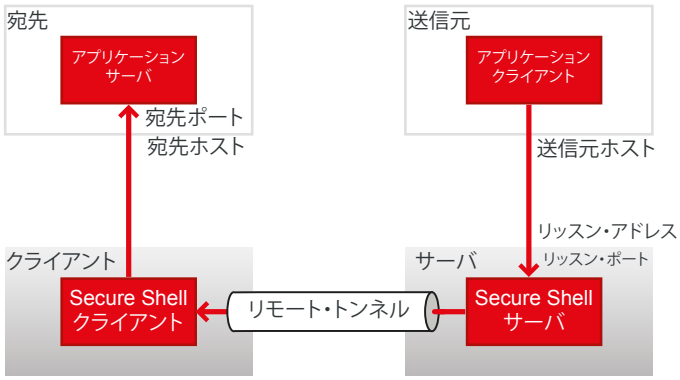


図6.4 リモート・トンネリングの用語

たとえば以下のコマンドを実行すると、サーバのポート番号 1234 で受信するトラフィックはすべて、クライアントのポート番号 23 にトンネルされます。図 6.5 を参照してください。

```
sshclient$ sshg3 -R 1234:localhost:23 username@sshserver
```

コマンドの転送先アドレスはトンネルの (ローカル) エンド・ポイントで解決されます。この場合、`localhost` はクライアント・ホストを指します。

図6.5 リモート・トンネル

トンネルは、接続ブローカーの設定ファイルで接続プロファイルにて定義することもできます。定義されたトンネルは、プロファイルとの接続が行われたときに自動的に開かれます。

以下は、`ssh-broker-config.xml` ファイルの例です。

```
<profile id="id1" host="sshserver.example.com">
...
  <tunnels>
    <remote-tunnel type="tcp"
```

```
                    listen-port="1234"
                    dst-host="localhost"
                    dst-port="23" />
   ...
   </tunnels>
</profile>
```

Tectia コネクション設定 GUI でのトンネリングの設定は、プロファイルごとに [**接続プロファイル**] → [**トンネル**] で行います。「トンネリングの定義」 を参照してください。

6.3. X11 転送

X11 転送はリモート・トンネリングの特殊なケースです。

Tectia Client は Unix と Windows の両方のプラットフォームで X11 転送をサポートしています。Windows では、XWindow Manager のパッケージも必要です。Tectia Server は Unix プラットフォームでのみ X11 転送をサポートしています。

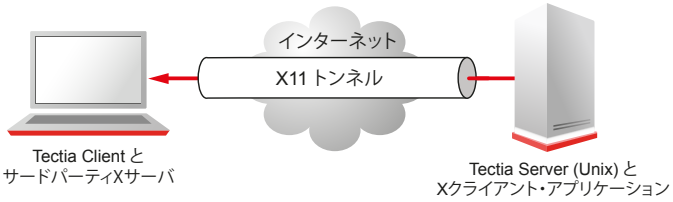


図6.6 X11 転送

X11 転送は、`ssh-broker-config.xml` ファイル (`default-settings` または接続 `profile` の下) で以下の行を設定することにより、クライアントで有効にできます。

```
<forwards>
  <forward type="X11" state="on"/>
</forwards>
```

デフォルトでは、X11 転送はオフになっています。

Tectia コネクション設定 GUI では、デフォルト接続については [**デフォルト接続**] → [**トンネル**] で、プロファイルごとの場合は [**接続プロファイル**] → [**トンネル**] で X11 転送を有効にできます。「デフォルトのトンネリング設定の定義」 及び 「トンネリングの定義」 を参照してください。

Windows で X11 転送が動作することをテストするには、XWindow Manager を使用します。リモート・システムにログインし、「`xclock &`」と入力します。これによって、転送接続のテストに使用できる X クロック・プログラムが起動します。

X クロック・ウィンドウが正常に表示されれば、X11 転送は機能しています。X クロックが失敗し、ディスプレイを開けないというエラーが出る場合は、XAuth がリモート・ホストに正しくインストールされているかどうか確認してください。

注意

クライアントで `DISPLAY` 変数は設定しないでください。ほとんどの場合、暗号化が無効になってしまいます。(Secure Shell でトンネルされた X 接続は、特別なローカル・ディスプレイ設定を使用します。)

6.4. エージェント転送

エージェント転送はリモート・トンネリングの特殊なケースです。エージェント転送では、ユーザが個別に認証することなく、Secure Shell の接続と公開鍵認証の情報が 1 つのサーバから別のサーバに転送されます。認証データはローカル・マシン以外のマシンに保存する必要がなく、認証パスフレーズや秘密鍵がネットワーク上を移動することは絶対にありません。

Tectia Client は認証エージェントの機能を提供し、接続ブローカーは認証エージェントとしての機能を OpenSSH クライアントにも提供します。Tectia Server は Unix プラットフォームでエージェント転送をサポートしています。したがって、エージェント転送チェーンのスタート・ポイントとエンド・ポイントは Windows ホストでも Unix ホストでもかまいませんが、転送チェーンの途中にあるすべてのホストは Unix ホストでなければならず、Secure Shell クライアントとサーバ・コンポーネントの両方がインストールされている必要があります。

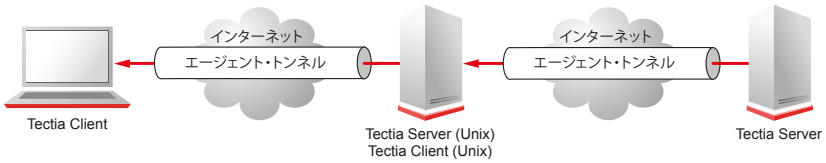


図6.7 エージェント転送

工場出荷時の設定では、エージェント転送は有効 (オン) になっています。

エージェント転送の有効と無効は、デフォルトの設定においてでも、接続プロファイルごとに個別にでもクライアント側で切り替えることができます。

エージェント転送を無効にするには、`ssh-broker-config.xml` ファイルの `default-settings` または接続 `profile` で以下の行を設定します。

```
<forwards>
  <forward type="agent" state="off" />
</forwards>
```

Tectia コネクション設定 GUI では、デフォルト接続については [**デフォルト接続**] → [**トンネル**] で、プロファイルごとの場合は [**接続プロファイル**] → [**トンネル**] でエージェント転送を無効にできます。「デフォルトのトンネリング設定の定義」 及び 「トンネリングの定義」 を参照してください。

第7章 Tectia Clientのトラブルシューティング

接続や認証、設定の問題が発生した場合、Tectia Client をデバッグ・モードで実行することで解決を試みることができます。デバッグの詳細については、7.3 の手順を参照してください。

トラブルシューティングに必要な情報を収集し、SSH のテクニカル・サポートに送信することもできます。Tectia のオンライン・サポート・リソースへのアクセス、及び SSH テクニカル・サポートへのお問い合わせについては、1.2 を参照してください。

SSH のテクニカル・サポートには、以下の情報を提供してください。

- Tectia Client の詳細レベルの出力。7.1 を参照してください。
- Tectia トラブルシューティング・ツール の出力から得たシステム情報。この情報は、報告された問題の分析に役立ちます。テクニカル・サポートはこの情報に基づいて、Tectia 製品が動作している環境の詳細を正確に把握できます。7.2 を参照してください。
- 可能な場合は、デバッグの情報。7.3 を参照してください。

7.1. 基本的なトラブルシューティング情報の収集

接続に関するほとんどの問題は、 **sshg3** を詳細モードで実行し、出力を調べることで解決できます。

コマンド **-v** (または **--verbose**) を入力すると、診断出力が表示されます。

```
$ sshg3 -v user@server.example.com
```

その後で、古い接続試行から診断出力を取得することもできます。以下のコマンドを実行すると、古い接続試行とその接続 ID が一覧表示されます。

```
$ ssh-broker-ctl list-connections --disconnected
```

以下のコマンドを実行すると、古い接続試行から取得した診断出力が表示されます。

```
$ ssh-broker-ctl connection-status <connection-id>
```

Windows では、接続に失敗した場合のエラー・ダイアログ・メッセージが Tectia - SSH ターミナルに用意されています。また、Tectia - SSH ターミナルを開いて [**ヘルプ**] → [**トラブルシューティング...**] を選択することで、Tectia Client で表示される接続情報と過去 5 回のメッセージを取得できます。その場合は、図 7.1 のようなウィンドウが開きます。

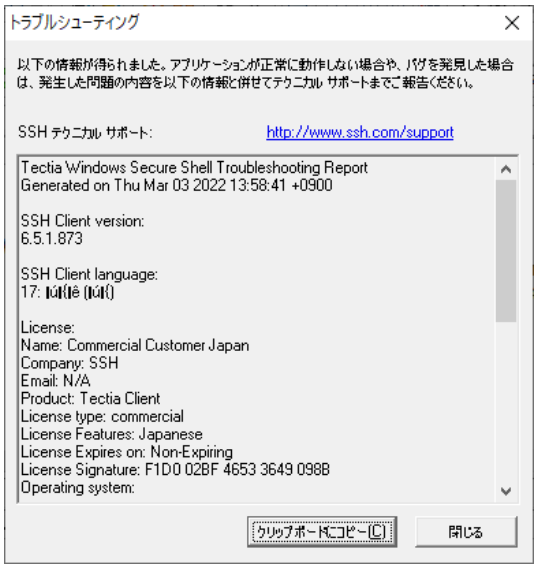


図7.1 トラブルシューティング・データを含むウィンドウ

7.2. トラブルシューティングのためのシステム情報の収集

Tectia Client には、オペレーティング・システムやハードウェア、インストールされている Tectia 製品のバージョンやその設定について、必要なデータを自動的にファイルに収集するトラブルシューティング・ツールが含まれています。このトラブルシューティング・ツールはシステム構成に関する以下の情報を収集します。

- オペレーティング・システム (OS) のバージョンとインストールされているパッチ
- OS の設定ファイルやその他の OS 情報 (PAM、syslog、resolver、ifconfig など)
- ハードウェア情報 (マシン・モデル、セキュリティ・クラス、CPU バージョンなど)
- OS のステータス (予約済みポートやソケットごとの接続数など)
- Tectia のバイナリ。ツールは実際のインストール・パッケージのバージョンおよびデバッグ・パッケージもチェックし検出します
- Tectia のグローバル設定。Unix では `/etc/` 及び `/opt/` ディレクトリから、Windows の場合は以下のデフォルト・インストール・ディレクトリから
  - "`C:\Program Files (x86)\SSH Communications Security\SSH Tectia`" (64 ビット Windows バージョンの場合)

- ユーザのホーム・ディレクトリにあるユーザ固有の Tectia の設定: `$HOME/.ssh2` (Unix の場合)、及び `"C:\Documents and Settings\<username>"` または `"C:\Users\"` (Windows の場合)
- トラブルシューティング・ツールを実行しているユーザ・アカウント
- Unix では、秘密鍵を含め、指定されたユーザの設定ディレクトリに保存されているすべての情報を収集するかどうかを設定できます。この情報を参考にして、テクニカル・サポートはユーザの状況をよりよく再現できます。

システム情報を収集するには、コマンド・プロンプトを開き、以下のコマンドを入力します。

Unix では、トラブルシューティング・ツールを以下のコマンドで実行します。

```
# ssh-troubleshoot [options] info [command-options]
```

Windows では、トラブルシューティング・ツールを以下のコマンドで実行します。

```
ssh-troubleshoot.cmd [options] info
```

コマンド・オプションの詳細については、ssh-troubleshoot(8) を参照してください。

収集されたデータは、以下のような名前の結果ファイルに保存されます。

- Unix の場合: `ssh-troubleshoot-data-<hostname>-<timestamp>.tar`
- Windows の場合: `ssh-troubleshoot-data-<hostname>-<timestamp>.log`

ファイル名には、情報を収集したホストを示す `hostname` と、情報がファイルに保存された日時を示す `timestamp` が付けられます。タイムスタンプのフォーマットは `yyyymmdd-hhmmUTC` です。したがって、レポートにはローカル・タイムではなく UTC が使用されます。

このファイルを SSH のテクニカル・サポートに送信し、解析を依頼してください。

警告

出力ファイルにはセキュリティ上重要なデータが含まれている可能性があるため、取り扱いには十分に注意してください。

7.3. 接続ブローカーのデバッグ・モードへの設定

接続ブローカーは Tectia Client に含まれるコンポーネントです。接続ブローカーは Tectia Client とコマンドライン・ツール **sshg3, scpg3**、及び **sftpg3** のすべての暗号操作と認証関連のタスクを処理します。

7.1 で説明されている詳細レベルの出力で問題が解決しない場合は、既存の実行中の Broker をデバッグ・モードに設定してください。既存の開いている接続は稼働したままになるため、マルチユーザ・システムや、多くの自動化スクリプトが同時に実行されるような場合にも適しています。また、新しい接続試行からデバッグ・ログを取得することもできます。

接続ブローカーをデバッグ・モードに設定するには、以下の手順に従ってください。

1. シェル (Unix の場合) またはコマンド・プロンプト・ウィンドウ (Windows の場合) を開きます。

2. すでに接続ブローカーが起動している場合は、この手順はスキップします。接続ブローカーがまだ起動していない場合は、以下のコマンドを実行します。

```
$ ssh-broker-g3
```

3. 以下のコマンドを実行して、接続ブローカーをデバッグ・モードに設定します。

```
$ ssh-broker-ctl debug --log-file=<logfile> <debug-level>
```

コマンドの中の、

- `logfile` では、デバッグ出力先のファイルを指定します。
- `debug-level` は 0 (デバッグ情報なし) から 99 までの整数で、必要なデバッグ情報の量を指定します。

注意

推奨のデバッグ・レベルは 1 ～ 9 です。この数値が高いほど、トラブルシューティングの出力が詳細になり、デバッグが性能に影響するようになります。

Windows では、[**Tectia 接続ステータス**] ウィンドウの [**ログ**] ビューでもデバッグ・モードを設定できます。[**Tectia 接続ステータス**] ウィンドウを開くには、Windows タスクバーの通知領域で **Tectia** アイコンを右クリックし、[**ステータス**] を選択します。

図7.2 Windows での接続ブローカーのデバッグ・モードの設定

以下のコマンド例では、接続ブローカーのデバッグ・モードがレベル 4 に設定され、デバッグ情報が `broker.log` というログ・ファイルに出力されます。

```
$ ssh-broker-ctl debug --log-file=broker.log 4
```

4. いずれかのクライアントを使用してサーバに接続します。

```
$ sshg3 user@host
```

5. broker.log ファイルで接続のデバッグ情報を確認します。

Unix では、コマンドライン・ツールで引数 -D を指定することによってデバッグ出力を表示することもできます。たとえば、以下のコマンドを実行すると、デバッグ・レベル 2 のデバッグ出力が表示されます。

```
$ sftpg3 -D2 user@host
```

Windows では、コマンドライン・ツールの他に、[**Tectia 接続ステータス**] ウィンドウにもデバッグ出力を表示できます。

注意

デバッグは性能を低下させるので、デバッグ出力を収集した後は、Tectia Client のデバッグ・モードを忘れずに無効にしてください。

Unix と Windows では、以下のコマンドでデバッグ・モードを無効にできます。

```
$ ssh-broker-ctl debug --clear
```

Windows では、図 7.3 に示すように、[**Tectia 接続ステータス**] ウィンドウでデバッグ・レベルを 0 に戻すことでデバッグ・モードを無効にすることもできます。

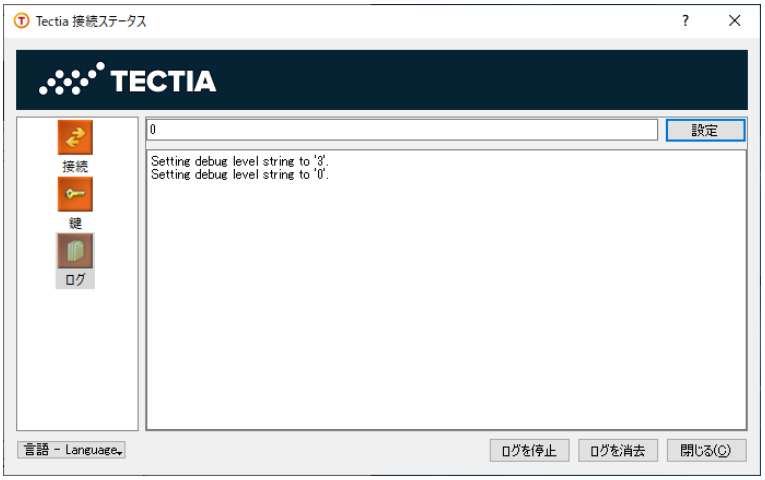


図7.3 Windows での接続ブローカーのデバッグ・モードの無効化

7.4. よくある問題への回答

本項では、いくつかの問題に対する回避策を紹介します。

GSSPI 認証のトラブルシューティング

Windows 5.x または 6.x クライアントから Windows 4.x サーバに GSSAPI 認証で接続する場合、GSSAPI が正しく設定されているにもかかわらず認証に失敗する時は、クライア

ント側のコンピュータで LMHOSTS 検索を無効にしなければならないことがあります。以下の手順に従ってください。

1. [**コントロールパネル**] → [**ネットワーク接続**] を選択します。
2. [**ローカル エリア接続**] で [**プロパティ**] を右クリックして選択します。
3. [**ローカル エリアの接続プロパティ**] ダイアログボックスの [**全般**] タブで、[**インターネットプロトコルバージョン (TCP/IP)**] を選択し、[**プロパティ**] ボタンをクリックします。
4. [**インターネットプロトコルバージョン (TCP/IP)のプロパティ**] ダイアログボックスの [**全般**] タブで、[**詳細設定**] ボタンをクリックします。
5. [**TCP/IP 詳細設定**] ダイアログボックスの [**WINS**] タブで、[**LMHOSTSの参照を有効にする**] チェックボックスを選択解除します。
6. クライアント側のコンピュータを再起動します。

パスワード・ウィンドウのフォーカスが外れる

Windows クライアントで、パスワードや公開鍵のパスフレーズのウィンドウのフォーカスが外れることがあります。この状態では、パスワード・ウィンドウは画面上でアクティブに表示されますが、パスワードを入力し始めても、ボックスにアスタリスクが表示されません。実際のフォーカスが他のウィンドウにある場合、パスワードやパスフレーズがそこに表示されることがあります。

パスワード・ウィンドウのフォーカスが外れる現象は、Tectia 接続ブローカーの [ステータス] ウィンドウを開いているときに最も多く発生します。したがって、最初の回避策として、ログイン開始時に [ステータス] ウィンドウが開いていないことを確認してください。

この問題の恒久的な解決策は、アプリケーションの動作を制御する Windows の設定を変更することです。Windows のデフォルト設定では、アプリケーションがフォーカスを奪わないようにする設定が有効になっています。回避策として、アプリケーションがフォーカスを奪えるように設定を変更できます。この設定は Tectia 接続ブローカーだけでなく、すべての Windows アプリケーションに影響することに注意してください。

設定を変更するには、Microsoft Tweak UI ユーティリティ・プログラムをダウンロードする必要があります。以下の手順に従ってください。

1. Microsoft Tweak UI ユーティリティを http://windows.microsoft.com/en-US/windows/xp-downloadsからダウンロードします。
2. ユーティリティに付属のインストール・ウィザードに従って、コンピュータに Tweak UI をインストールします。
3. [スタート] メニューからプログラムを起動します。
4. [**一般**] - [**フォーカス**] ビューに移動し、下図に示すように [**他のアプリケーションから強制的にフォーカスを奪うことを防ぐ**] チェックボックスを選択解除します。

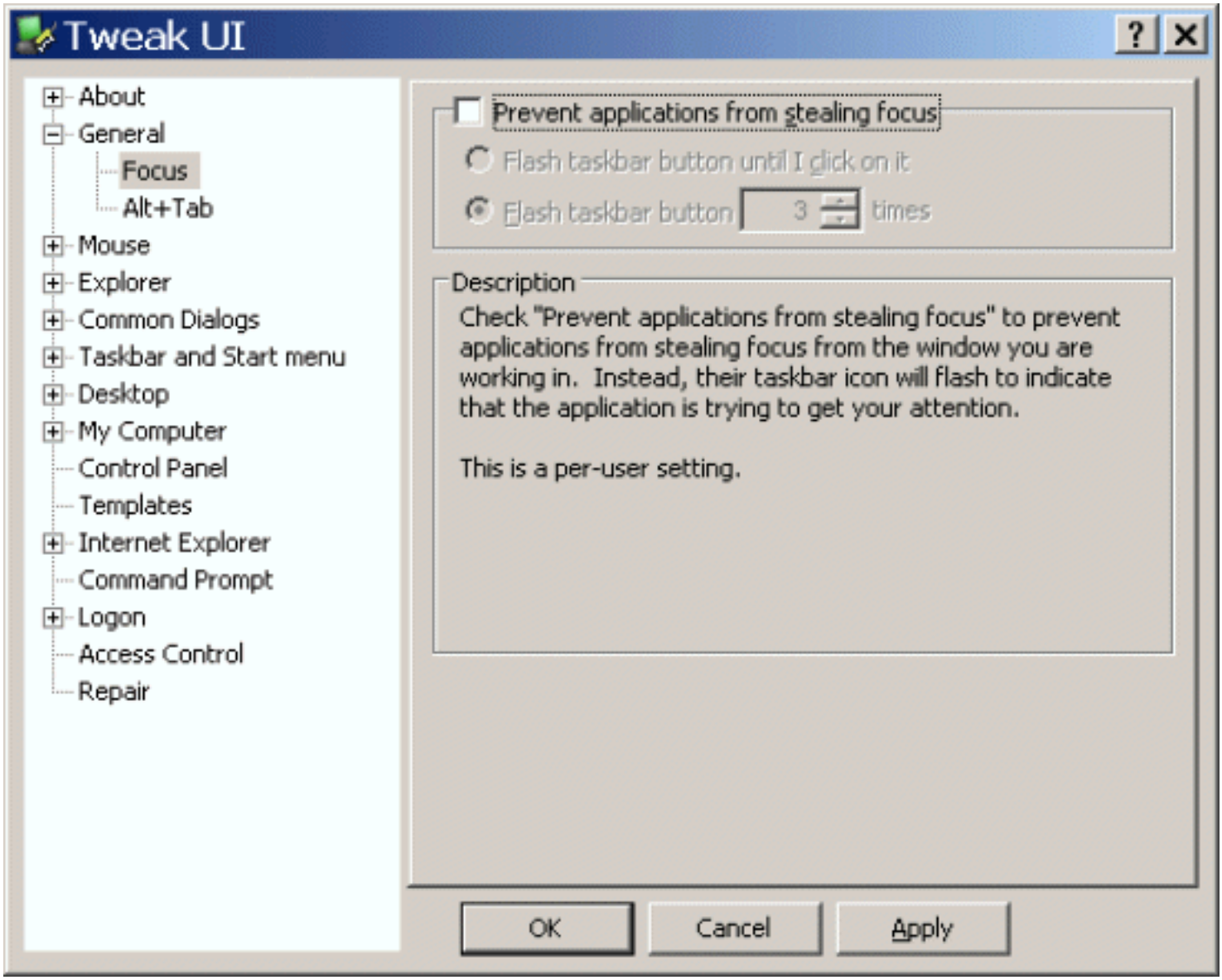


図7.4 Microsoft Tweak UI ユーティリティ

5. [**適用**] または [**OK**] をクリックします。

付録A 接続ブローカー設定ツール

接続ブローカーは Tectia Client に含まれるコンポーネントで、Tectia Client のすべての暗号操作と認証関連のタスクを処理します。このため、認証や接続プロファイルの設定はすべて接続ブローカーの設定で行います。

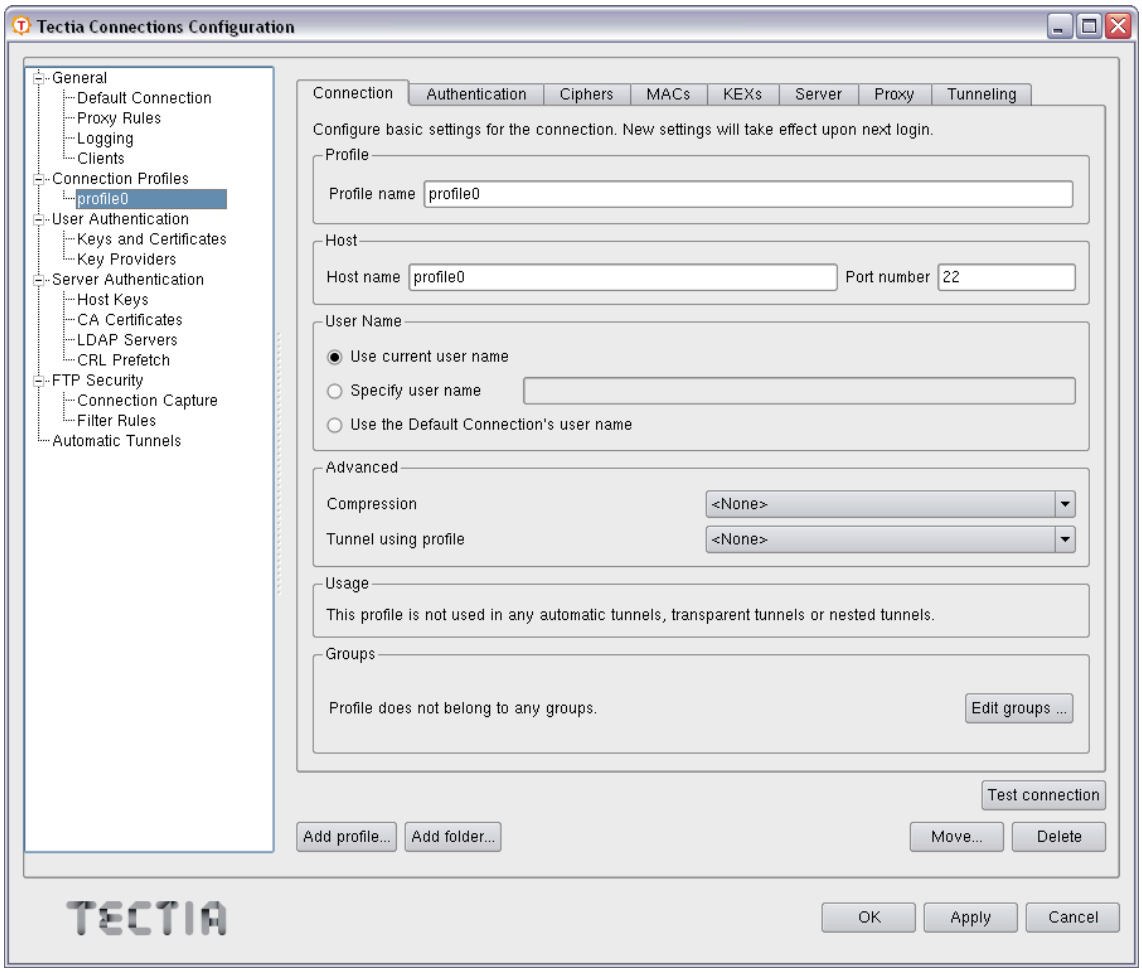
接続ブローカーの設定は Tectia コネクション設定　GUI から編集及び閲覧できます。手順については、A.1 を参照してください。

接続ブローカーは、XML ベースの設定ファイル `ssh-broker-config.xml` に設定を保存します。アスキー・テキスト・エディタや　XML　エディタで設定ファイルを直接編集することもできます。設定ファイルの詳細については、ssh-broker-config(5)　を参照してください。設定ファイルのエレメントとその属性のクイック・リファレンスについては、A.4　を参照してください。

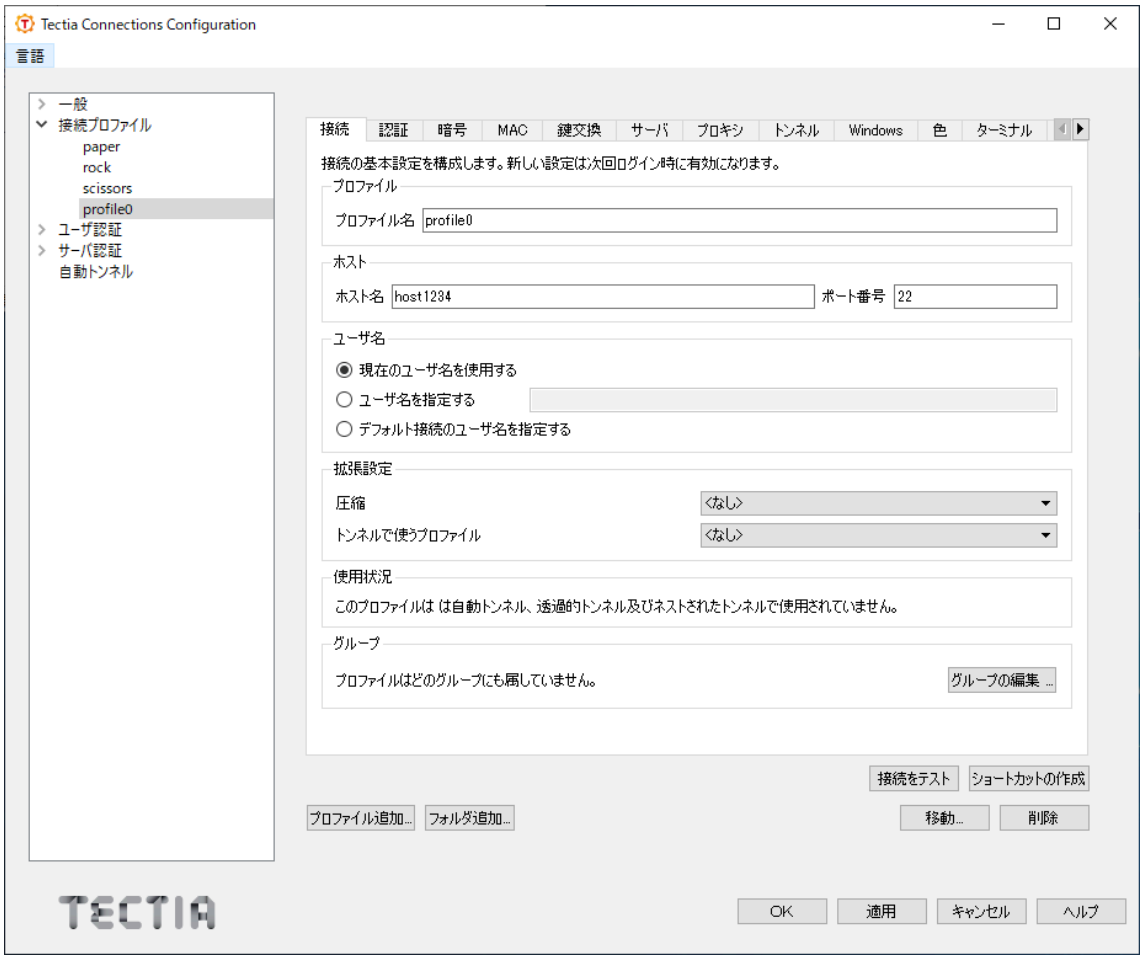
A.1. Tectia コネクション設定　GUI

Tectia コネクション設定　GUI を使用して、Tectia Client に含まれる接続ブローカーの認証及び接続プロファイルの設定を編集できます。

Tectia コネクション設定　GUI は Windows 及び Linux 上で動作し、Tectia Client 及び Tectia ConnectSecure で利用できます。OS プラットフォームや製品バージョンの違いにより、GUI のオプションに違いがある場合があります。以降、主にWindows 上で動作する Tectia Client の画面をスクリーンショットとして示します。重要な違いがある場合は、Linux バージョンの画面も示します。



図A.1 Linux の [Connection Profiles] タブ



図A.2 Windows の [接続プロファイル] タブ

Linux では、Tectia コネクション設定　GUI のインターフェイスは Windows バージョンと異なり、中でも注目すべき点として、以下の例外があります。

- 接続プロファイルの　[**Windows**]、[**色**]、[**ターミナル**]、[**ファイル転送**]、及び[**お気に入りフォルダ**] のタブは使用できません。
- [**鍵プロバイダ**] ページの `Microsoft Crypto API` オプションは使用できません。
- Linux の GUI には、ヘルプ・コンテンツにアクセスするための [**ヘルプ**] ボタンがありません。

注意

KDE 及び Gnome のウィンドウ・マネージャのみサポートしています。その他の互換性のあるマネージャについては、Qt　フレームワークのウェブサイトを確認してください。

A.1.1. GUI の起動

Windows

Windows では、以下の複数の方法で **Tectia コネクション設定　GUI** にアクセスできます。

- Windows タスクバーの通知領域にある Tectia トレイ・アイコン  を右クリックしてショートカット・メニューを表示し、[**設定**] を選択します。
- Tectia SSHターミナル GUI がアクティブになっている場合は、ツールバーの Tectia コネクション設定　GUI アイコン  をクリックするか、またはメニュー・バーの [**編集**]→[**Tectia 接続**] を選択します。

Windows では、Tectia SSHターミナル GUI のユーザ・インターフェイスの設定を行うための GUI が Tectia Client に別途用意されています。Tectia SSHターミナル GUI のツールバーにある  アイコンをクリックして、[**設定**] ツールを開きます。GUI の設定方法については、付録 B を参照してください。

Linux

Linux では、以下を実行することで **Tectia コネクション設定　GUI** を起動できます。

```
$ /opt/tectia/bin/ssh-tectia-configuration
```

以下のオプションがあります。

`-f, --config=FILE`

設定ファイル `FILE` を使用します。

`-a, --broker-address=ADDR`

指定されたアドレスを使って、別の接続ブローカープロセスに接続します。

`-d, --debug=STR`

デバッグ文字列を `STR` に設定します。

`--convert`

古い設定ファイルを変換します。

`--new-profile`

新しいプロファイルを追加します。

`--profile-host=HOST`

新しいプロファイルを追加するときのプロファイル・ホスト名。

`--profile-port=PORT`

新しいプロファイルを追加するときのプロファイル・ポート。

`--profile-user=USER`

新しいプロファイルを追加するときのプロファイル・ユーザ。

`--edit-profile=NAME`

既存のプロファイル `NAME` を編集します。

`--ui-mode=MODE`

ユーザ・インターフェイス・モード。指定可能な値は `standard` 及び `file-transfer` です。

`-V, --version`

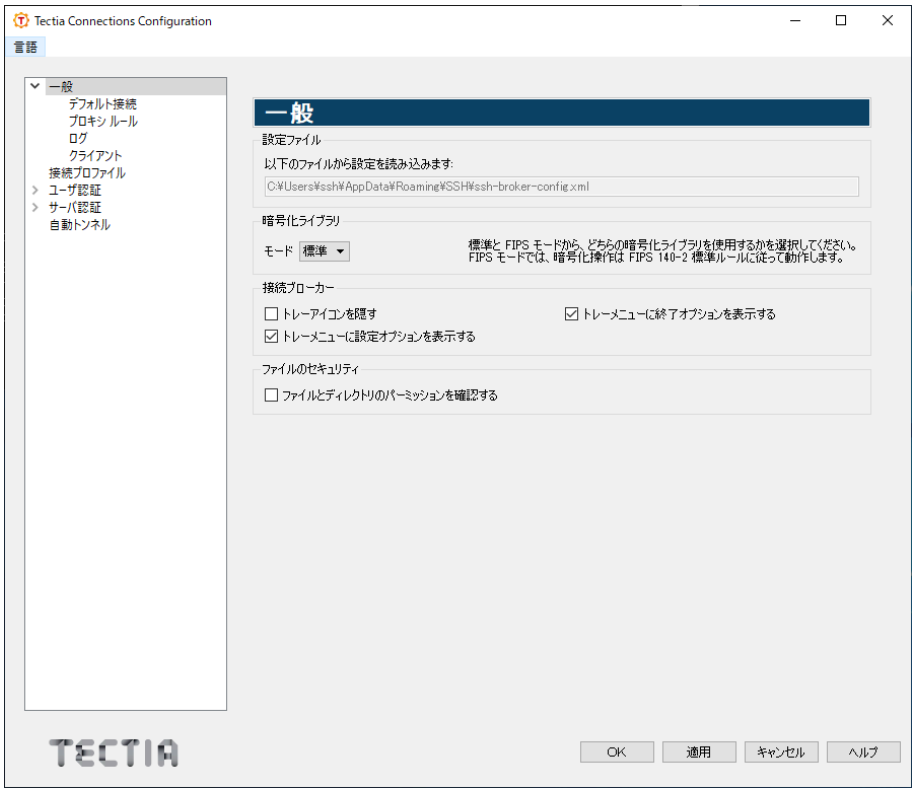
バージョンを表示します。

`-h, --help`

使用状況を表示します。

A.1.2. 一般設定の定義

[一般] ページでは、使用する暗号化ライブラリの選択と、Tectia トレイ・アイコンの設定を定義できます。



図A.3 一般設定

設定ファイル

ユーザ固有のブローカー設定ファイルの場所を表します。デフォルトの場所は、Windows では "`%APPDATA%\SSH\ssh-broker-config.xml`" Linux では "`$HOME/.ssh2/ssh-broker-config.xml`" です。

設定ファイルを保存するたびに、古い設定ファイルのバックアップが、Windows では "`%APPDATA%\SSH\ssh-broker-config.xml.bak`" に、Linux では "`$HOME/.ssh2/ssh-broker-config.xml.bak`" に保存されます。

暗号化ライブラリ

Tectia Client は、連邦情報処理標準 (FIPS) 140-2 に準拠して検証されたバージョンの暗号化ライブラリを使用して、FIPS モードで動作させることができます。このモードでは、FIPS 140-2 標準の規則に従って暗号化処理が行われます。FIPS モードでは、OpenSSL 暗号化ライブラリが使用されます。

暗号化ライブラリについて、[**標準**] または [**FIPS**] 140-2 認証バージョンのどちらを使用するか選択します。

デフォルトの設定については 「暗号の定義」、「MAC の定義」 を、プロファイル固有の設定については 「暗号の定義」 及び 「MAC の定義」 を参照してください。

接続ブローカー

Windows タスクバーの通知領域から Tectia のトレイ・アイコンを隠すかどうか、ショートカット・メニューに [**終了**] 及び [**設定**] のオプションを表示するかどうかを選択します。

ファイルのセキュリティ (Linux でのみ使用可能)

ユーザ固有の構成ファイル (`$HOME/.ssh2/ssh-broker-config.xml`) 及び秘密鍵ファイルのアクセス・パーミッションのチェックを有効にするには、[**ファイルとディレクトリのパーミッションを確認する**] チェックボックスを選択します。デフォルトでは、ファイル及びディレクトリのアクセス・パーミッションはチェックされません。

ファイル及びディレクトリのアクセス・パーミッションをチェックする場合、以下の制御が適用されます。

- ユーザ設定ファイルに期待されるパーミッション: ユーザのみが読み取りと書き込みの権限を持ちます。パーミッションがこれ以上広がると、接続ブローカーは起動しません。
- 秘密鍵ファイルに期待されるパーミッション: ユーザのみが読み取りと書き込みの権限を持ちます。パーミッションがこれ以上広がると、チェックを通らない鍵は無視されます。

デフォルト接続設定の定義

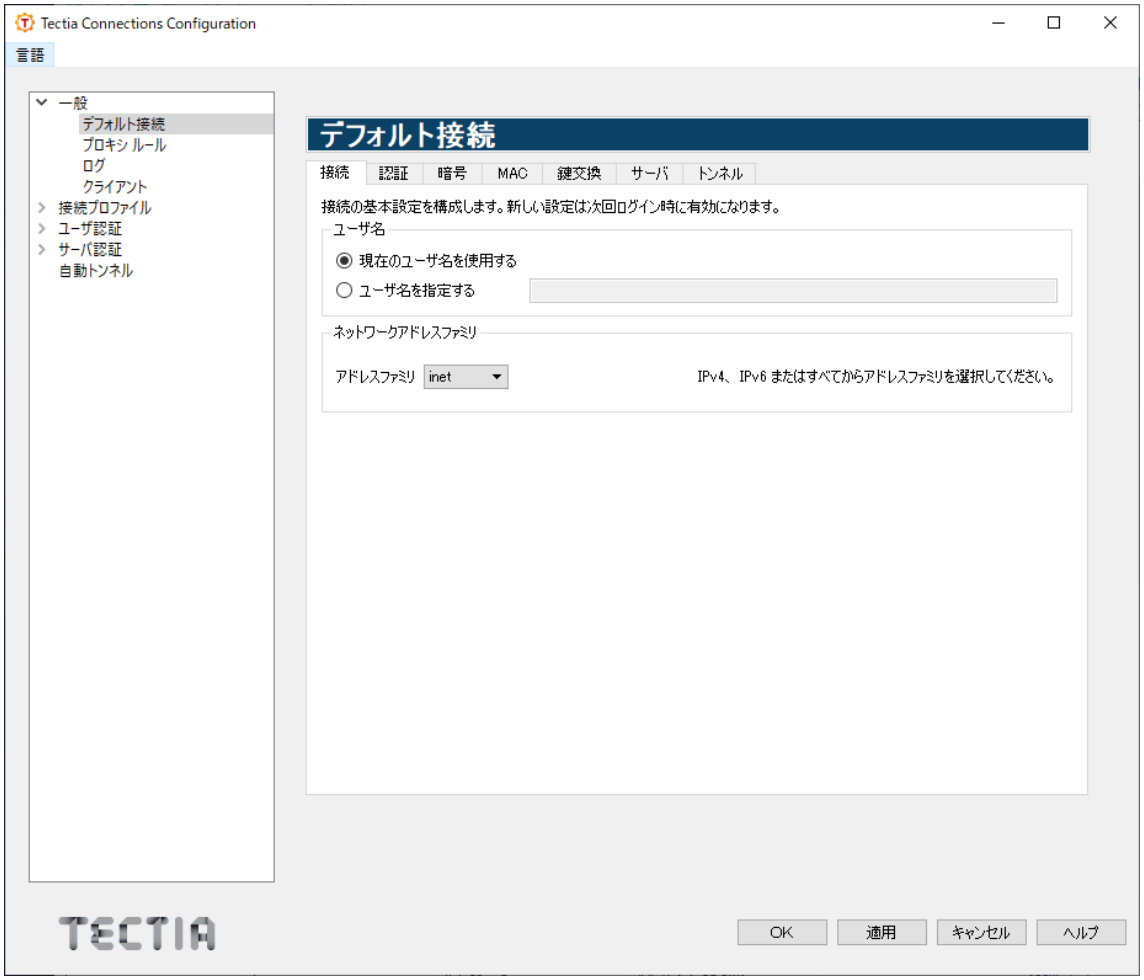
[**デフォルト接続**] ページでは、ユーザ名 (「接続設定の定義」)、認証 (「認証の定義」)、暗号 (「暗号の定義」)、MAC (「MAC の定義」)、鍵交換 (「鍵交換の定義」)、サーバ接続

(「サーバ接続の定義」)、及びトンネリング (「デフォルトのトンネリング設定の定義」) のデフォルト設定を編集できます。

新規に作成された接続プロファイルは、ここで定義されたデフォルトの設定を引き継ぎます。これらの値は、プロファイル固有のタブ付きページでカスタマイズでき、デフォルトの設定を上書きします「認証の定義」、「暗号の定義」、「MAC の定義」、及び「サーバ接続の定義」を参照してください。

接続設定の定義

[**接続**] タブでは、リモート・サーバに接続する際に使用するデフォルトのユーザ名を定義できます。この接続は、複数のユーザがそれぞれのシステム・ユーザ名で、または共通のユーザ・アカウントでプロファイルを共同で使用するときに便利です。



図A.4 接続に関するユーザ名とネットワーク・アドレス・ファミリの設定

[**現在の Windows ユーザ名を使用する**] オプションを選択すると、現在ログインしているユーザの Windows ユーザ名が、リモート・サーバへの接続に自動で適用されます。

[**ユーザ名を指定する**] オプションを選択し、汎用的なユーザ名を入力します。名前の大文字と小文字は区別されることに注意してください。

接続プロファイルまたは接続試行で別のユーザ名が指定されない限り、ここで指定されたユーザ名が接続時に使用されます。このオプションを選択し、ユーザ名フィールドを空にすると、ユーザ名の入力を接続ブローカーから求められます。

原則として、"%USERNAME%" という値を入力できますが、これには [**現在の Windows ユーザ名を使用する**] を選択したのと同じ効果があります。

ホスト名を指定した場合、またはプロファイルにホスト名が含まれている場合、接続ブローカーは [**ネットワークアドレスファミリ**] の設定に基づいてアドレス解決を試みます。[inet] を選択した場合、接続ブローカーはホスト名を IPv4 アドレスのみで解決します。[inet6] を選択した場合、接続ブローカーはホスト名を IPv6 アドレスのみで解決します。[すべて] を選択すると、接続ブローカーは使用可能な任意の IP アドレス (IPv4 または IPv6) でホスト名を解決します。

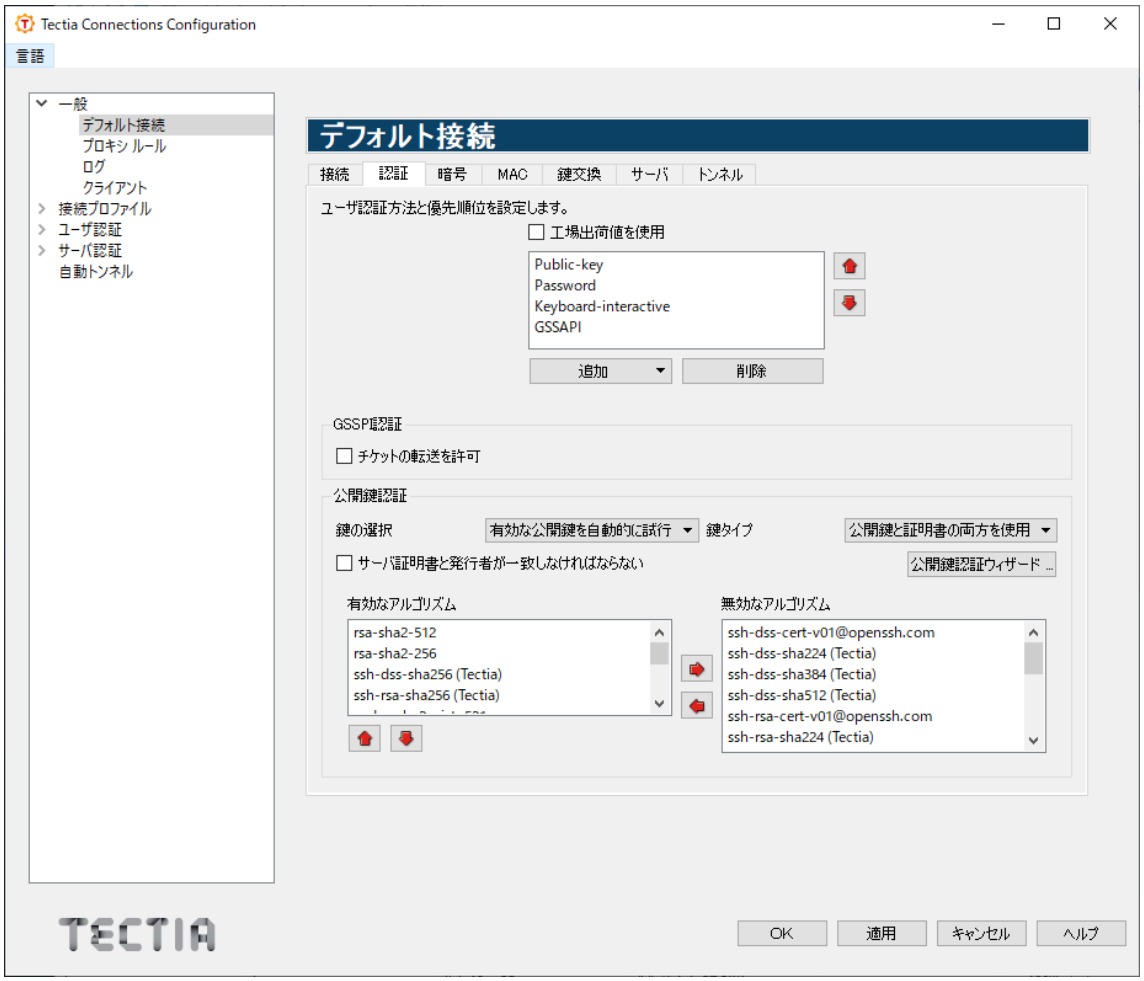
注意

接続プロファイルまたはコマンドラインのいずれかを使用して、接続用の IP アドレス (IPv4 または IPv6 のいずれか) を直接指定できます。この設定は、ユーザが指定するネットワーク・ファミリ・アドレスを制限するものではありません。たとえば、ネットワーク・アドレス・ファミリを IPv6 に設定していても、指定した IPv4 アドレスへの接続が確立されます。

このタブで設定した内容は、ユーザが次にログインしたときに有効になります。

認証の定義

[**認証**] タブでは、デフォルトのユーザ認証方法を定義できます。



図A.5 Tectia Client の認証方法

工場出荷時の認証方法を使用するには、[**工場出荷値を使用**] チェックボックスを選択します。認証方法のカスタム・リストを定義する場合は、このチェックボックスを選択解除します。

Tectia Client 6.6 では、工場出荷時の認証方法は順に以下の通りです。

- Public-key
- Password
- Keyboard-interactive
- GSSAPI

これらの認証方法は、IBM z/OS で使用できない GSSAPI を除き、すべてのプラットフォームでサポートされています。

新しい認証方法をリストに追加するには、[**追加**] をクリックし、ドロップダウン・メニューから方法を選択します。

認証方法を削除するには、リストから方法を選択し、[**削除**] をクリックします。

矢印ボタンを使用して、認証方法の優先順位を並べ替えます。Secure Shell サーバで許可されている最初の方法が使用されます。ログインするために、サーバが複数の認証方法に成功することを要求することもあります。

ユーザ認証には、以下の方法があります。

- **Public-key**: ユーザは公開鍵認証の使用を要求されます。A.1.4 も参照してください。
- **Password**: ユーザは、認証のためにパスワードの入力を要求されます。
- **Keyboard-interactive**: キーボード・インタラクティブは、Secure Shell クライアントが RSA SecurID や PAM など、複数の異なるタイプの認証方法をサポートできるように設計されています。キーボード・インタラクティブの詳細については、4.8 を参照してください。
- **GSSAPI**: GSSAPI (ジェネリック・セキュリティ・サービス・アプリケーション・プログラミング・インターフェイス) は、異なるセキュリティ対策を 1 つのインターフェイスで利用できるようにした、共通のセキュリティ・サービス・インターフェイスです。GSSAPI の詳細については、4.9 を参照してください。

Tectia Client で複数の接続にわたる Kerberos チケットの転送を許可するには、[**GSSPI認証**] フィールドにある [**チケットの転送を許可**] チェックボックスを選択します。

[**公開鍵認証**] を使用する場合は、使用する鍵タイプや鍵の選択方法も定義できます。

鍵の選択 では、ユーザの公開鍵をサーバに提示する際に接続ブローカーが使用するポリシーを定義します。ドロップダウン・リストからモードを選択します。以下のオプションがあります。

- **有効な公開鍵を自動的に試行**(デフォルト)。このポリシーでは、クライアントは以下の順に鍵を試行します。
  1. 公開鍵が使用可能で、秘密鍵にパスフレーズがない鍵 (ユーザの操作なし)
  2. 公開鍵が使用可能であるが、秘密鍵にパスフレーズが設定されている鍵 (鍵がサーバに受け入れられるという前提で、パスフレーズの提供が必要)
  3. 残りの鍵。つまり、秘密鍵だけでなく、公開鍵にもパスフレーズが必要な鍵。
- **鍵選択プロンプトから選択** - このポリシーでは、使用可能な鍵のリストから鍵を選択するよう接続ブローカーから求められます。選択した鍵による認証に失敗した場合、再度別の鍵を選択するようクライアントから求められます。

鍵タイプ では、公開鍵認証の際に、プレーンな公開鍵のみを試すか、証明書のみを試すかを定義します。ドロップダウン・リストから鍵の種類を選択します。デフォルトでは、プレーンな公開鍵と証明書の両方を試します。

[**サーバ証明書と発行者が一致しなければならない**] チェックボックスを選択すると、接続ブローカーはユーザに提示されるユーザ証明書の一覧をフィルタリングします。クライアント側のユーザ証明書は、その発行者名が、サーバが要求したまたは受け入れた証明書発行者と比較されることでフィルタリングされます。デフォルトでは、フィルタリングは行われません。このオプションは、テスト用ロールと管理者ロールなど、ユーザが同じサーバに対して

異なるアクセス権を持つ複数の証明書を有している場合に便利です。接続ブローカーは、リモート・ホストで適用可能な該当する証明書を選別するので、ユーザは絞り込まれた証明書の中から正しい証明書を選択できます。

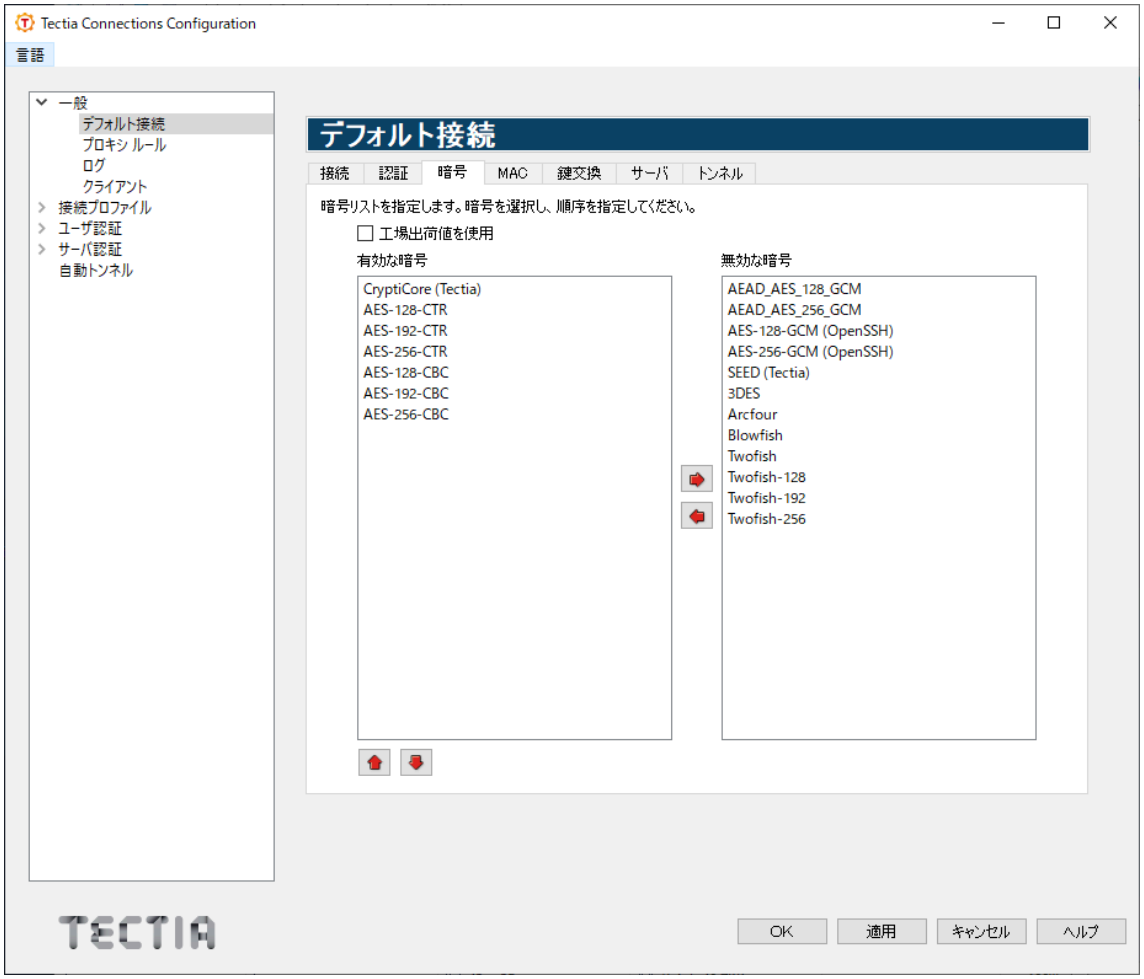
新しい公開鍵ペアを生成し、鍵の公開部分をサーバにアップロードするには、[**公開鍵認証ウィザード**] ボタンをクリックします。詳細については、「公開鍵認証ウィザードの使用」を参照してください。

[**有効なアルゴリズム**] には、ユーザの公開鍵の認証と署名に使用される公開鍵署名アルゴリズムがリ一覧表示されます。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方に設定されているものです。アルゴリズムの順序は、上下の矢印ボタンを使用して変更できます。アルゴリズムを [**無効なアルゴリズム**] リストに移動させるには、そのアルゴリズムを選択して右矢印ボタンをクリックします。

工場出荷時の公開鍵署名アルゴリズムは F.4 を参照してください。

暗号の定義

[**暗号**] タブでは、使用する暗号化アルゴリズムを定義できます。



図A.6 暗号リストの定義

工場出荷時のアルゴリズムを使用するには、[**工場出荷値を使用**] チェックボックスを選択します。暗号リストを定義する場合は、矢印ボタンを使用します。暗号は、指定された順に試行されます。

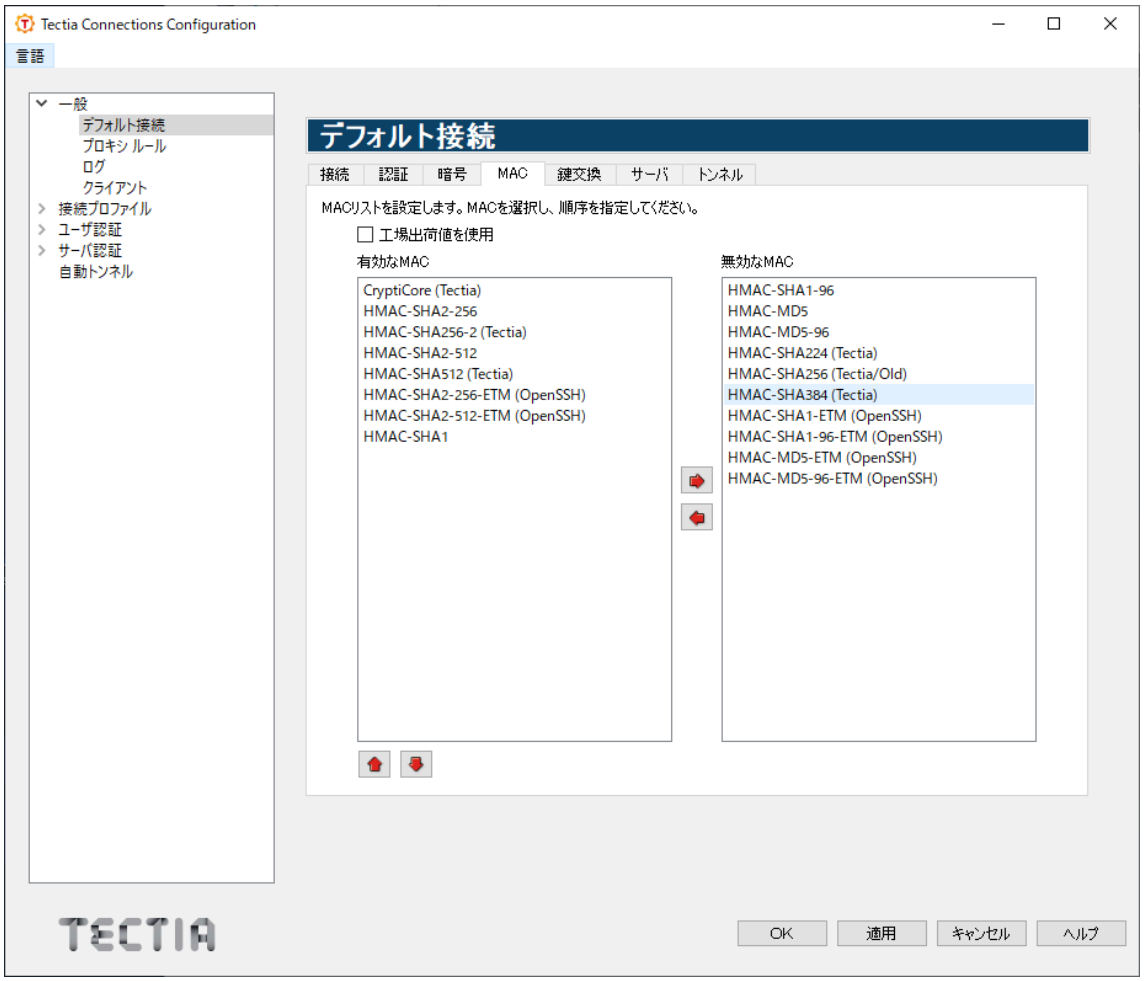
工場出荷時の暗号は F.1 を参照してください。

工場出荷時の暗号は順に以下の通りです。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

MAC の定義

[**MAC**] タブでは、使用するメッセージ完全性アルゴリズムを設定できます。



図A.7 MAC リストの定義

工場出荷時のアルゴリズムを使用するには、[**工場出荷値を使用**] チェックボックスを選択します。MAC リストを定義する場合は、矢印ボタンを使用します。MAC は、指定された順に試行されます。

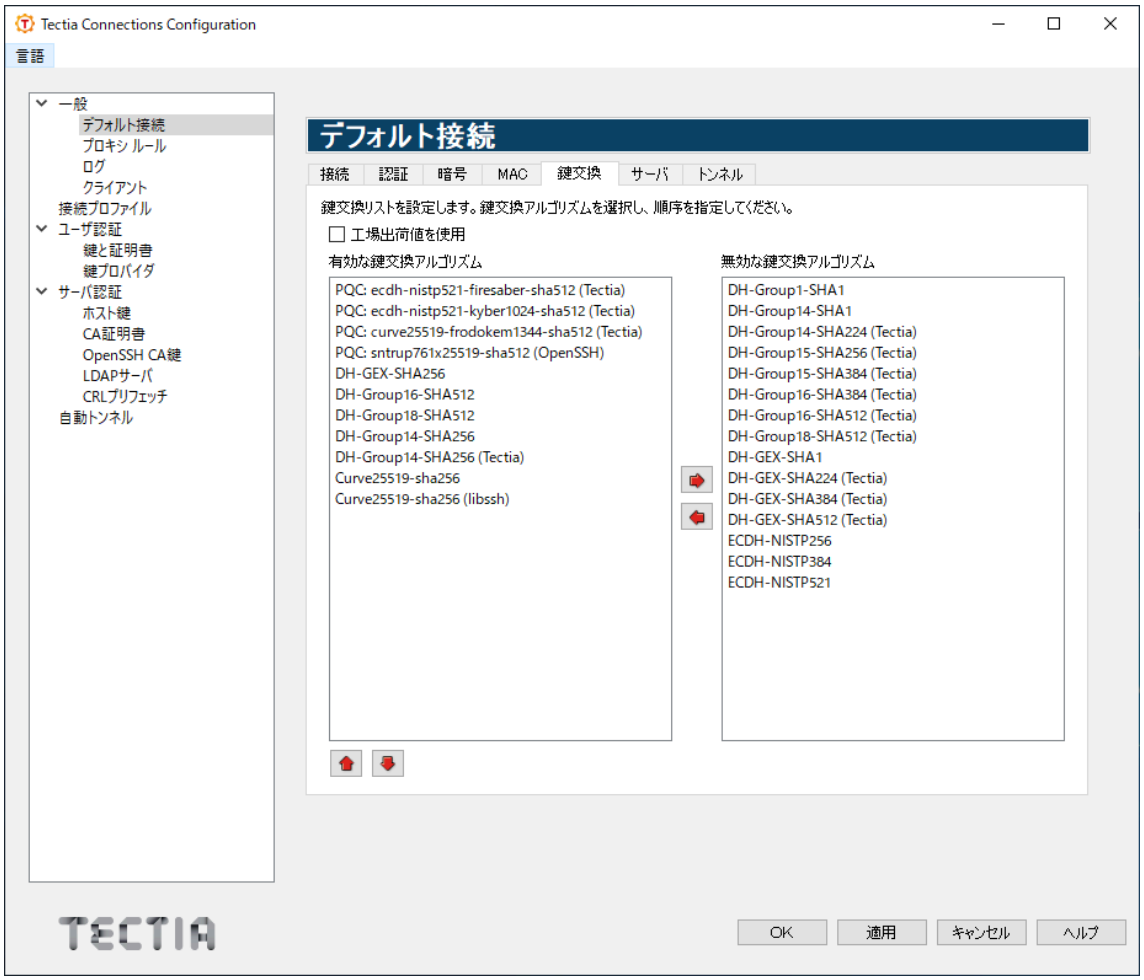
工場出荷時の MAC は F.3 を参照してください。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

(OpenSSH) の記載のあるアルゴリズムは、接続ブローカーの設定ファイルの `@openssh.com` で終わるアルゴリズムに対応しています。

鍵交換の定義

[**鍵交換**] タブでは、使用する鍵交換方法を設定できます。



図A.8 鍵交換リストの定義

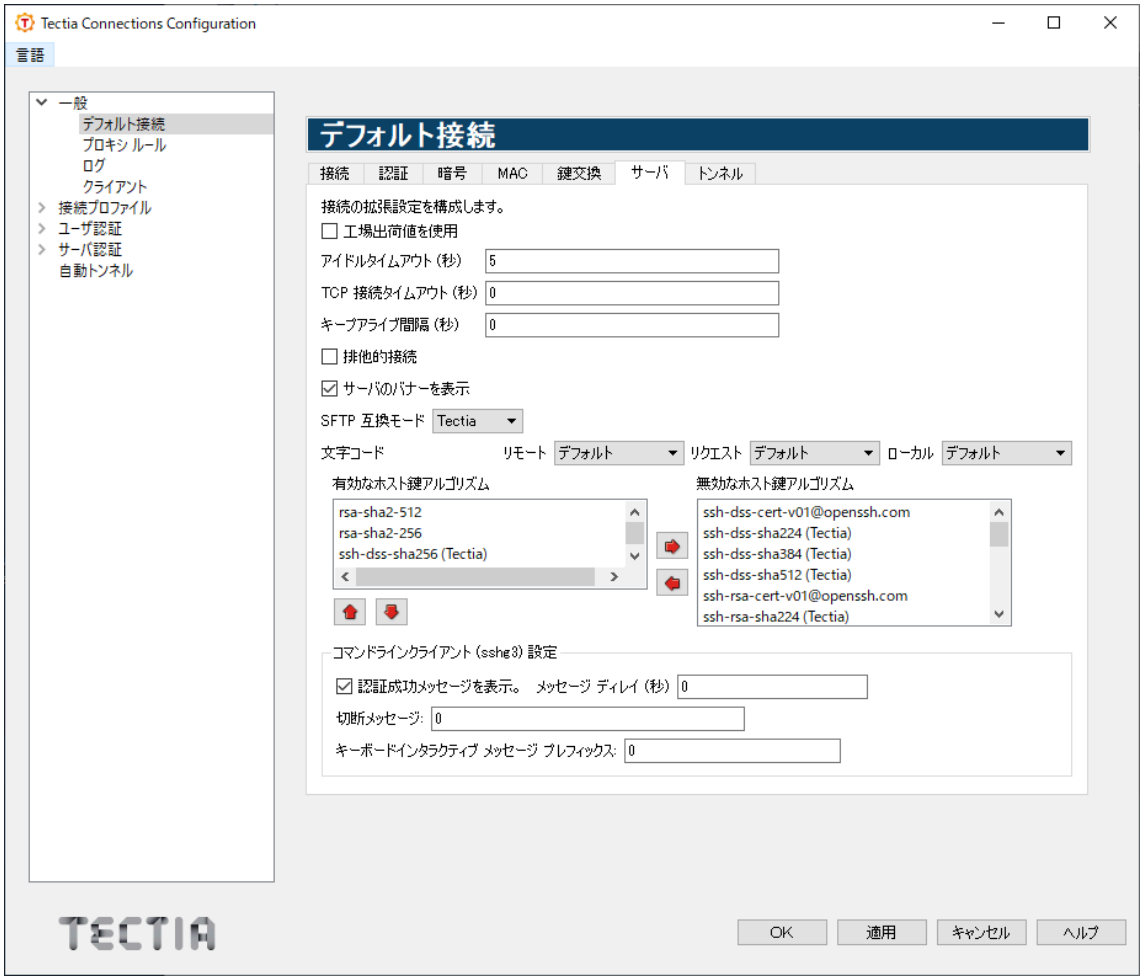
工場出荷時の方法を使用するには、[**工場出荷値を使用**] チェックボックスを選択します。鍵交換リストを定義する場合は、矢印ボタンを使用します。鍵交換方法は、指定された順に試行されます。

工場出荷時の鍵交換は F.2 を参照してください。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

サーバ接続の定義

[サーバ] タブでは、詳細なサーバ接続設定を定義できます。



図A.9 サーバ接続設定の定義

工場出荷値を使用

サーバ接続設定にデフォルト値を使用するには、チェックボックスを選択します。

アイドルタイムアウト

接続を自動的に閉じるまでにどのくらいの長さのアイドル時間(すべての接続チャネルが閉じた後)が許可されるのか を指定します。デフォルトは 5 秒です。長い時間を設定すると、セッション (Tectia SSHターミナル GUI など) を閉じた後でもサーバへの接続を維持できます。この時間の間は、再認証を行わずにサーバとの新しいセッションを開始できます。時間を 0 (ゼロ) に設定すると、サーバとの最後のチャネルが閉じられたときに、接続がすぐに終了します。

TCP 接続タイムアウト

Secure Shell サーバへの TCP 接続を試行する時間を指定します。タイムアウトを秒単位で定義します。その時間が経過してもリモート・サーバがダウンしているか、または到

達できない場合、TCP 接続が解放されます。値を 0 (ゼロ) に設定すると、この Tectia の設定が無効になり、システムのデフォルトの TCP タイムアウトが使用されます。デフォルトでは、システムのタイムアウトが使用されます。

キープアライブ間隔

Secure Shell サーバにキープアライブ・メッセージを送信する間隔 (秒単位) を指定します。デフォルトは 0 で、この場合、キープアライブ・メッセージは送信されません。

排他的接続

現在開いている接続を再利用するのではなく、常に新しい接続を開くようにする場合は、このチェックボックスを選択します。

サーバのバナーを表示

ログイン前にサーバ・バナー・メッセージ・ファイル (存在する場合) を表示させる場合は、このチェックボックスを選択します。

認証成功メッセージを表示

`AuthenticationSuccessMsg` メッセージを出力してログに記録しない場合は、このチェックボックスを選択解除します。デフォルトでは、このメッセージは有効になっています。

メッセージ ディレイ (秒)

認証成功のメッセージが表示され時間の長さを指定します。デフォルト値は `2` です。値を `0` にすると、メッセージは表示されず、ログに記録されるだけになります。

切断メッセージ

切断したときに表示されるメッセージを指定します。デフォルトでは無効です。使用できる変数の一覧は **disconnect-message** を参照してください。

キーボード インタラクティブ メッセージ プレフィクス

キーボード・インタラクティブのプロンプトの前に表示される文字列です。デフォルト値は `${host}>` です。使用できる変数の一覧は **keyboard-interactive** を参照してください。

SFTP 互換モード

SFTP でファイルを転送するのに適したモードを選択します。この設定は、 **get/mget/sget** 及び **put/mput/sput** コマンドの動作と、 **sftpg3** クライアントが使用する再帰レベルに影響します。以下のオプションがあります。

- `Tectia` (デフォルト) - **sftpg3** は、カレント・ディレクトリとそのすべてのサブディレクトリから再帰的にファイルを転送します。
- `OpenSSH` - 指定されたディレクトリから通常のファイルとシンボリック・リンクのみをコピーします。サブディレクトリはコピーされません。それ以外の場合、 **get** コマンドのセマンティクスは変更されません。

- FTP - **get/put** コマンドは **sget/sput** として実行されます。つまり、1 つのファイルが転送され、サブディレクトリはコピーされません。

再帰の深さは、 **sftpg3** クライアントのコマンド **get/put/mget/mput** でコマンドライン・オプション `--max-depth="LEVEL"` を指定して上書きできます。詳細については、sftpg3(1) を参照してください。

有効なホスト鍵アルゴリズム

このリストには、ホスト鍵または証明書を用いたサーバ認証に使用されるホスト鍵署名アルゴリズムが表示されます。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方の設定ファイルで定義されているものです。このようにすることで、SHA-2 のような特定のアルゴリズムのみを使用するようにサーバから強制できます。

ホスト鍵アルゴリズムは、指定された順に試行されます。例外: サーバのホスト鍵がすでにクライアントのホスト鍵ストアに存在する場合は、そのアルゴリズムが優先されます。アルゴリズムの順序は、上下の矢印ボタンを使用して変更できます。

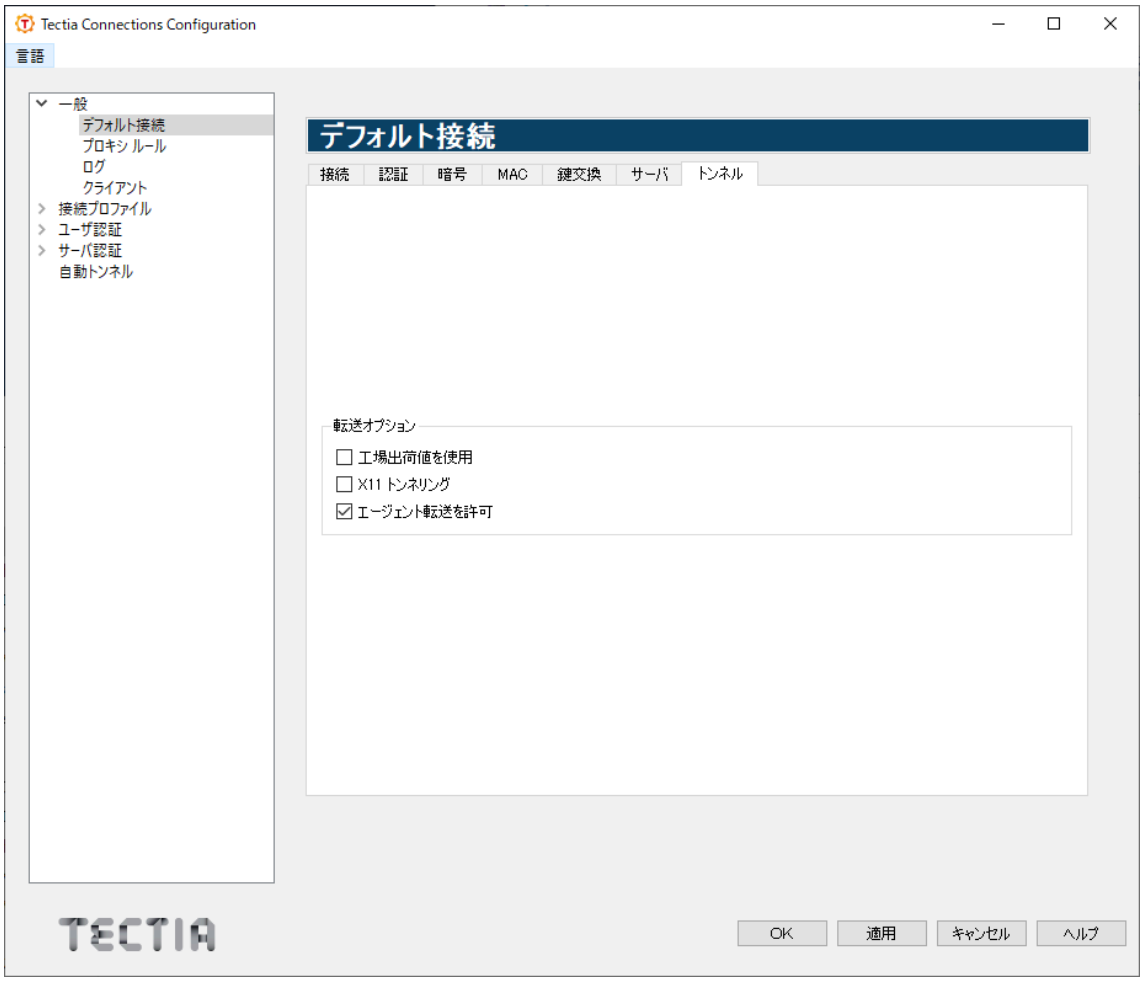
工場出荷時のホスト鍵アルゴリズムは F.4 を参照してください。

無効なホスト鍵アルゴリズム

ここにリストアップされているホスト鍵アルゴリズムは、サーバ認証には使用されません。ホスト鍵アルゴリズムを無効にするには、[**有効なホスト鍵アルゴリズム**] リストでそのアルゴリズムを選択し、右矢印ボタンをクリックします。

デフォルトのトンネリング設定の定義

[**トンネル**] タブでは、X11 接続とエージェント転送 (トンネリング) のデフォルト設定を定義できます。デフォルトは、新しい接続プロファイルと、独自のトンネリング設定が定義されていない接続プロファイルに適用されます。



図A.10 デフォルトのトンネリング設定の定義

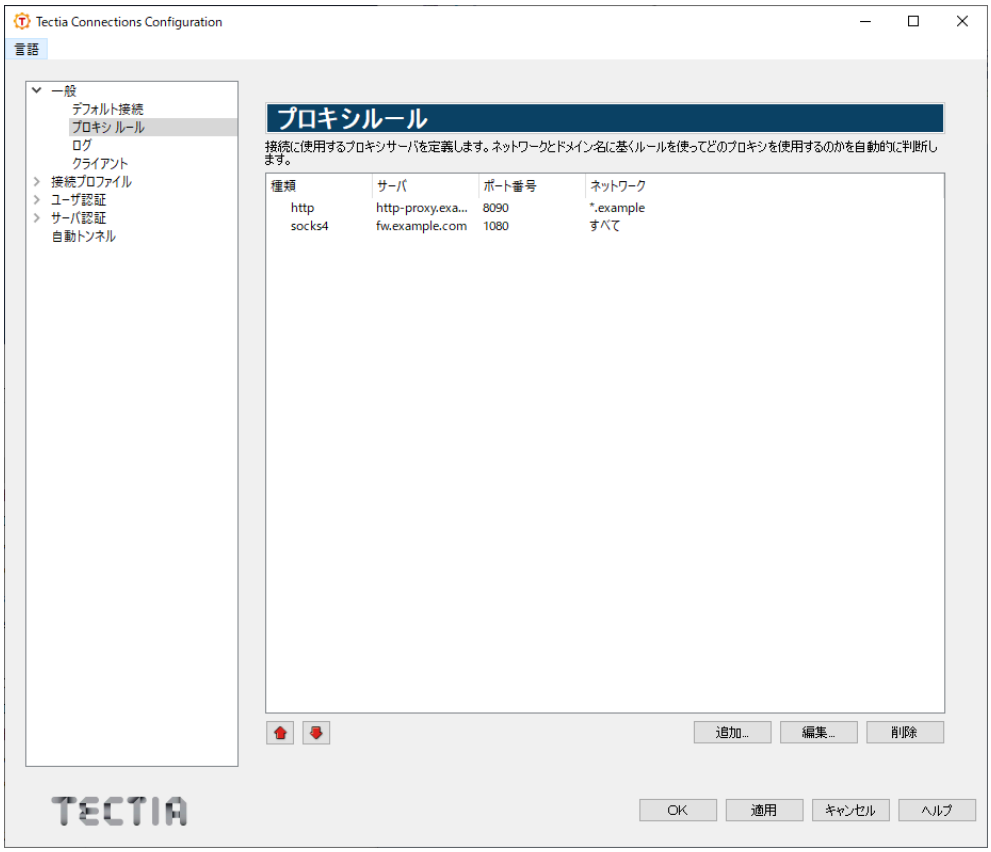
X11 転送及びエージェント転送に工場出荷値を適用するには、[**工場出荷値を使用**] チェックボックスを選択します。工場出荷値では、X11 転送は無効 (選択解除)、エージェント転送は有効 (選択済み) になっています。

クライアント側で X11 転送を許可するには、[**X11 トンネリング**] チェックボックスを選択します。

クライアント側でエージェント転送を無効にするには、[**エージェント転送を許可**] チェックボックスを選択解除します。

プロキシ・ルールの定義

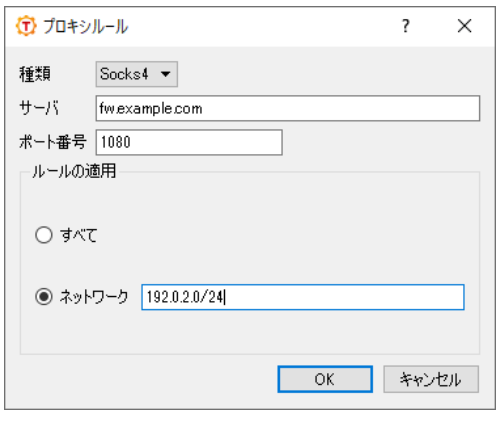
[**プロキシルール**] ページでは、接続に使用するプロキシ・ルールを定義できます。



図A.11 プロキシ・ルールの定義

新しいプロキシ・ルールを追加するには、以下の手順に従ってください。

1. [**追加**] をクリックします。[**プロキシルール**] ダイアログボックスが開きます。
2. ルールの [**種類**] を選択します。タイプは [**Direct**] (プロキシなし)、[**Socks4**]、[**Socks5**]、及び [**Http**] から選択できます。



図A.12 プロキシ設定の定義

直接接続以外の場合は、プロキシ [**サーバ**] のアドレスと [**ポート**] を入力します。

また、プロキシ・ルールを [**すべて**] の接続に適用するか、指定した [**ネットワーク**] への接続にのみ適用するかを選択します。[**ネットワーク**] フィールドには、カンマ (,) で区切られた条件を 1 つ以上入力できます。条件では IP アドレスまたは DNS 名を指定できます。

IPアドレス/ポートの条件には、アドレス・パターンとオプションのポート範囲 (`ip_pattern[:port_range]`) を指定できます。

`ip_pattern` は、以下のいずれかの形式にすることができます。

- 単一の IP アドレス `x.x.x.x`
- `x.x.x.x-y.y.y.y` の形式による IP アドレス範囲
- `x.x.x.x/y` の形式による IP サブネットワーク・マスク

DNS名の条件は、ホスト名 ("*" と "?" を含む正規表現) とポート範囲 (`name_pattern[:port_range]`) から構成されます。

[**OK**] をクリックします。

プロキシ・ルールを編集するには、リストからルールを選択し、[**編集**] をクリックします。

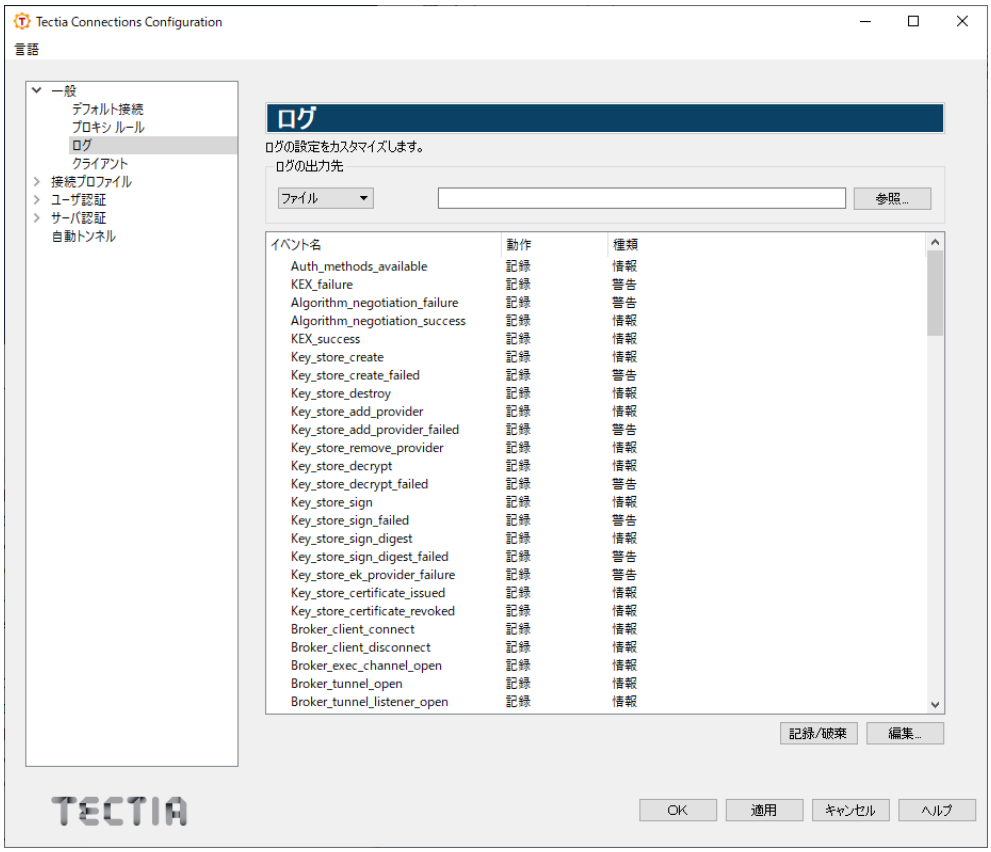
プロキシ・ルールを削除するには、リストからルールを選択し、[**削除**] をクリックします。

ルールは上から下に読まれます。ルールの順序を変更するには、矢印ボタンを使用します。

これらの一般的なプロキシ・ルールを接続プロファイルで使用するには、プロファイル設定で使用するように選択する必要があります。「プロキシ設定の定義」 を参照してください。

ログ設定の定義

[**ログ**] ページでは、ログを有効にし、イベント・ログに記録される情報をカスタマイズできます。デフォルトでは、ログは無効になっています。



図A.13 ログの設定

Tectia Client の内部イベントのログを有効にするには、ログの保存方法を選択します。[**ログの出力先**] フィールドで、以下の手順に従ってください。

- [**ファイル**] を選択すると、右のフィールドに指定した名前のファイルにログ・データが保存されます。正確なファイル名を入力するか、または既存のファイルを参照します。
- [**イベント ログ**] を選択すると、Tectia Client のデータがホストのイベント・ログに保存されます。

各プログラム内部イベントには、関連する [**動作**] と [**種類**] があります。これらは適切なデフォルト値があり、明示的なログ設定が行われない場合に使用されます。

動作は [**記録**] または [**破棄**] のいずれかです。

イベントのタイプは以下のいずれかです。

- **情報**
- **警告**
- **エラー**
- **セキュリティ成功**

- **セキュリティ失敗**

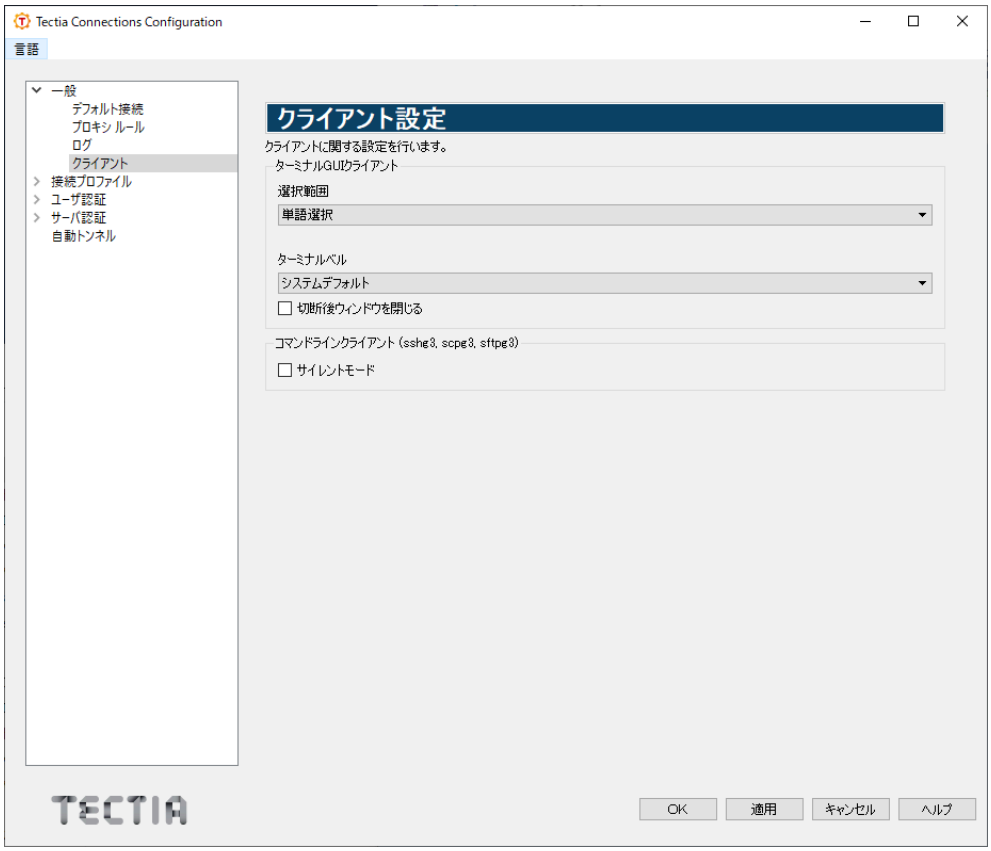
ログ・イベントについては、付録E を参照してください。

イベントをログに記録するかどうかを変更するには、リストからイベントを選択し、[**記録/破棄**] をクリックします。SHIFT キーまたは CTRL キーを押しながらクリックすると、複数のイベントを選択できます。

イベントの動作と種類をカスタマイズするには、リストからイベントを選択し、[**編集**] をクリックします。SHIFT キーまたは CTRL キーを押しながらクリックすると、複数のイベントを選択できます。[**監査イベントの編集**] ダイアログボックスが開きます。イベントの [**動作**] ([記録] または [破棄]) 及び [**種類**] ([情報、警告、エラー、セキュリティ成功] または [セキュリティ失敗]) を選択し、[**OK**] をクリックします。

クライアント設定の定義

[**クライアント設定**] ページでは、クライアントに関連する設定を定義できます。



図A.14 クライアントの設定

GUIクライアント

[**選択範囲**] オプションでは、ダブルクリックでテキストを選択したときの Tectia SSH ターミナル GUI の動作を定義します。以下のオプションがあります。

- **単語選択** (デフォルト) - 一度に 1 単語が選択され、スペースとすべての句読点は区切り文字として使用されます。
- **パス選択** - スペースに挟まれた文字列が選択されます。つまり、選択範囲が \/.-_ の文字まで広がります。たとえば、ファイルへのパス内の任意の場所をダブルクリックすると、そのパス全体が選択されます。

Tectia ターミナルが接続先サーバからの音声通知を繰り返すかどうかを定義するには、[**ターミナルベル**] オプションを使用します。このオプションは Unix サーバとの接続にのみ適用されます。以下のオプションがあります。

- **システムデフォルト** (デフォルト) - 接続先サーバのシステムで定義されたデフォルトのアラートを鳴らします。
- [**PCスピーカー**] - ユーザの PC スピーカからビープ音を鳴らします。
- **無効** - すべての可聴通知をミュートにします。

[**切断後ウィンドウを閉じる**] オプションを選択すると、CTRL+D キーを押してサーバ・セッションから切断するときに、Tectia SSHターミナル GUI のウィンドウも閉じるように定義できます。デフォルトでは、ターミナルは開いたままで、サーバ接続のみが閉じられます。

コマンドラインクライアント

[**サイレントモード**] の設定では、コマンドライン・クライアントが警告、エラー・メッセージ、及び認証成功メッセージを抑制するかどうかを定義します。この設定はコマンドライン・ツール **scpg3**、**sshg3**、及び **sftpg3** に影響します。

A.1.3. 接続プロファイルの定義

[**接続プロファイル**] では、接続する Secure Shell サーバごとに個別の接続設定を行えます。また、同じサーバに対して、たとえば異なるユーザ・アカウントで複数のプロファイルを設定することもできます。

リモート・サーバへの接続を開くには、[**接続をテスト**] をクリックします。サーバのホスト鍵を取得するために、一度サーバに接続する必要があります。Tectia Client により、受け取った鍵の確認が求められます。できればサーバ管理者に、受け取った鍵が有効であることを確認し、有効な鍵を保存します。この後、ローカルに保存された鍵の情報が自動的に認証処理に使用されます。

- 接続プロファイルを追加するには、[**接続プロファイル**] ページで [**プロファイル追加**] をクリックします。プロファイルの名前を入力し、[**OK**] をクリックします。デフォルトでは、プロファイル名はサーバのホスト名としても使用されます。

  新しく作成された接続プロファイルは、[**一般**] → [**デフォルト**] ページで定義された認証、暗号、MAC、鍵交換、及び詳細なサーバ設定のデフォルト値を継承します (「デフォルト

接続設定の定義」)。これらの値は、プロファイル固有のタブ付きページでカスタマイズできます。

「接続設定の定義」、「認証の定義」、「暗号の定義」、「MAC の定義」、「鍵交換の定義」、「サーバ接続の定義」、「プロキシ設定の定義」、「トンネリングの定義」、「ウィンドウ設定の定義」、「色の設定の定義」、「ターミナル設定の定義」、「ファイル転送設定の定義」、及び 「お気に入りフォルダの定義」 の説明に従い、タブ付きビューでプロファイル設定を定義します。

- 接続プロファイルは、接続先のサーバごとにフォルダにまとめることができます。接続プロファイル用のフォルダを追加するには、[**接続プロファイル**] ページで [**プロファイル追加**] をクリックします。フォルダの名前を入力し、[**OK**] をクリックします。フォルダを選択して [**プロファイル追加**] をクリックすると、フォルダに接続プロファイルが追加されます。プロファイルはフォルダ内に作成されます。

- プロファイルを別のプロファイル・フォルダに移動するには、リストからプロファイルを選択し、[**移動**] をクリックします。ドロップダウン・リストから、プロファイルの移動先フォルダを選択し、[**OK**] をクリックします。

- 接続プロファイルまたはプロファイル・フォルダの名前を変更するには、[**接続プロファイル**] のプロファイルまたはフォルダ名を右クリックし、[**名前の変更**] をクリックします。新しい名前を入力し、 **Enter** キーを押して、 [**OK**] または [**適用**] をクリックします。

- 接続プロファイルまたはプロファイル・フォルダを削除するには、プロファイルまたはフォルダを選択し、[**削除**] をクリックします。確認を求められます。 [**OK**] をクリックして削除を実行します。

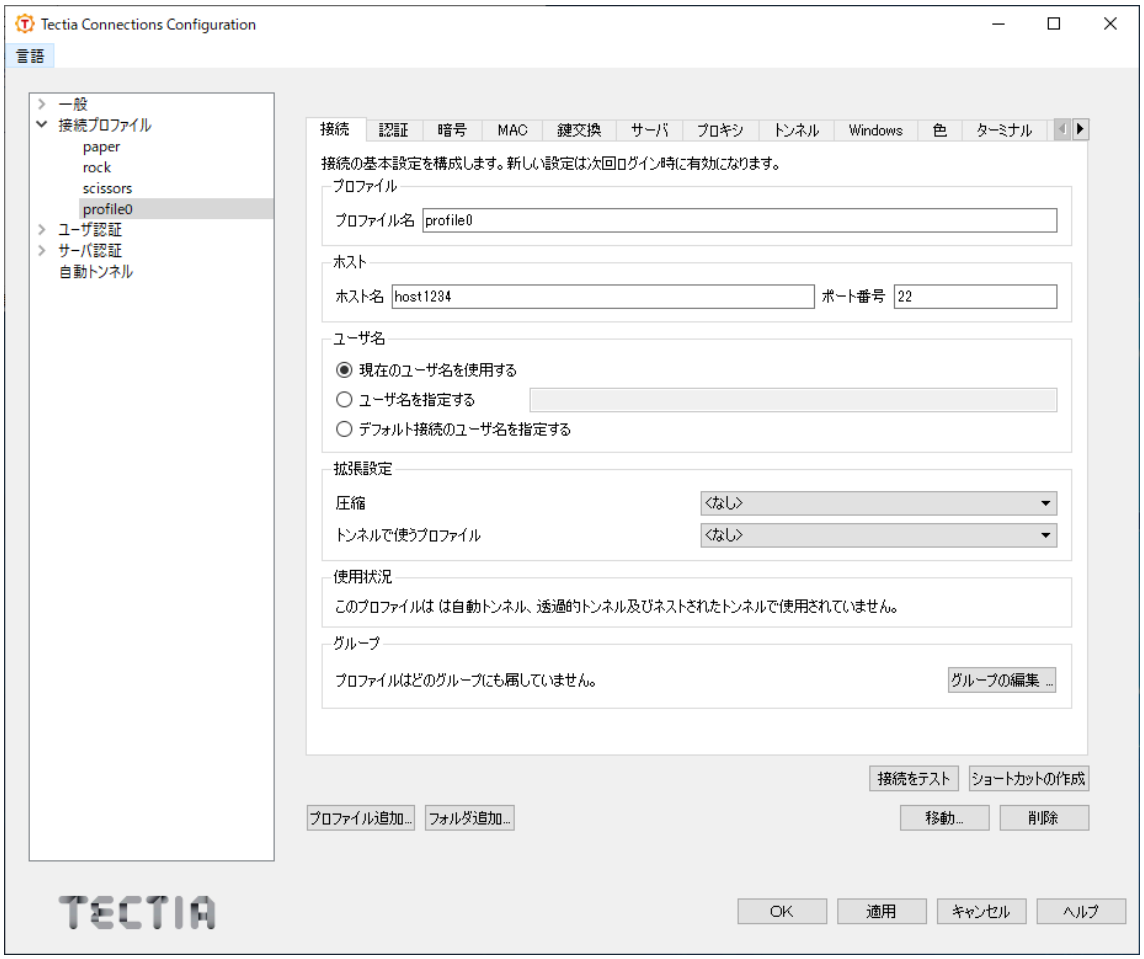
  プロファイル・フォルダを削除すると、その中にあるすべてのプロファイルも削除されることに注意してください。

-

-

接続設定の定義

[**接続**] タブでは、接続で使用するプロトコル設定を定義できます。変更された接続設定は次回のログイン時に有効になります。



図A.15 接続プロファイルの設定

ホスト名

プロファイルを使って接続するリモート・ホスト・コンピュータのホスト名または IP アドレスを指定します。

ポート番号

Secure Shell サーバのリスナ・ポートを定義します。デフォルトの SSH ポート番号は 22 です。リモート・サーバが別のポートを使用していることが分かっている場合は、[**ポート番号**] フィールドにその番号を入力します。

注意

Secure Shell サーバ・プログラムは、リモート・ホスト・コンピュータ上の指定されたポートをリッスンしている必要があり、そうでない場合は接続試行に失敗します。リモート・ホスト・コンピュータがリッスンしているポートが不明な場合は、リモート・ホストのシステム管理者に問い合わせてください。

ユーザ名

現在ログインしている Windows のユーザ名で常に接続する場合は、[**現在の Windows ユーザ名を使用する**] を選択します。この設定は、ユーザ名として %USERNAME% (パーセント記号に注意) を定義するのと同じような働きをします。 %USERNAME% は、環境変数から実際のユーザ名を読み取ります。

リモート・ホスト・コンピュータへの接続時にこのプロファイルが使用するユーザ名を定義する場合は、[**ユーザ名を指定する**] を選択してユーザ名を入力します。

接続時に毎回手動でユーザ名を入力する場合は、[**ユーザにユーザ名の入力を求める**] を選択します。

[**一般**] - [**デフォルト接続**] の設定で定義された汎用的なユーザ名を適用する場合は、[**デフォルト接続のユーザ名を指定する**] を選択します。

詳細設定

[**圧縮**] では、使用する圧縮設定をドロップダウン・メニューから選択します。有効な選択肢は [**zlib**] と [**なし**] です。圧縮はデフォルトでは無効です。

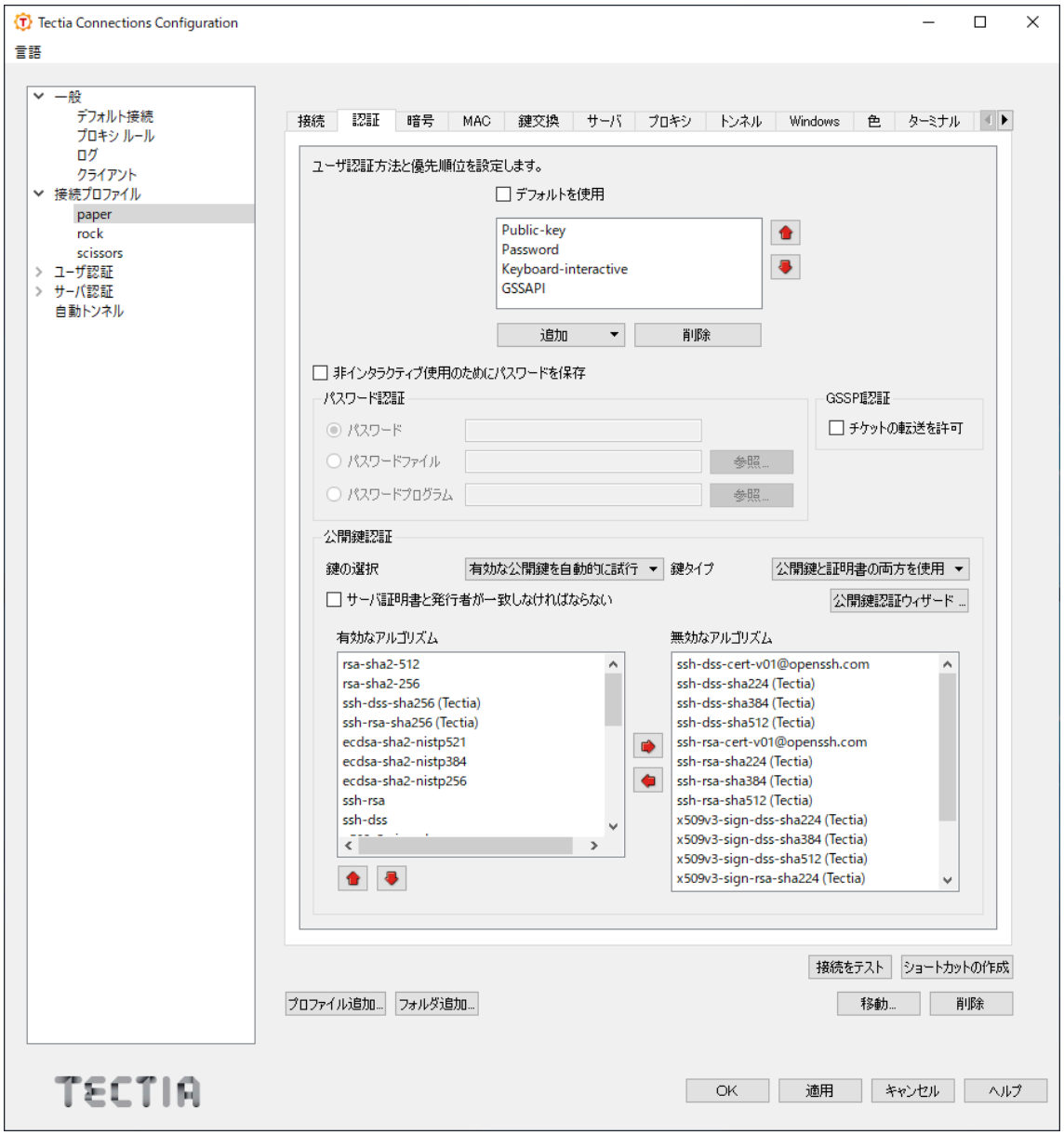
[**トンネルで使うプロファイル**] でドロップダウン・リストを使用して、ネストされたトンネルを作成するためのプロファイルを選択します。最初のトンネルは、現在の接続プロファイルで定義されているサーバに作成され、そこから、[**トンネルで使うプロファイル**] の設定で選択されたプロファイルで定義されているホストに 2 番目のトンネルが作成されます。デフォルトでは、トンネリングは無効になっています。

使用状況

このフィールドには、定義されているプロファイルがどこで使用されているのか情報が表示されます。

認証の定義

[**認証**] タブでは、プロファイルのユーザ認証方法を定義できます。



図A.16 プロファイルの認証方法の設定

1. [**デフォルト接続**] ページ（「認証の定義」）で定義されている認証方法を使用するには、[**デフォルトを使用**] チェックボックスを選択します。認証方法のカスタム・リストを定義する場合は、このチェックボックスを選択解除します。

   新しい認証方法をリストに追加するには、 [**追加**] をクリックし、ドロップダウン・メニューから方法を選択します。

   認証方法を削除するには、リストから方法を選択し、[**削除**] をクリックします。

   矢印ボタンを使用して、認証方法の優先順位を並べ替えます。Secure Shell サーバで許可されている最初の方法が使用されます。ログインするために、サーバが複数の認証方法に成功することを要求することもあります。

ユーザ認証には、以下の方法があります。

- **Public-key:** 公開鍵認証を使用します。A.1.4 も参照してください。
- **Password:** 認証にパスワードを使用します。
- **Keyboard-interactive:** キーボード・インタラクティブは、Secure Shell クライアントが RSA SecurID や PAM など、複数の異なるタイプの認証方法をサポートできるように設計されています。 キーボード・インタラクティブの詳細については、4.8 を参照してください。
- **GSSAPI:** GSSAPI (ジェネリック・セキュリティ・サービス・アプリケーション・プログラミング・インターフェイス) は、異なるセキュリティ対策を 1 つのインターフェイスで利用できるようにした、共通のセキュリティ・サービス・インターフェイスです。GSSAPI の詳細については、4.9 を参照してください。

2. 非対話型の接続でプロファイルを使用する場合は、 [**パスワード認証**] フィールドで、プロファイルにパスワードを保存することを選択できます。

   実際のパスワード文字列を入力するには、 [**パスワード**] を選択します。

   パスワードを含むファイルへのパスを入力するには、 [**パスワードファイル**] を選択します。

   パスワードを出力するプログラムまたはスクリプトのパスを入力するには、[**パスワードプログラム**] を選択します。

   警告

   このオプションを使用してパスワードを指定する場合は、意図するユーザ以外は `ssh-broker-config.xml` ファイル、パスワード・ファイル、またはプログラムにアクセスできないようにすることが非常に重要です。

   注意

   コマンドライン・オプションでパスワードを指定すると、この設定を上書きします。

3. Tectia Client で複数の接続にわたる Kerberos チケットの転送を許可するには、[**GSSPI認証**] フィールドにある [**チケットの転送を許可**] チェックボックスを選択します。

4. [**公開鍵認証**] を使用する場合は、使用する鍵タイプや鍵の選択方法も定義できます。

   鍵の選択 では、ユーザの公開鍵をサーバに提示する際に接続ブローカーが使用するポリシーを定義します。ドロップダウン・リストからモードを選択します。以下のオプションがあります。

   - **有効な公開鍵を自動的に試行**(デフォルト)。このポリシーでは、クライアントは以下の順に鍵を試行します。

a. 公開鍵が使用可能で、秘密鍵にパスフレーズがない鍵 (ユーザの操作なし)

b. 公開鍵が使用可能であるが、秘密鍵にパスフレーズが設定されている鍵 (鍵がサーバに受け入れられるという前提で、パスフレーズの提供が必要)

c. 残りの鍵。つまり、秘密鍵だけでなく、公開鍵にもパスフレーズが必要な鍵。

- **鍵選択プロンプトから選択** - このポリシーでは、使用可能な鍵のリストから鍵を選択するよう接続ブローカーから求められます。選択した鍵による認証に失敗した場合、再度別の鍵を選択するようクライアントから求められます。

鍵タイプ では、公開鍵認証の際に、プレーンな公開鍵のみを試すか、証明書のみを試すかを定義します。ドロップダウン・リストから鍵の種類を選択します。デフォルトでは、プレーンな公開鍵と証明書の両方を試します。

[**サーバ証明書と発行者が一致しなければならない**] チェックボックスを選択すると、ユーザに提示されるリストに含まれるユーザ証明書を接続ブローカーがフィルタリングします。クライアント側のユーザ証明書は、その発行者名が、サーバが要求したまたは受け入れた証明書発行者と比較されることでフィルタリングされます。デフォルトでは、フィルタリングは行われません。このオプションは、テスト用ロールと管理者ロールなど、ユーザが同じサーバに対して異なるアクセス権を持つ複数の証明書を有している場合に便利です。接続ブローカーは、リモート・ホストで適用可能な該当する証明書を選別するので、ユーザは絞り込まれた証明書の中から正しい証明書を選択できます。

公開鍵ペアを生成し、リモート・サーバにアップロードするには、[**公開鍵認証ウィザード**] ボタンをクリックします。手順については、「公開鍵認証ウィザードの使用」 を参照してください。

[**有効なアルゴリズム**] には、ユーザの公開鍵の認証と署名に使用される公開鍵署名アルゴリズムがリストアップされます。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方に設定されているものです。アルゴリズムの順序は、上下の矢印ボタンを使用して変更できます。アルゴリズムを [**無効なアルゴリズム**] リストに移動させるには、そのアルゴリズムを選択して右矢印ボタンをクリックします。

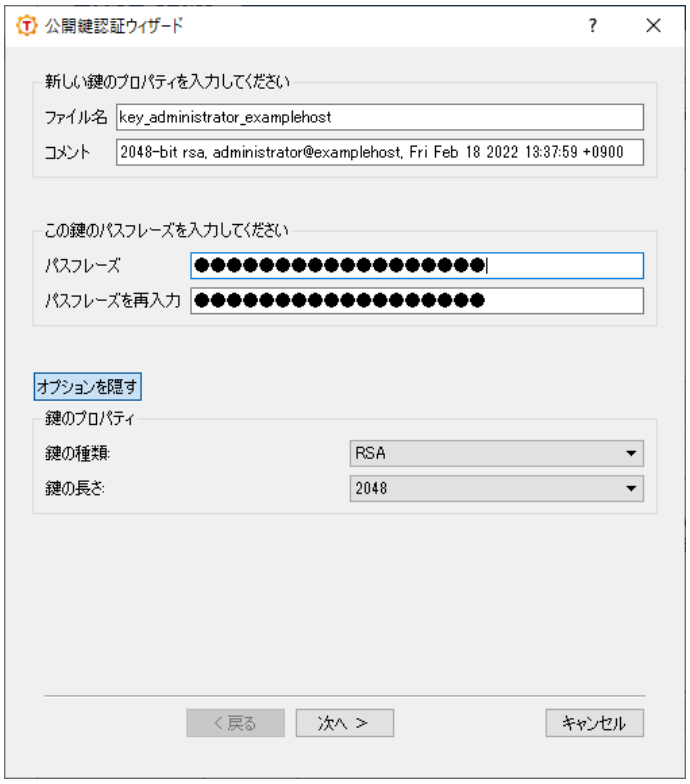
5. [OK] をクリックして、接続プロファイルを保存します。

公開鍵認証ウィザードの使用

Windows では、Tectia の [**公開鍵認証ウィザード**] を使用して、公開鍵ペアの生成とアップロードを行えます。このウィザードは、秘密鍵と公開鍵の 2 つの鍵ファイルを生成します。

新しい秘密鍵と公開鍵は、ローカル・コンピュータの `%APPDATA%\SSH\UserKeys` ディレクトリに保存されます。秘密鍵ファイルにはファイル拡張子がありません。公開鍵は秘密鍵と同じベース・ファイル名ですが、ファイル拡張子は `.pub` になります。

[**ユーザ認証**] にある [**鍵と証明書**] ページを選択し、[**鍵生成**] をクリックして、[公開鍵認証ウィザード] を開始します。



図A.17 [公開鍵認証ウィザード]

鍵のプロパティと、鍵ペアを保護するために必要なパスフレーズを定義します。鍵を使用して自分自身を認証する際には、常にこのパスフレーズの入力が要求されます。

[ファイル名]

鍵ファイルに付ける一意の名前を入力します。Tectia Client では、ユーザ名とホスト名を組み合わせた名前が提案されます。

[コメント]

このフィールドには、鍵ペアについて説明する、短いコメントを記入します。たとえば、その鍵が使用される接続について説明できます。このフィールドは必須ではありませんが、後で鍵を特定するのに役立ちます。

[パスフレーズ]

鍵を使用するときに入力しなければならないフレーズを入力します。このパスフレーズには、パスワードに似た機能があり、秘密鍵をある程度保護します。

注意

FIPS モードでは、暗号化されていない秘密鍵を FIPS モジュールからエクスポートすることが FIPS 規定において禁じられているため、パスフレーズなしでユーザ鍵を生成することはできません。

推測するのが難しいフレーズを入力します。理想はアルファベットと数字の両方を使用した 20 文字以上です。句読文字も使用できます。 パスフレーズも秘密鍵もネットワークに送信されることはありませんが、ローカルでアクセスできる場合は辞書攻撃が使用できます。使いやすさのためには、パスフレーズを空白にするのではなく認証エージェントの使用を推奨します。ssh-groker-g3 はデフォルトで認証エージェントとして機能します。

パスフレーズはしっかりと記憶し、書き留めないようにしてください。

ユーザが操作できない接続では、空のパスフレーズを使用することもできます。

[パスフレーズを再入力]

パスフレーズを再入力します。これにより、入力ミスがないことが確認できます。

生成する鍵の種類と長さを、デフォルトとは異なるものに定義するには、[**詳細オプション**] をクリックします。デフォルトでは、Tectia Client は 3072 ビットの RSA 鍵のペアを生成します。

[**鍵のプロパティ**] フィールドでは、以下が選択できます。

[鍵の種類]

生成する鍵の種類を選択します。利用可能なオプションは Ed25519、 RSA、 ECDSA、 及び DSA です。

注意

FIPS モードでは (FIPS 186-5 に準拠し) RSA、 ECDSA および Ed25519 がサポートされます。DSA は非推奨です。

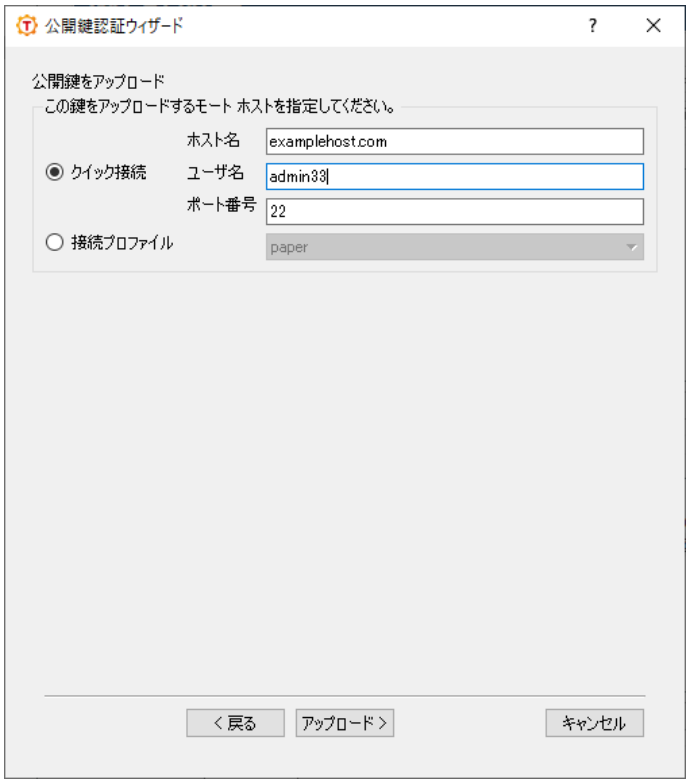
[鍵の長さ]

生成する鍵の長さ (複雑さ) を選択します。以下のオプションから選択できます。

- DSA/RSA 鍵: 2048、 3072、 4096、 5120、 6144、 7168、 8192 ビット
- ECDSA 鍵: 256、 384、 521 ビット
- Ed25519 鍵: 256 ビット

同じ種類の鍵でも、大きいほどよりセキュアになりますが、それとともに生成時間が長くなります。256 ビットの ECDSA 鍵と 3072 ビットの DSA または RSA 鍵は同等のセキュリティを提供します。

新しい鍵が生成されるとすぐに、ウィザードはリモート・サーバへの鍵のアップロードに進みます。既存の鍵をリモート・サーバにアップロードする場合は、[鍵と証明書] ビューで鍵ファイルを選択し、[**アップロード**] をクリックします。どちらの場合も、以下のダイアログが表示されます。



図A.18 鍵のアップロード

ウィザードの [**公開鍵をアップロード**] ビューで、鍵をアップロードするリモート・ホストを定義します。

[クイック接続]

このオプションを選択すると、リモート [**ホスト名**] と [**ユーザ名**] を定義できます。デフォルトの Secure Shell ポートは 22 です。

[接続プロファイル]

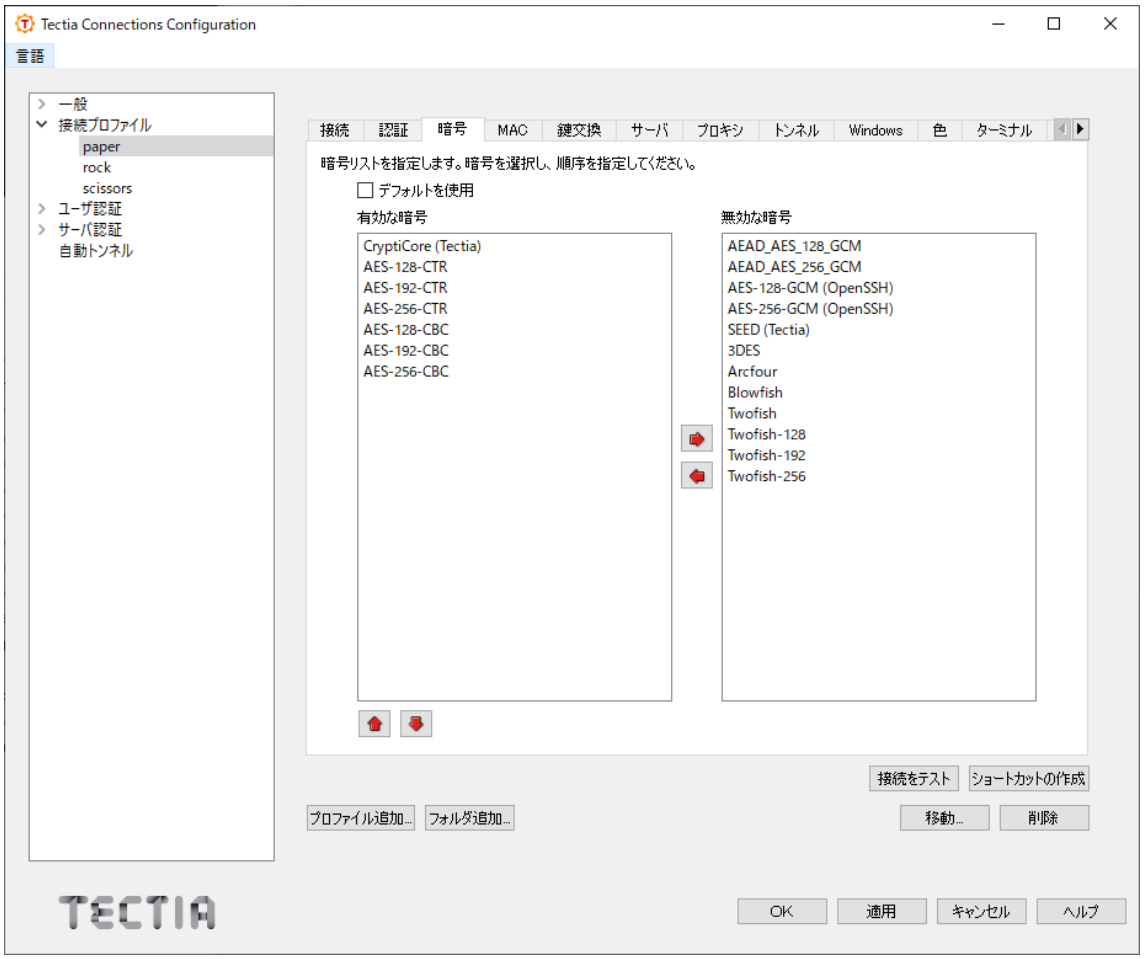
目的のリモート・ホストとユーザ名を規定する [**接続プロファイル**] をドロップダウン・リストから選択します。

[**アップロード**] をクリックして、選択したサーバに鍵をアップロードします。すでにリモート・サーバ・ホストに接続されている場合は、鍵のアップロードがすぐに開始されます。接続されていない場合は、サーバへの認証が求められます (デフォルトではパスワード)。

公開鍵はデフォルトのユーザ・ホーム・ディレクトリ (Windows では `%USERPROFILE%\.ssh2`、Unix では `$HOME/.ssh2`) にアップロードされます。

暗号の定義

[**暗号**] タブでは、プロファイルに使用する暗号化アルゴリズムを定義できます。



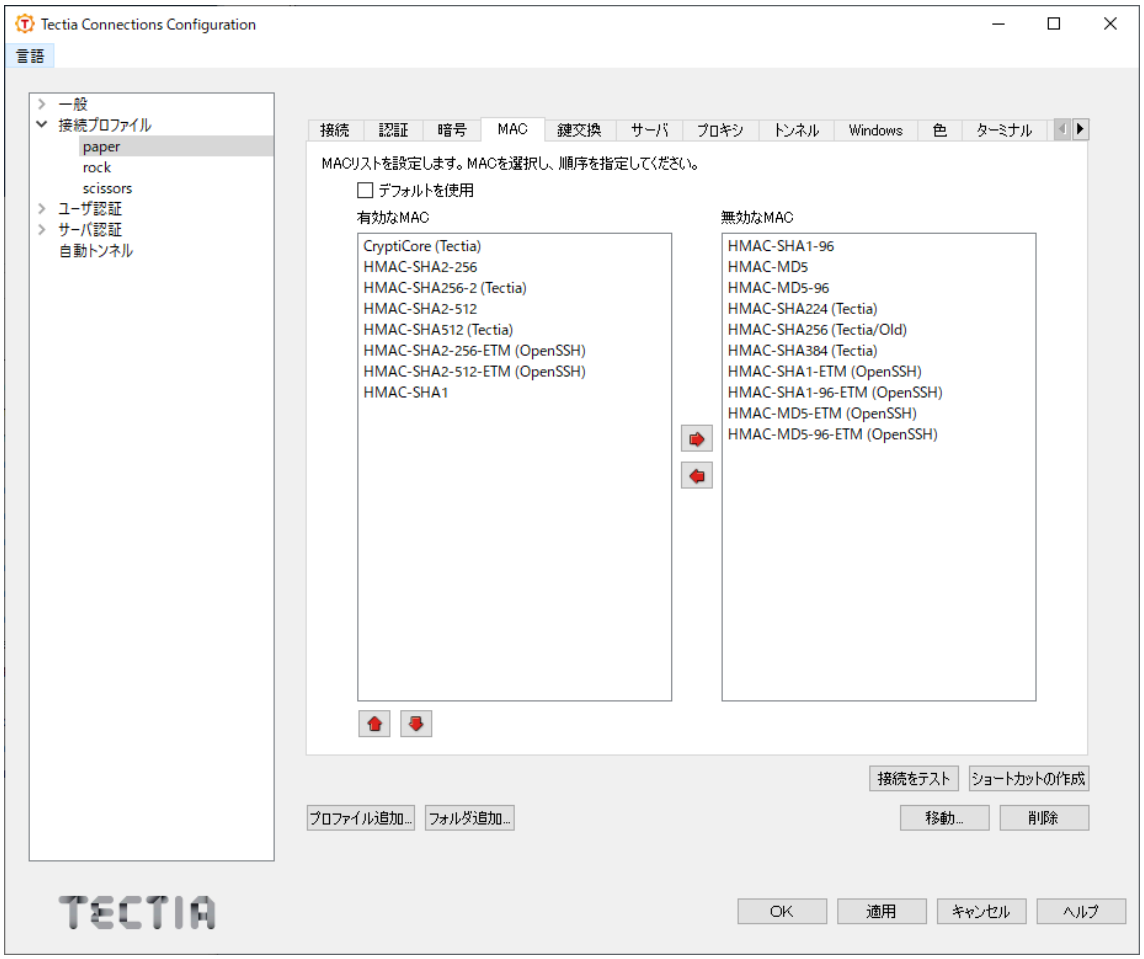
図A.19 プロファイルの暗号リストの定義

[**デフォルト接続**] ページ（「暗号の定義」）で定義されているアルゴリズムを使用するには、[**デフォルトを使用**] チェックボックスを選択します。暗号リストを定義する場合は、矢印ボタンを使用します。暗号は、指定された順に試行されます。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

MAC の定義

[**MAC**] タブでは、プロファイルに使用するメッセージ完全性アルゴリズムを設定できます。



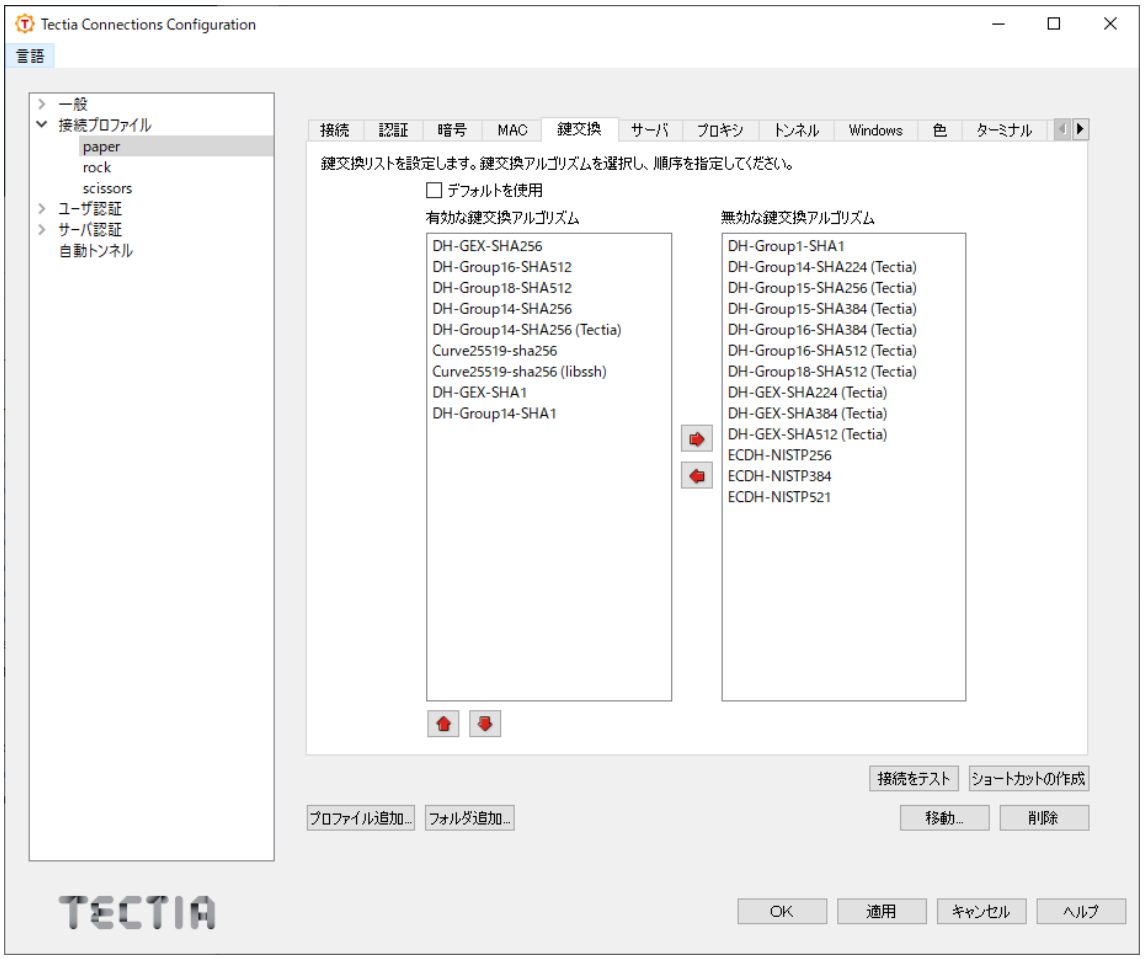
図A.20 プロファイルの MAC リストの定義

[**デフォルト接続**] ページ (「MAC の定義」) で定義されているアルゴリズムを使用するには、[**デフォルトを使用**] チェックボックスを選択します。MAC リストを定義する場合は、矢印ボタンを使用します。MAC は、指定された順に試行されます。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

鍵交換の定義

[**鍵交換**] タブでは、プロファイルに使用する鍵交換方法を設定できます。



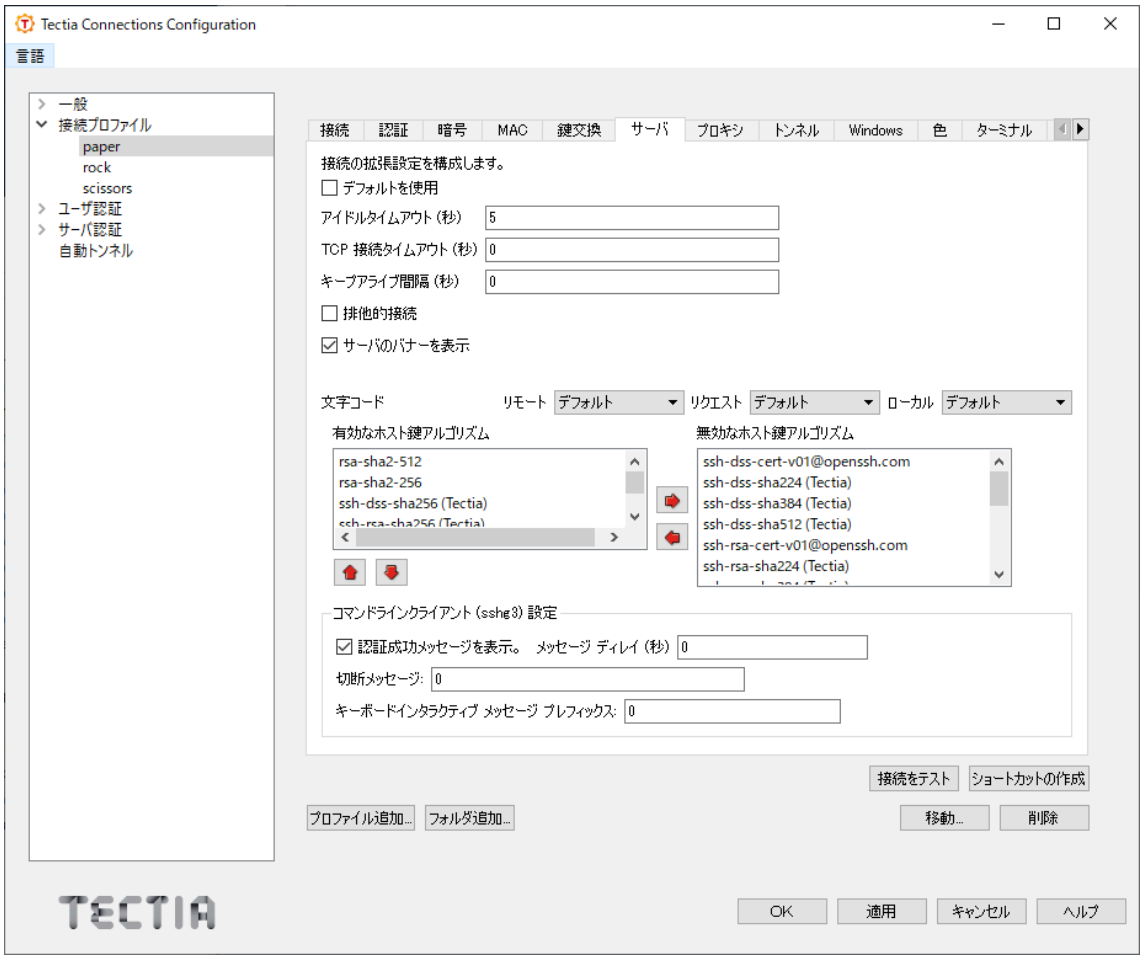
図A.21 プロファイルの鍵交換リストの定義

[**デフォルト接続**] ページ (「鍵交換の定義」) で定義されている方法を使用するには、[**デフォルトを使用**] チェックボックスを選択します。鍵交換リストを定義する場合は、矢印ボタンを使用します。鍵交換は、指定された順に試行されます。

Tectia 独自のアルゴリズムには **(Tectia)** の記載があり、Tectia 製品でのみ動作します。それらは、接続ブローカーの設定ファイルの `@ssh.com` で終わるアルゴリズムに対応しています。

サーバ接続の定義

[**サーバ**] タブでは、プロファイルの詳細なサーバ接続設定を定義できます。



図A.22 プロファイルのサーバ接続設定の定義

デフォルトを使用

サーバ接続設定に、[**デフォルト接続**] ページ (「サーバ接続の定義」) で定義されている値を使用するには、チェックボックスを選択します。

アイドルタイムアウト

接続を自動的に閉じるまでにどのくらいの長さのアイドル時間(すべての接続チャネルが閉じた後)が許可されるのか を指定します。デフォルトは 5 秒です。長い時間を設定すると、セッション (Tectia SSHターミナル GUI など) を閉じた後でもサーバへの接続を維持できます。この時間の間は、再認証を行わずにサーバとの新しいセッションを開始できます。時間を 0 (ゼロ) に設定すると、サーバとの最後のチャネルが閉じられたときに、接続がすぐに終了します。

TCP 接続タイムアウト

Secure Shell サーバへの TCP 接続を試行する時間を指定します。タイムアウトを秒単位で定義します。定義した時間が経過してもリモート・サーバがダウンしているか、または到達できない場合、TCP 接続が解放されます。値を 0 (ゼロ) に設定すると、システムのデフォルトの TCP タイムアウトが使用されます。

キープアライブ間隔

Secure Shell サーバにキープアライブ・メッセージを送信する間隔 (秒単位) を指定します。デフォルトは 0 で、この場合、キープアライブ・メッセージは送信されません。

排他的接続

現在開いている接続を再利用するのではなく、プロファイルが常に新しい接続を開くようにする場合は、このチェックボックスを選択します。

サーバのバナーを表示

ログイン前にサーバ・バナー・メッセージ・ファイル (存在する場合) を表示させる場合は、このチェックボックスを選択します。

有効なホスト鍵アルゴリズム

このリストには、ホスト鍵または証明書を用いたサーバ認証に使用されるホスト鍵署名アルゴリズムが表示されます。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方の設定ファイルで定義されているものです。このようにすることで、SHA-2 のような特定のアルゴリズムのみを使用するようにサーバから強制できます。

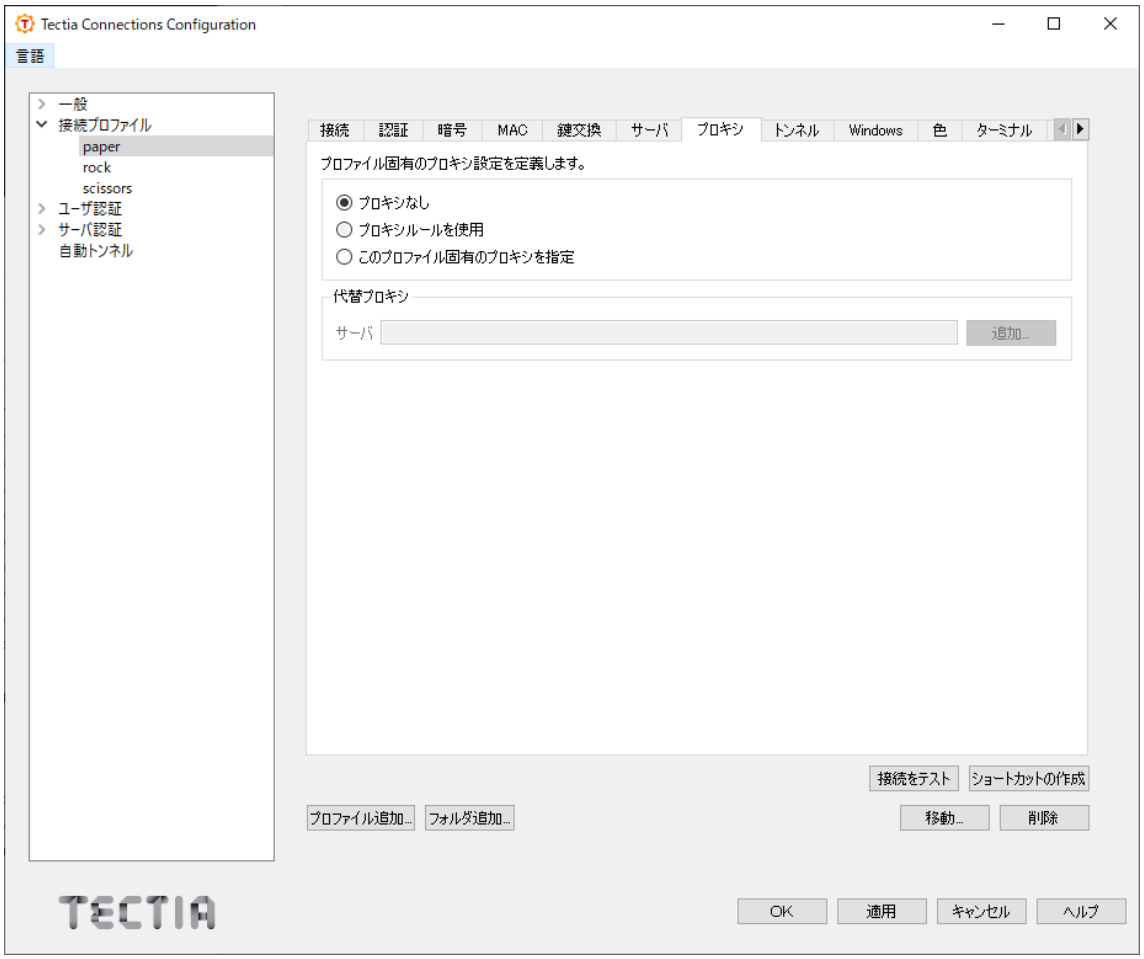
ホスト鍵アルゴリズムは、指定された順に試行されます。ただし、1 つ例外があり、サーバのホスト鍵がすでにクライアントのホスト鍵ストアに存在する場合は、そのアルゴリズムが優先されます。アルゴリズムの順序は、上下の矢印ボタンを使用して変更できます。

無効なホスト鍵アルゴリズム

このリストに表示されているホスト鍵アルゴリズムは、サーバ認証には使用されません。ホスト鍵アルゴリズムを無効にするには、[**有効なホスト鍵アルゴリズム**] リストでそのアルゴリズムを選択し、右矢印ボタンをクリックします。

プロキシ設定の定義

[**プロキシ**] タブでは、プロファイルのプロキシ設定を選択できます。



図A.23 プロファイルのプロキシ設定の定義

プロキシなし

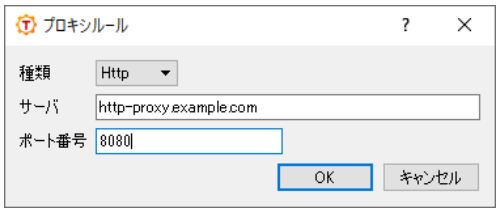
プロキシを使用しない場合は、このオプションを選択します。

プロキシルールを使用

[一般] の[**プロキシルール**] ページ（「プロキシ・ルールの定義」）で定義されたプロキシ・ルールを使用するには、このオプションを選択します。

このプロファイル固有のプロキシを指定

このプロファイルに新しいプロキシ定義を追加するには、[**追加**] をクリックします。



図A.24 プロファイルの代替プロキシの定義

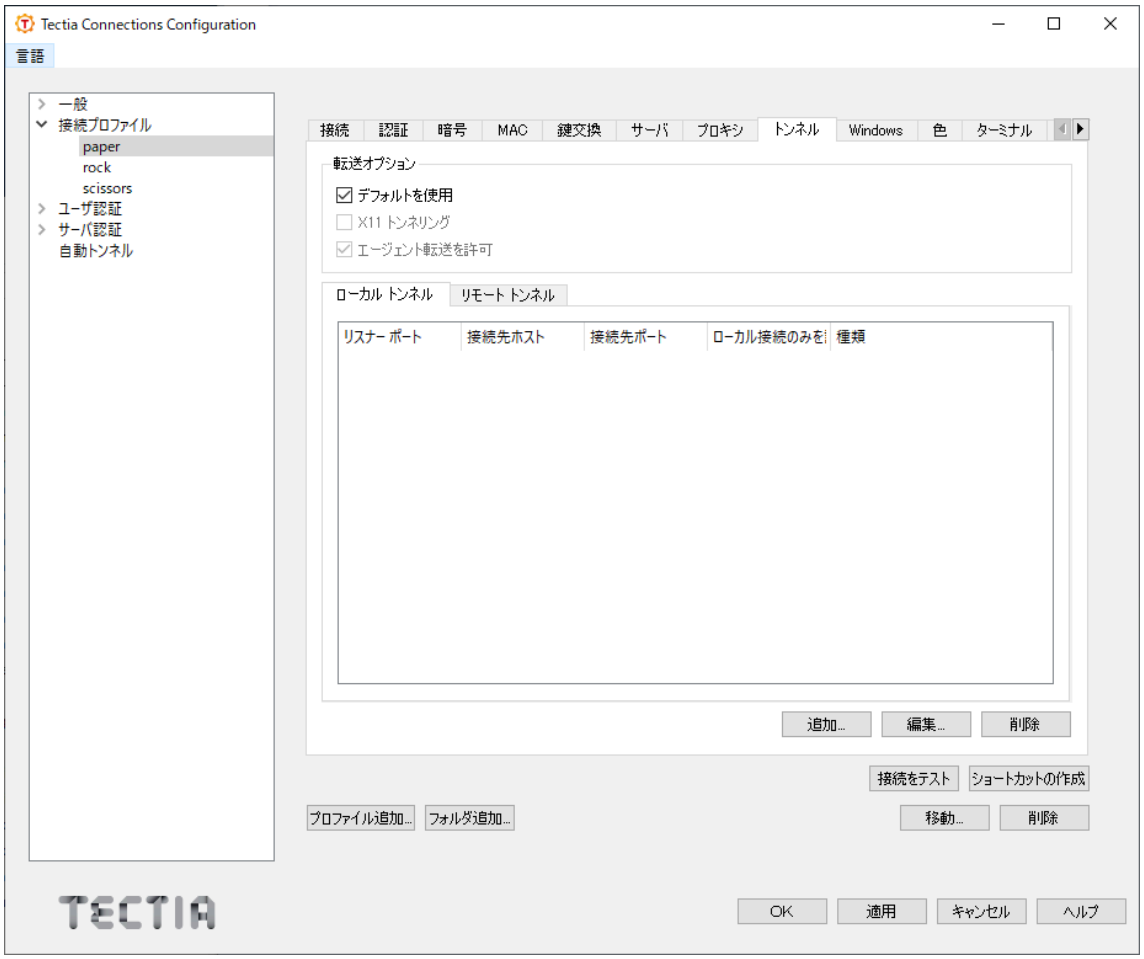
ルールの [**種類**] を選択します。タイプは [**Direct**]、[**Socks4**]、[**Socks5**]、及び [**Http**] から選択できます。

直接接続以外の場合は、プロキシ [**サーバ**] のアドレスと [**ポート番号**] を入力します。

トンネリングの定義

トンネリング (ポート転送) とは、そのままではセキュアではない TCP トラフィックを、暗号化された Secure Shell 接続経由で転送する方法です。たとえば POP3 や SMTP、HTTP 接続など、そのままではセキュアではない接続をセキュアにすることができます。

接続プロファイルのトンネリング設定は [**トンネル**] タブで行います。変更されたトンネリング設定は、次回のログイン時に有効になります。



図A.25 プロファイルによるトンネリングの定義

注意

トンネルを使用するクライアント/サーバ・アプリケーションは、暗号化トンネルを使用しない場合と同様に、自身の認証手順を実行します (存在する場合)。

転送オプション

X11 転送及び/またはエージェント転送を有効にするか、一般的なデフォルトの転送設定をプロファイルに適用するかどうかを、接続プロファイルごとに個別に定義できます。

デフォルトを使用

プロファイルが、[**デフォルト**] - [**トンネル**] タブ (「デフォルトのトンネリング設定の定義」) で定義されている X11 転送及びエージェント転送のデフォルト設定に従うようにするには、このオプションを選択します。

X11 トンネリング

この接続プロファイルで　X11　転送を許可するには、このチェックボックスを選択します。

Tectia Client は、リモート・ホスト・コンピュータから、ローカル・コンピュータ上で動作する X Windows サーバへの X11 グラフィック接続をセキュアにトンネル (転送) できます。

注意

Windows コンピュータ上で X エミュレータ (eXceed や Reflection X など) がパッシブ・モードで動作していることが X11 トンネリングの前提条件です。

X11 トラフィックをトンネル (転送) するには、以下の手順を実行してください。

1. Windows にX サーバ (X エミュレーション) プログラム (eXceed、Reflection X など) をインストールします。
2. Tectia Client を起動します。
3. [接続プロファイル] ページの [**トンネル**] タブを選択し、[**X11 トンネリング**] チェックボックスが選択されていることを確認します。
4. Tectia Client の設定を保存します。
5. Tectia Client を再起動し、リモート・ホストにログインします。
6. X サーバ (X エミュレーション) プログラムを起動します。
7. トンネリングをテストするために、Tectia Client から xterm または xclock を実行します。

詳細については、6.3 を参照してください。

エージェント転送を許可

この接続プロファイルでクライアント側のエージェント転送を許可するには、このチェックボックスを選択します。

エージェント転送では、ユーザがそれぞれのサーバに個別に認証することなく、Secure Shell の接続と公開-鍵認証の情報が 1 つのサーバから別のサーバに転送されます。

詳細については、6.4 を参照してください。

ローカル・トンネル

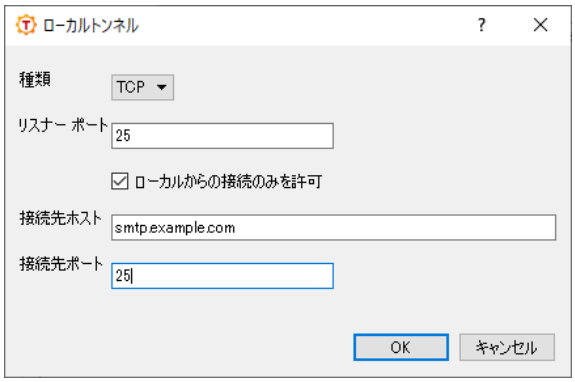
アプリケーションのトンネリングに定義できるトンネルには、ローカル (送信) トンネルとリモート (受信) トンネルの 2 種類があります。

ローカル・トンネルは、ローカル・コンピュータが指定されたローカル・ポートから接続先のリモート・ホスト・コンピュータの指定されたポートへ転送する TCP 接続を保護します。リモート・ホスト・コンピュータを越えて接続を転送することも可能ですが、その場合の接続は Tectia Client と Tectia Server の間でのみ暗号化されます。

リモート・トンネルは、リモート・ホストが指定されたリモート・ポートからローカル・コンピュータの指定されたポートへの転送する TCP 接続を保護します。

ローカル・トンネルの定義を編集するには、[**ローカル トンネル**] タブをクリックします。

新しいローカル・トンネルを追加するには、[**追加**] をクリックします。[**ローカルトンネル**] ダイアログボックスが開きます。



図A.26 ローカル・トンネルの定義

以下のフィールドは、ローカル・トンネルを定義するために使用します。

- **種類:** トンネルの種類をドロップダウン・リストから選択します。TCP または FTP を選択できます。FTP 接続をトンネルする場合は、トンネルの種類を FTP に設定します。その他のプロトコルの場合は、トンネルの種類を TCP に設定します。

注意

Secure Shell サーバと FTP サーバが異なるコンピュータにある場合、FTP トンネリングは、FTP がパッシブ・モードで実行されるように設定されている場合にのみ機能します。Secure Shell サーバと FTP サーバが同じコンピュータに

ある場合、FTP　がパッシブ・モードとアクティブ・モードのどちらで動作しているかに関係なく、トンネリングは機能します。FTP　トンネリングの詳細については、6.1.2 を参照してください。

- **リスナー　ポート**:　これは、トンネルがリッスンする、またはキャプチャするローカル・ポートの番号です。

  注意

  正常に接続するために、トンネルを作成する対象のプロトコルまたはアプリケーションには固定のポート番号 (IMAP の場合は 143 など) が必要な場合があります。その他のプロトコルまたはアプリケーションについても、オフセット (VNC の場合は 5900 など) を考慮に入れなければならない場合があります。

- **ローカルからの接続のみを許可**:　ローカル接続に限って確立を許可する場合は、このオプションを選択します。この場合、作成されたトンネルは他のコンピュータでは使用できません。デフォルトでは、ローカル接続のみが許可されます。これはほとんどの状況に適した選択です。

  外部からの接続も許可する場合は、それに伴うセキュリティへの影響も十分に検討してください。

- **接続先ホスト**: このフィールドでは、トンネリングの接続先ホストを定義します。デフォルト値は `localhost` です。

  注意

  接続先ホストは　Secure　Shell　サーバによって解決されるため、この場合の `localhost` は接続先の Secure Shell サーバ・ホストを指します。

- **接続先ポート**: 接続先ポートは、接続先ホストで転送される接続に使用されるポートを定義します。

トンネルの定義を編集するには、リストからトンネルを選択し、[**編集**]　をクリックします。[**ローカルトンネル**] ダイアログが開きます。

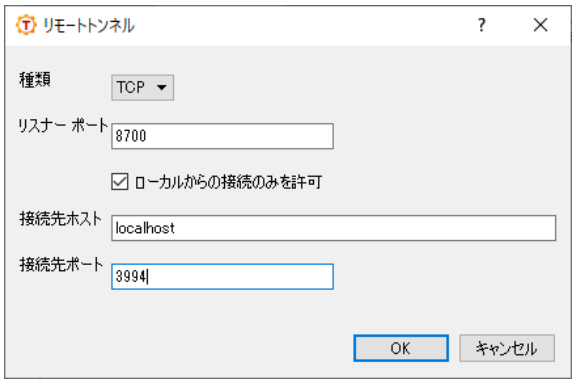
トンネルの定義を削除するには、リストからトンネルを選択し、[**削除**]　をクリックしてトンネルを削除します。選択されているトンネルは、確認ダイアログが表示されることなくすぐに削除されることに注意してください。

ローカル・トンネルの詳細については、6.1 を参照してください。

リモート・トンネル

リモート　(受信)　トンネルは、リモート・ホストが指定されたリモート・ポートからローカル・コンピュータの指定されたポートへ転送する TCP 接続を保護します。

受信トンネルの定義を編集するには、[**リモート　トンネル**] タブをクリックします。[**追加**] をクリックすると、[**リモートトンネル**] ダイアログボックスが開きます。



図A.27 リモート・トンネルの定義

以下のフィールドは、リモート・トンネルを定義するために使用します。

- **種類**: トンネルの種類をドロップダウン・リストから選択します。TCP または FTP を選択できます。FTP トンネリングの詳細については、6.1.2 を参照してください。
- **リスナー ポート**: トンネルがリッスンする、またはリモート・ホスト・コンピュータからキャプチャするポートを入力します。

> **注意**
>
> 特権ポート (1024 未満) は、リモート・ホスト・コンピュータのルート権限でログインしたときにのみ転送できます。

- **接続先ホスト**: ポート転送の接続先ホストを定義します。デフォルト値は localhost です。

> **注意**
>
> この場合の localhost は、使用しているローカル・コンピュータを指します。また、リモート・ホスト・コンピュータからの接続がローカル・コンピュータを越えて転送される場合、その接続はセキュリティで保護されていないことに注意してください。

- **接続先ポート**: 接続先ホストで転送される接続に使用されるポートを定義します。

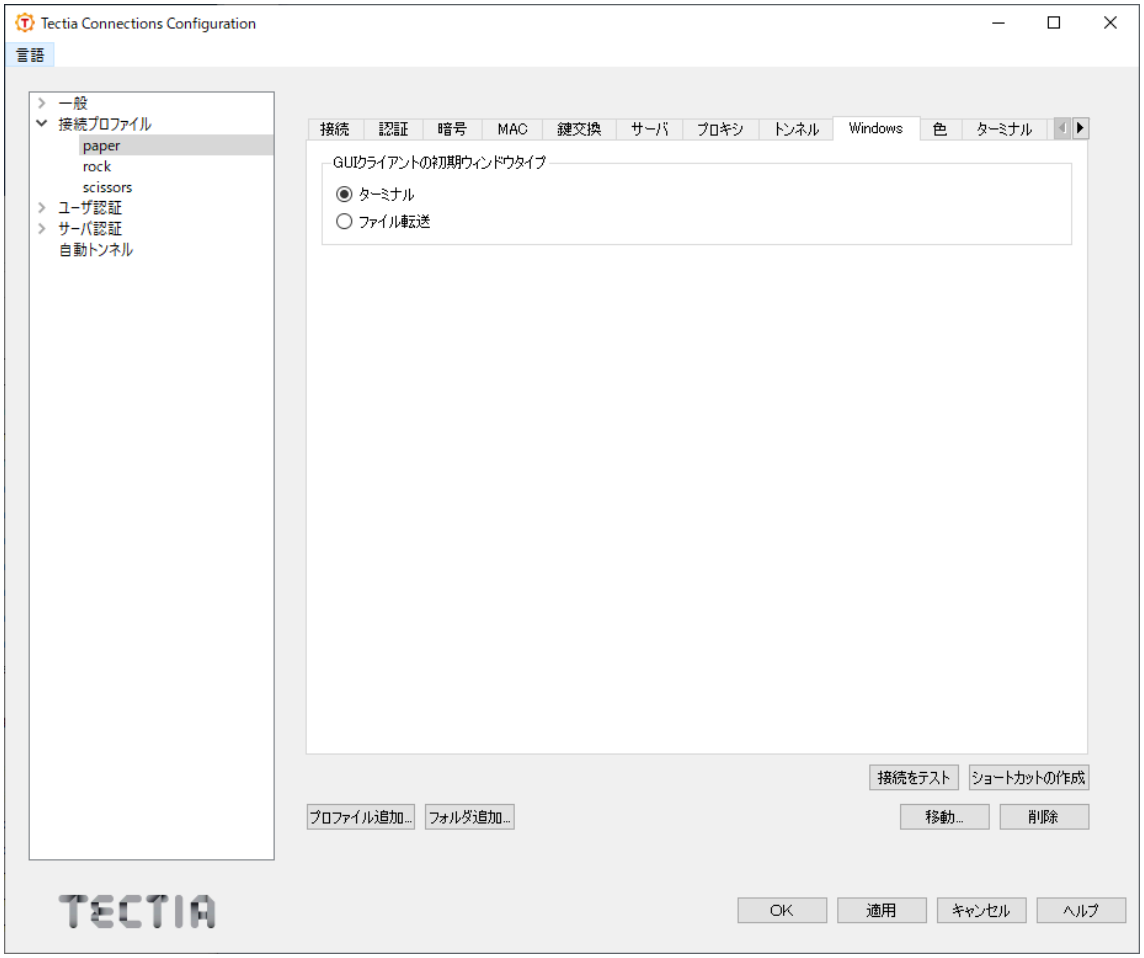
トンネルの定義を編集するには、リストからトンネルを選択し、[**編集**] をクリックします。[**リモートトンネル**] ダイアログが開きます。

トンネルの定義を削除するには、リストからトンネルを選択し、[**削除**] をクリックしてトンネルを削除します。選択されているトンネルは、確認ダイアログが表示されることなくすぐに削除されることに注意してください。

リモート・トンネルの詳細については、6.2 を参照してください。

ウィンドウ設定の定義

最初に開く Tectia ウィンドウの種類は、[**Windows**] タブで設定します。このプロファイルにアクセスすると、選択されている GUI バージョン (**Tectia SSHターミナル GUI** または **Tectia セキュア・ファイル転送 GUI**) が最初に開きます。



図A.28 初期の Tectia ウィンドウの種類の定義

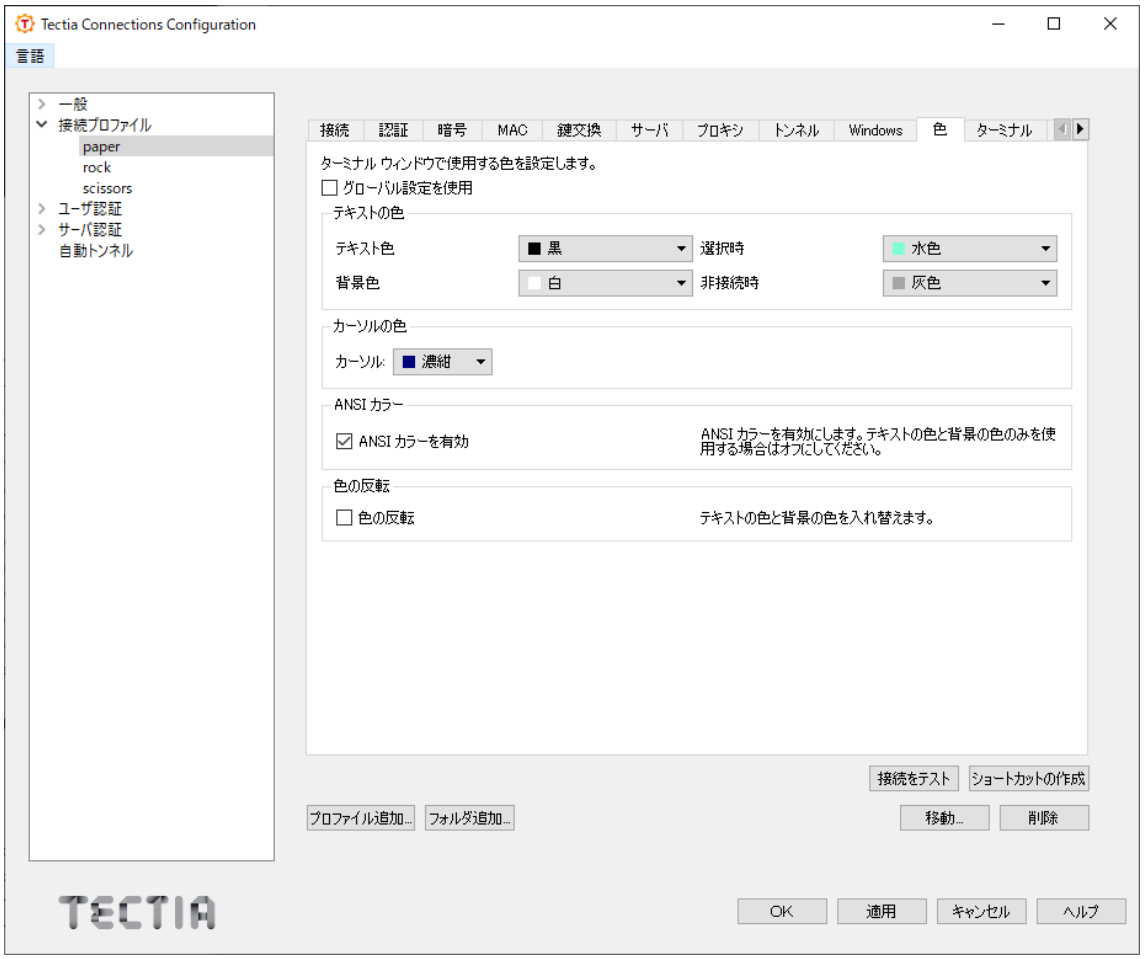
注意

[**プロファイル追加**] オプションを使用して Tectia コネクション設定　GUI からプロファイルを追加すると、新しいプロファイルの初期のウィンドウの種類は、現在の GUI ビューと同じになるように自動的に設定されます。

色の設定の定義

Tectia SSHターミナル GUI で使用する色は、[**色**] ページで選択できます。

色の設定は、グローバルに定義することも、プロファイルごとに定義することもできます。Tectia　ターミナルの　[グローバル設定]　で色が定義されている場合、[**グローバル設定を使用**]　オプションは使用できませんが、色の設定はすべての接続プロファイルに影響します。B.1.3 を参照してください。



図A.29 Tectia ターミナルの色の定義

グローバル設定を使用: このプロファイルにグローバル色設定を適用する場合は、このチェックボックスを選択します。このチェックボックスが選択されている場合、色設定を変更することはできません。

テキストの色

テキストの色は、接続されているウィンドウと接続されていないウィンドウの両方で、ターミナル・ウィンドウの背景色及びテキストの色に影響します。

- **テキスト色:** 希望するテキスト色をドロップダウン・メニューから選択します。テキスト色は、リモート・ホスト・コンピュータと接続しているウィンドウのテキストに使用されます。色は 16 色から選択できます。デフォルトのテキスト色は黒色です。
- **背景色:** 希望する背景色をドロップダウン・メニューから選択します。色は 16 色から選択できます。デフォルトの背景色は白色です。
- **選択時:** マウスで選択したテキストに対して希望する背景色を、ドロップダウン・メニューから選択します。色は 16 色から選択できます。デフォルトの選択色は水色です。

- **非接続時:** リモート・ホスト・コンピュータに接続されていないターミナル・ウィンドウに対して希望するテキスト色を選択します。色は 16 色から選択できます。灰色は、接続されていないターミナル・ウィンドウのデフォルトのテキスト色です。

カーソルの色

希望するカーソルの色をドロップダウン・メニューから選択します。色は 16 色から選択できます。デフォルトのカーソルの色は濃紺色です。

ANSI カラー

ANSI 制御コードを使用すると、ターミナル・ウィンドウのテキストの色を変更できます。ANSI カラーの設定では、この機能を使用するかどうかを選択できます。ANSI カラーを無効にしている場合でも、ターミナル・ウィンドウで使用するテキストの色や背景色を自由に選択できます。

ターミナル・ウィンドウで ANSI カラーを使用できるようにするには、[**ANSI カラーを有効**] チェックボックスを選択します。デフォルトでは、ANSI カラーが選択されています。

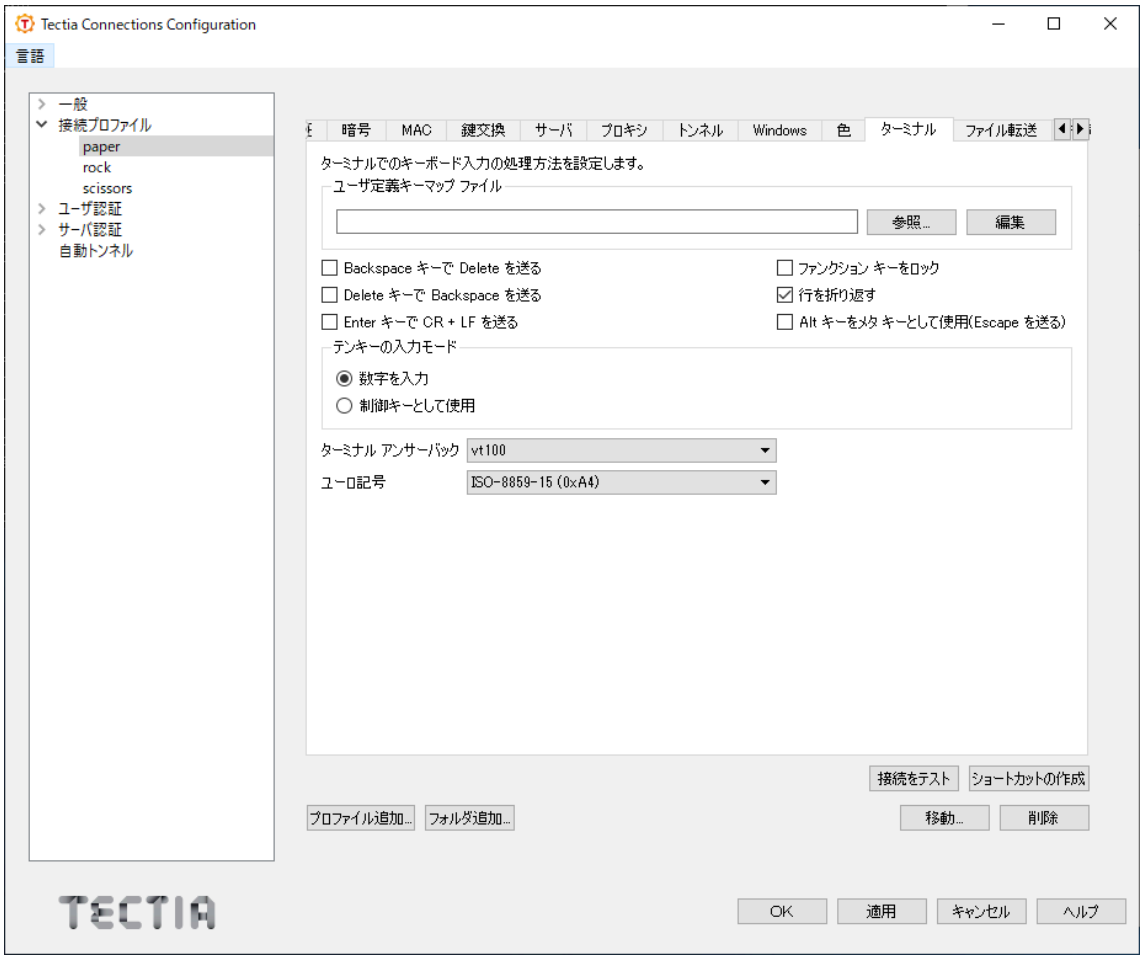
色の反転

表示色を反転させることで、ポジティブ (明に暗) 表示からネガティブ (暗に明) 表示へと素早く切り替えて、視認性を向上させることができます。

テキスト色と背景色を入れ替えるには、[**色の反転**] チェックボックスを選択します。この設定は、[**OK**] をクリックした時点でターミナル・ウィンドウ全体に影響します。

ターミナル設定の定義

Tectia Client ターミナルで使用する設定は、[**ターミナル**] タブを使って設定します。キーボードのマッピングは、新しい接続を開始したとき、またはターミナルをリセットしたときに有効になります。



図A.30 Tectia のターミナル設定の定義

ユーザー定義キーマップ ファイル

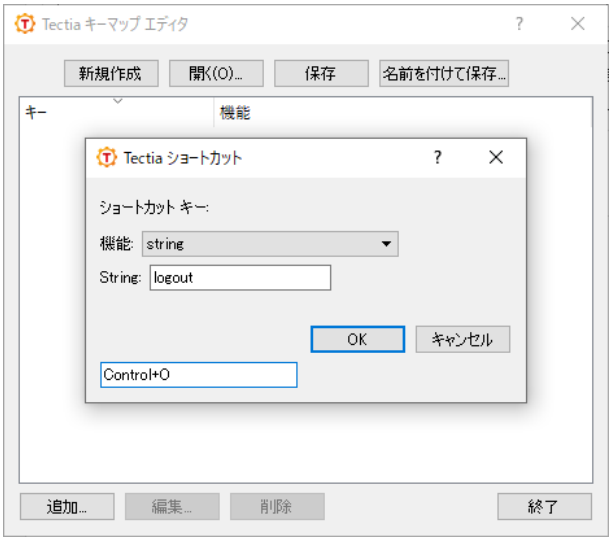
このオプションを使用して追加のキーボード・ショートカットを作成したり、既存のショートカットを変更したりします。追加のキー・マッピングは、`.sshmap` のファイル拡張子を持つ別のテキスト・ファイルに保存されます。テキスト・フィールドには、現在のキーマップ・ファイルが表示されます。

代替のキーマップ設定ファイルを定義している場合は、テキスト・フィールドにパスとファイル名を入力するか、または [**参照**] をクリックすることで、そのファイルを読み込むことができます。[**参照**] をクリックすると [**開く**] ダイアログボックスが開き、代替のキーマップ・ファイルを参照できます。

[**編集**] をクリックすると、現在のキー・マッピングを変更したり、新しいキー・マッピングを追加したりすることができます。[**編集**] をクリックすると [**Tectia キーマップ エディタ**] が開き、[**追加**] をクリックして新しいキー・マッピングを作成できます。[**追加**] をクリックすると [**Tectia ショートカット**] ダイアログボックスが開きます。

キーボード・ショートカットを定義するには、[**機能**] ドロップダウン・リストで、キーをマッピングする機能を選択します。機能によっては、機能を選択したときに表示され

る追加のテキストボックスやドロップダウン・リストを使用して、より詳細に定義できます。ダイアログボックスの左下にあるテキストボックスを選択した状態で、機能にマッピングするキーまたはキーの組み合わせを押します。



図A.31 Tectia キーマップエディタを使用したキーボード・ショートカットの追加

新しいキー・マッピングを使用するには、Tectia Client を再起動し、マッピングを作成したのと同じ接続プロファイルを使用してサーバに再接続します。キー・マッピングは、この特定の接続プロファイルにのみ適用されることに注意してください。

定義済みのキーボード入力

BackSpace キーを削除操作にマッピングする場合は、[**Backspace キーで Delete を送る**] チェックボックスを選択します。

Delete キーをバックスペース操作にマッピングする場合は、[**Delete キーで Backspace を送る**] チェックボックスを選択します。

Enter キーでキャリッジ・リターン (CR) とライン・フィード (LF) 文字を送信するようにマッピングする場合は、[**Enter キーで CR + LF を送る**] チェックボックスを選択します。選択解除すると、ライン・フィード文字のみが送信されます。

ファンクション・キーをロックする場合は、[**ファンクション キーをロック**] チェックボックスを選択します。

テキスト行をターミナル・ウィンドウの端で折り返す場合は、[**行を折り返す**] チェックボックスを選択します。デフォルトでは、行の折り返しは有効になっています。

Alt キーを Escape キーと同じように Meta キーとして機能させる場合は、[**Alt キーをメタ キーとして使用(Escape を送る)**] チェックボックスを選択します。このオプションを

選択すると、たとえば Alt + X キーを同時に押したときに、Escape キーの後に X キーを押した場合と同じ結果になります。

テンキーの入力モード

通常のキーボードの右側にあるテンキーをどのように機能させるかを選択します。

テンキーで数字を入力するには、[**数字を入力**] を選択します。

アプリケーションの操作にテンキーを使用するには、[**制御キーとして使用**] を選択します (テンキーがカーソル・キー、Home、End、Page Up、Page Down、Insert、及び Delete キーとして機能します)。

ターミナル アンサーバック

[**ターミナル アンサーバック**] ドロップダウン・リストを使用して、プロファイルに関連する Tectia Server で使用されているものと同じターミナル・アンサーバック・モードを選択します。

ユーロ記号

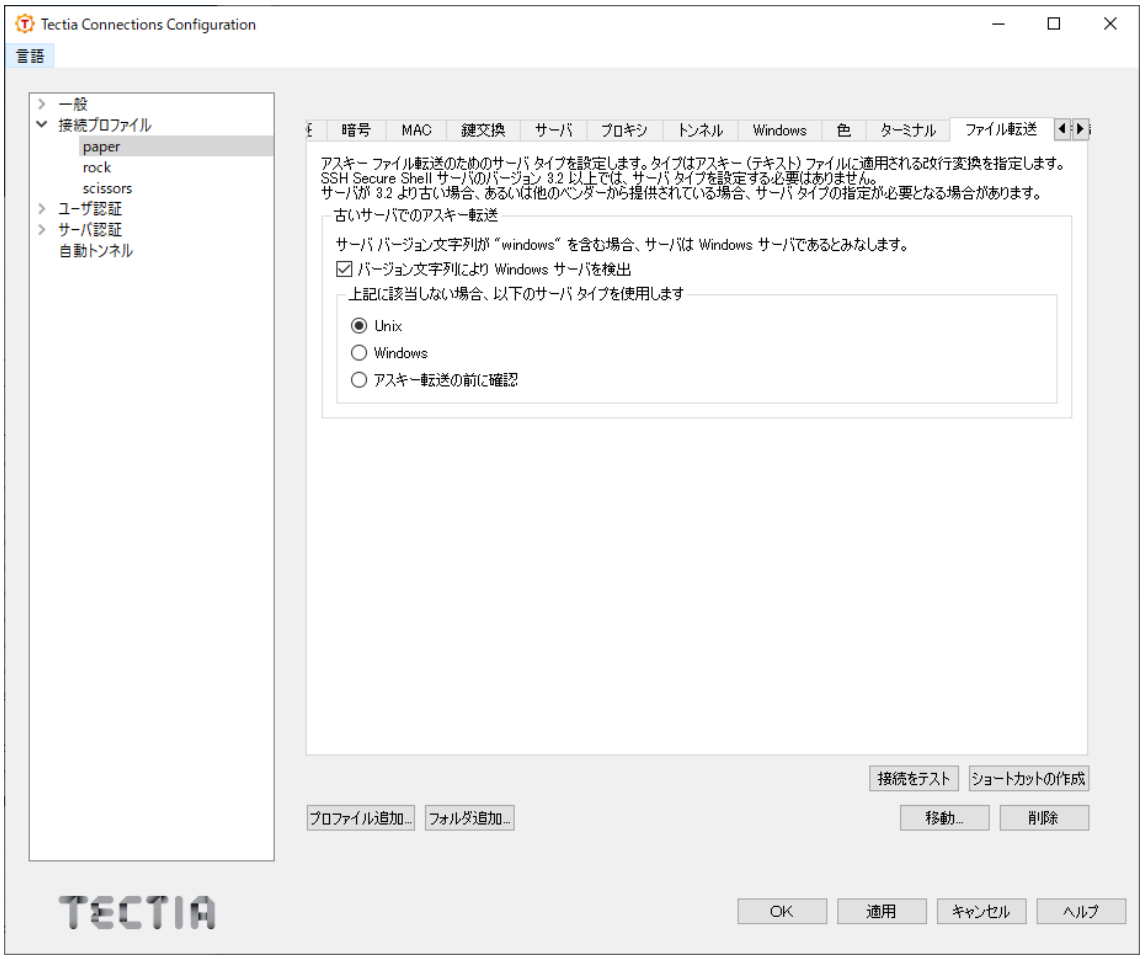
[**ユーロ記号**] ドロップダウン・リストを使用して、ユーロ記号 (€) のサポート・モードを選択します。

サポートされているオプションは Windows (ユーロは 0x80 としてマッピング) 及び ISO 8859-15 (ユーロは 0xA4 としてマッピング) です。プロファイルに関連する Tectia Server で使用されている文字コードと同じものを選択してください。

ただし、ユーロ記号のサポートを有効にすると、8 ビットのターミナル制御コードが無効になることに注意してください。

ファイル転送設定の定義

[**ファイル転送**] タブでは、どのファイルをアスキー・モードで転送するか、及びどの改行規則を適用するかを定義します。



図A.32 Tectia のファイル転送設定の定義

古いサーバでのアスキー転送

バージョン文字列により Windows サーバを検出: Secure Shell クライアントとサーバは、接続のセットアップ時にバージョン文字列を交換します。このチェックボックスを選択すると、Windows サーバが自動的に検出され、そのサーバに対して正しい設定が使用されます。この機能が正しく動作するためには、Windows サーバがバージョン文字列に "windows" を指定する必要があります。

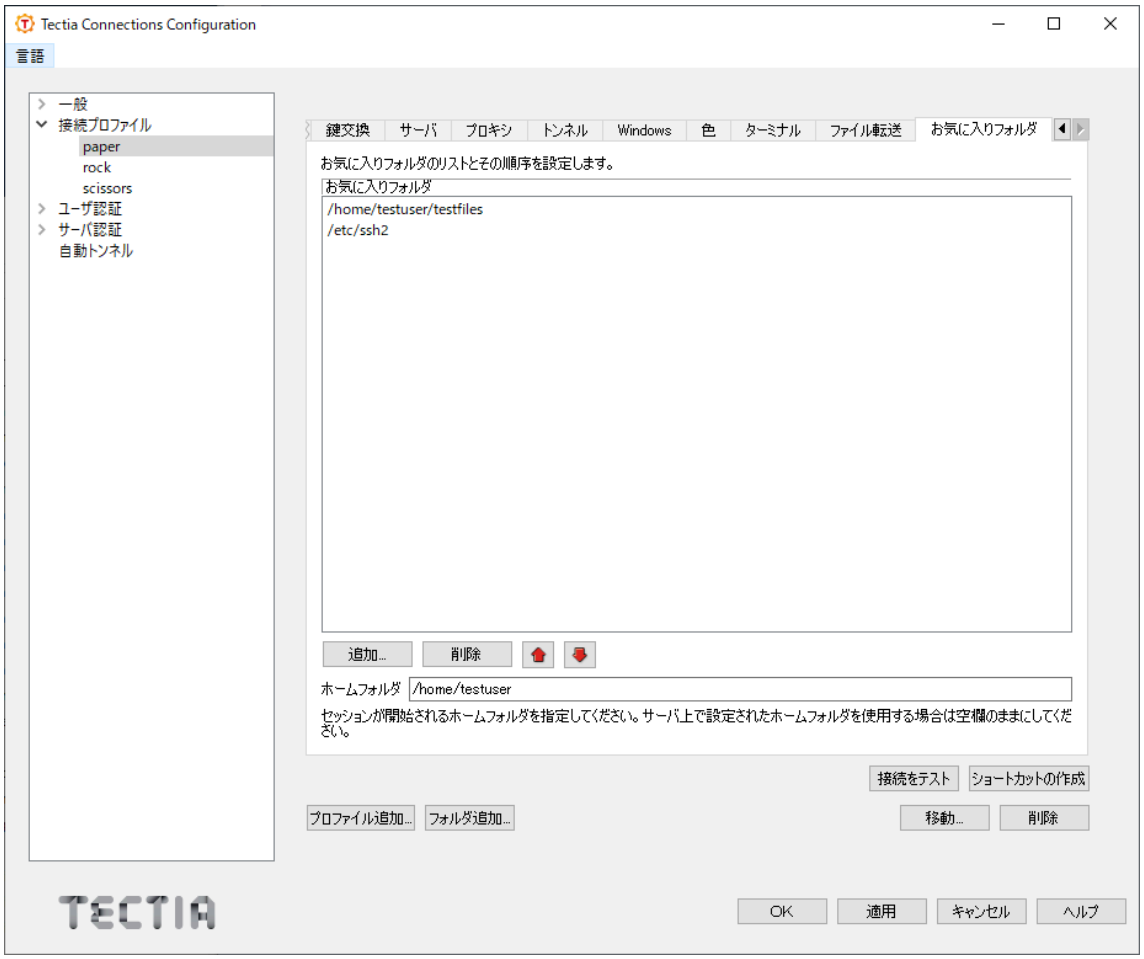
Unix 互換の改行 (LF) を使用するには、[**Unix**] チェックボックスを選択します。

Windows 互換の改行 (CRLF) を使用するには、[**Windows**] チェックボックスを選択します。

アスキー・ファイル転送を行う前に毎回、サーバの種類を指定するよう Tectia Client に確認させるには、[**アスキー転送の前に確認**] チェックボックスを選択します。

お気に入りフォルダの定義

[**お気に入りフォルダ**] タブでは、よく使用するリモート・ディレクトリのリストを作成できます。ファイル転送ウィンドウのドロップダウン・メニューからこれらのお気に入りを簡単に選択できるようになります。



図A.33 ファイル転送用のお気に入りリモート・フォルダの定義

お気に入りフォルダ

このリストには、現在の接続プロファイルに定義されているお気に入りフォルダが含まれています。リストの下にある [**追加**]、[**削除**]、及び矢印ボタンを使って、お気に入りの追加、削除、及び並べ替えができます。

Windows Secure Shell サーバ上にあるリモートのお気に入りを定義する場合、Windows サーバ上のフォルダを `/drive/folder/subfolder` のように指定する必要があります。

有効なお気に入りフォルダの定義は以下のようになります。

```
/C/Documents and Settings/All Users/Desktop
```

ホームフォルダ

[**ホームフォルダ**] フィールドには、このプロファイルに関連付けられた新しい SFTP 接続が開始されるディレクトリを入力できます。このフィールドを空にすると、リモート・ホスト・コンピュータのユーザ・アカウントに指定されているリモート・ホーム・フォルダが新しい接続に使用されます。

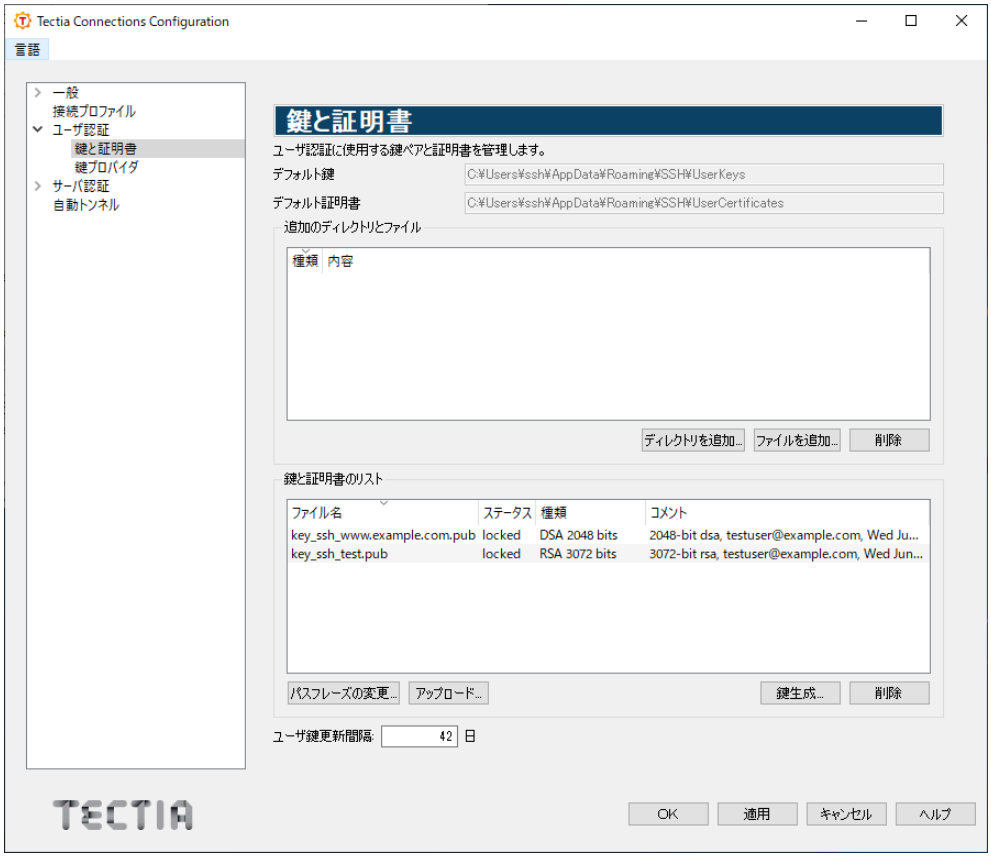
A.1.4. ユーザ認証の定義

[**ユーザ認証**] では、公開鍵認証と証明書認証に関する設定を行えます。「鍵と証明書の管理」 及び 「鍵プロバイダの管理」 を参照してください。

公開鍵認証の有効/無効を切り替える方法については、「デフォルト接続設定の定義」 及び 「認証の定義」 を参照してください。

鍵と証明書の管理

[**鍵と証明書**] ページでは、ユーザ認証に使用する鍵や証明書のファイル、及びそれらのディレクトリを追加したり、新しい鍵を生成したり、鍵をサーバにアップロードしたり、鍵のパスフレーズを変更したりすることができます。



図A.34 鍵と証明書の定義

デフォルト鍵

ユーザ鍵のデフォルトの場所。

デフォルト証明書

ユーザ証明書のデフォルトの場所。

追加のディレクトリとファイル

Tectia Client の設定に明示的に追加された、鍵のディレクトリ及びファイル。

- 鍵や証明書のディレクトリを追加するには、[**ディレクトリを追加**] ボタンをクリックします。
- 鍵や証明書のファイルを追加するには、[**ファイルを追加**] ボタンをクリックします。
- ディレクトリまたはファイルを削除するには、目的のディレクトリまたはファイルを選択して [**削除**] ボタンをクリックします。これによって、ディレクトリ、鍵または証明書ファイルへの参照が設定から削除されます。鍵自体がディスクから削除されることはありません。

鍵と証明書のリスト

Tectia Client が認識しているすべての公開鍵及び証明書がこのフィールドに一覧表示されます。つまり、[**デフォルト鍵**]、[**デフォルト証明書**]、及び [**追加のディレクトリとファイル**] フィールドに示されている場所に保存されている鍵と証明書です。また、外部鍵プロバイダから提供された鍵や証明書もここに表示されます (「鍵プロバイダの管理」 を参照)。

[**ステータス**] フィールドには、以下のいずれかの値が表示されます。

- **locked** - ファイルはパスフレーズで保護されており、パスフレーズは接続ブローカーに提供されていません。リモート・ホストにファイルをアップロードすると、ロックが解除されます。
- **open** - パスフレーズは接続ブローカーに提供されています。
- このフィールドが空の場合、ファイルはパスフレーズで保護されていません。

一覧表示されている鍵ファイルを選択し、下部のボタンをクリックすると、鍵の詳細を変更できます。

選択した鍵のパスフレーズを変更するには、[**パスフレーズの変更**] をクリックします。このコマンドは、あらゆる種類の鍵でサポートされているわけではないことに注意してください。

リモート・サーバに鍵をアップロードするには、[**アップロード**] をクリックします。アップロードできるのはプレーンな公開鍵のみです。「公開鍵の自動アップロード」 も参照してください。

鍵の生成ウィザードを起動するには、[**鍵生成**] をクリックします。新しい鍵は [**デフォルト鍵**] ディレクトリに追加され、[**鍵と証明書のリスト**] フィールドに表示されるようになります。ウィザードについては、「公開鍵認証ウィザードの使用」 を参照してください。

注意

公開鍵ファイルが保存されているユーザ固有の [Application Data] ディレクトリは、デフォルトで非表示になっています。非表示のディレクトリを表示するには、Windows エクスプローラの設定を変更します。 たとえば、メニューで [**整理**] → [**フォルダーと検索のオプション**] を選択します。 [**表示**] タブの [**ファイルとフォルダーの表示**] で、[**隠しファイル、隠しフォルダー、および隠しドライブを表示する**] を選択します。

ユーザ鍵更新間隔

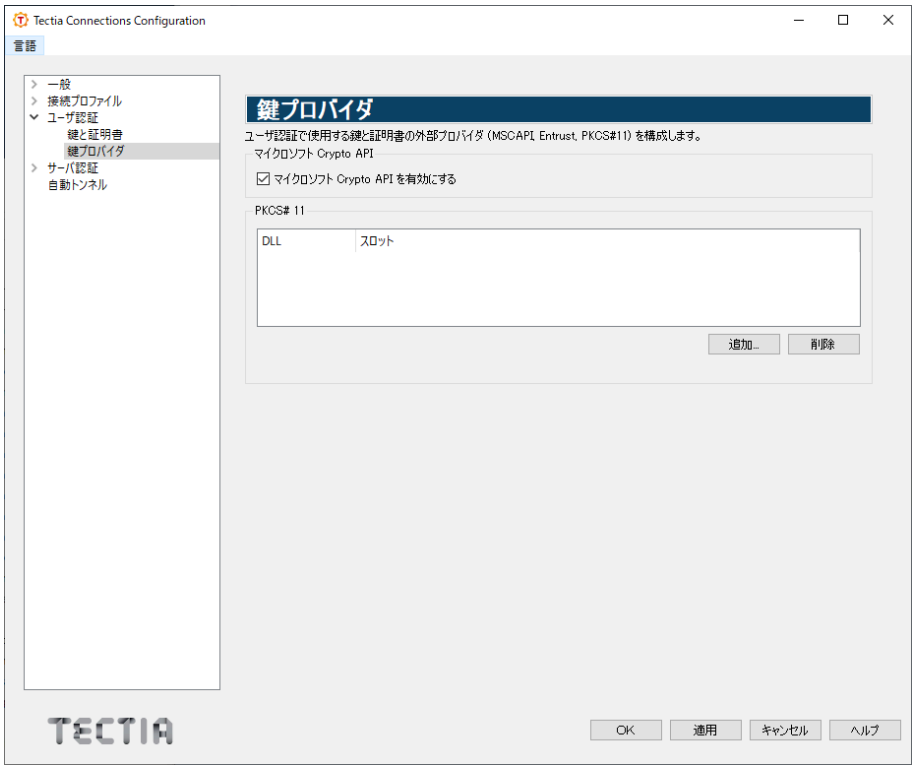
鍵の自動ローテーションが行われるまでの日数を設定します。これは、デフォルトの鍵の場所にあるユーザ鍵と、上記の追加ディレクトリとして定義された場所にあるユーザ鍵の両方に影響します。独立した鍵ファイルは鍵ローテーションをサポートしていません。ローテーション期間を　0　に設定すると、鍵の自動ローテーションは無効になります。

クライアントはホストに接続する際、鍵ローテーション期間より古い鍵があると、新しく生成された鍵への置き換えを試みます。サーバがユーザによる鍵のアップロードを許可していない場合、この試みは機能しません。

警告: 同じ秘密鍵が複数のクライアントにコピーされている場合、そのうちの一台から公開鍵が置き換わると、他の接続が失われます。

鍵プロバイダの管理

[**鍵プロバイダ**]　　ページでは、ユーザ認証に使用する外部鍵プロバイダの設定を定義できます。利用できる鍵プロバイダはマイクロソフト Crypto APIと PKCS #11 です。



図A.35 鍵プロバイダの定義

マイクロソフト Crypto API

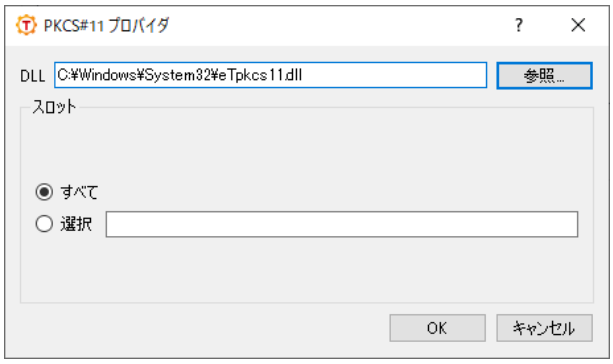
Tectia Client はマイクロソフト Crypto API (MSCAPI) 経由で鍵にアクセスできます。MSCAPI は、Microsoft Windows システムで使用される標準的な暗号化インターフェイスです。

マイクロソフト Crypto API (MSCAPI) プロバイダを有効にするには、[**マイクロソフト Crypto API を有効にする**] チェックボックスを選択します。MSCAPI プロバイダを有効にすると、Microsoft のアプリケーションで作成されたソフトウェアの鍵と証明書を使用できます。

PKCS #11

PKCS #11 プロバイダを使用すると、Tectia Client は PKCS #11 トークン (スマート・カードや USB トークンなど) に保存されている鍵と証明書を使用できます。

PKCS #11 プロバイダを定義するには、[**追加**] をクリックします。



図A.36 PKCS #11 プロバイダの定義 (例: Aladdin eToken DLL のパス)

PKCS #11 ドライバを含むダイナミック・ライブラリを定義するには、[**DLL**] を使用します。

スロットを定義するには、[**スロット**] を使用します。スロットは、トークンを含む可能性のある論理リーダです。スロットは製造元によって異なります。スロットは整数で定義されます。例: "`0,1`"、"`0-3, !2`"、"`2`"。

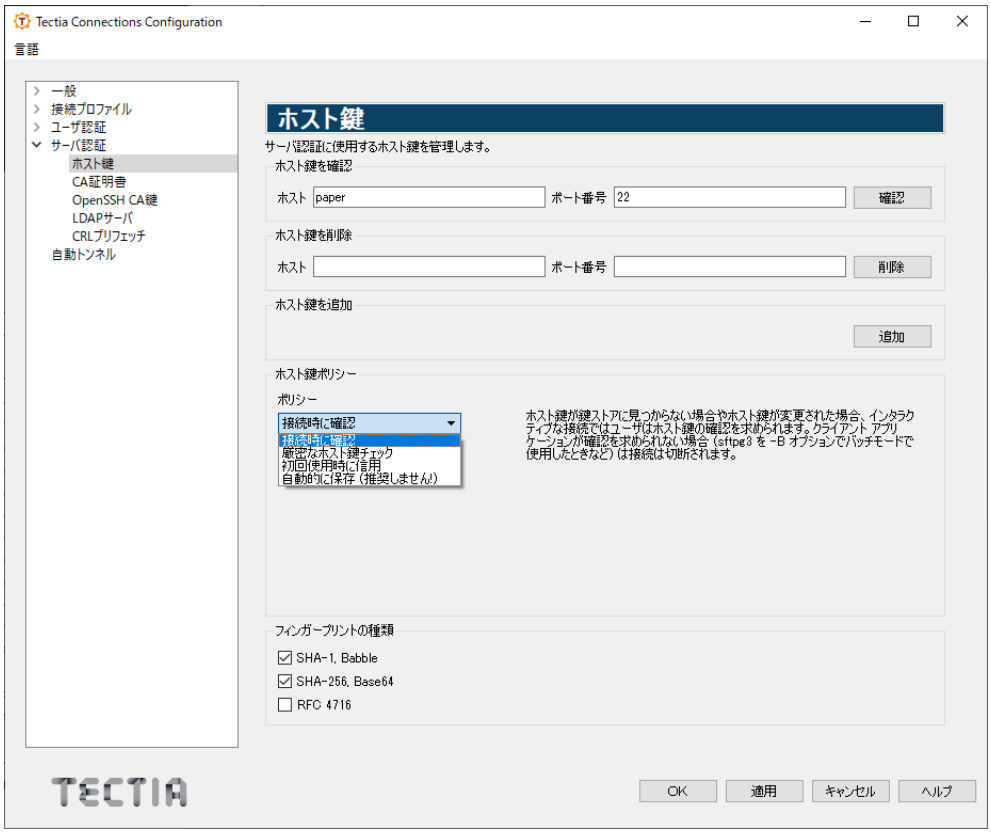
A.1.5. サーバ認証の定義

[**サーバ認証**] では、Tectia Client がリモート・サーバ・ホストを認証する方法を定義できます。

- サーバ認証で公開鍵を使用するには、「ホスト鍵の管理」 で説明されているように設定を定義します。
- 証明書を適用するには、「CA 証明書の管理」 で説明されているように設定を定義します。
- LDAP の使用に必要な設定は、「LDAP サーバ設定の管理」 で説明されています。
- 一定の間隔で証明書失効リスト (CRL) を取得する方法については、「CRL プリフェッチ設定の管理」 を参照してください。

ホスト鍵の管理

[**ホスト鍵**] ページでは、新しいホスト公開鍵を追加したり、ホスト鍵の受け入れポリシーを定義したり、サーバ認証で使用される既知のホスト鍵を表示及び管理したりすることができます。既知のホスト鍵とは、すでにユーザ固有の `%APPDATA%\SSH\Hostkeys` ディレクトリに保存されている鍵のことです。



図A.37 サーバのホスト鍵設定の定義

注意

バージョン 6.1.4 でホスト鍵ポリシーの設定が変更されました。Tectia コネクション設定 GUI は、古い [**厳密なホスト鍵チェック**]、[**不明なホスト鍵を許可**]、及び [**常にホスト鍵の入力を求める**] の設定に基づく新しいポリシーを使用するために、ユーザ固有の設定を自動的に更新します。新しいポリシーに合わせた古いポリシーの解釈については、表 A.2 を参照してください。

[**ホスト鍵**] ビューには以下のオプションがあります。

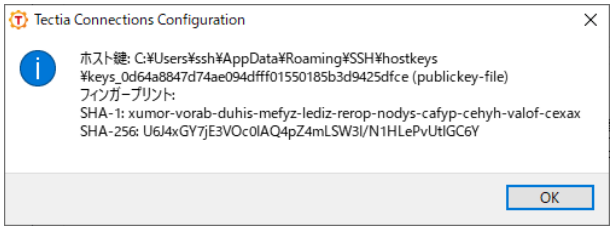
ホスト鍵を確認

リモート・サーバのホスト公開鍵が自分のクライアントに存在するかどうかを確認し、そのフィンガープリントを表示できます。ホスト鍵を確認するには、[**ホスト**] フィールドにサーバ名を、[**ポート番号**] フィールドにリスナ・ポート番号をそれぞれ入力し、[**確認**] をクリックします。

ワイルドカード文字は使用できないため、正確なホスト名とポートを指定してください。

指定したサーバのホスト公開鍵がクライアント上で見つかると、ホスト鍵の保存場所と公開鍵のフィンガープリントがダイアログボックスに表示されます。フィンガープリン

トは　SSH　Babble　形式で表示されます。この形式は、ダッシュで区切られた小文字5 文字からなる読み上げ可能な一連の単語で構成されています。以下の例を参照してください。



図A.38 サーバ・ホスト公開鍵の情報

サーバ・ホスト公開鍵の詳細については、4.2 を参照してください。

ホスト鍵を削除

既知のホスト公開鍵をクライアント側から削除するには、[**ホスト**]　フィールドに該当サーバ名を、[**ポート番号**] フィールドにリスナ・ポート番号をそれぞれ入力し、[**削除**] をクリックします。

ホスト鍵の削除を確定するか、またはキャンセルするかを求めるダイアログボックスが表示されます。

ホスト鍵を追加

新しいホスト鍵を既知のホスト鍵のディレクトリに追加するには、[**追加**]　ボタンをクリックします。接続ブローカーがファイル・マネージャ・ビューを開き、そこで鍵の場所を参照し、コピーするホスト鍵を選択できます。

ホスト鍵ポリシー

サーバ・ホスト鍵の確認と、不明なサーバ・ホスト鍵の処理に適用するポリシーを選択します。

注意

この設定は、クライアント側ホストのセキュリティに大きく影響します。

以下のオプションがあります。

- **接続時に確認**- デフォルト - ホスト鍵ストアに鍵が見つからない場合や、鍵が変更されている場合、サーバ・ホスト公開鍵を検証して受け入れるようユーザに要求します。ユーザは、鍵を `%APPDATA%\SSH\Hostkeys` に保存するか、保存せずに一度使用するか、またはキャンセルするかを決定できます。ホスト鍵が既知のホスト鍵のディレクトリで見つかったサーバか、またはホスト鍵が現在、ユーザによって受け入れられたサーバへの接続のみが許可されます。

このポリシーでは、ユーザからの応答を得るために対話型の接続が必要です。非対話型の接続で [**接続時に確認**] オプションが適用された場合、接続は終了します。

- **厳密なホスト鍵チェック** - サーバへの接続は、ホスト鍵がユーザの既知のホスト鍵のストレージで見つかった場合にのみ許可されます。それ以外の場合、接続は終了します。このオプションは、受け入れられるすべてのサーバ・ホスト鍵がすでにクライアントに保存されていることを想定しています。新しいホスト鍵は保存されず、ホスト鍵が変更されたサーバへの接続は終了します。

  このオプションは、ホスト鍵が他の手段で受信された後に、非対話型の接続で使用できます。このポリシーは中間者攻撃に対する最大の防御を発揮します。

- **初回使用時に信用** - 新しいホスト鍵は、それを受け入れることをユーザに求めることなく保存されます。変更されたホスト鍵を提供するサーバへの接続は終了します。このポリシーは、サーバ・ホスト鍵が他の手段で鍵ストレージに追加できない場合にのみ使用してください。

- **自動的に保存**- 非推奨 - 新しいホスト鍵は、それを受け入れることをユーザに求めることなく保存され、変更されたホスト鍵を提供するサーバへの接続も許可されます。変更された鍵はクライアントには保存されず、接続ブローカーでログが有効になっていれば、その鍵で接続を開いたことに関するデータがログに記録されます。

  このポリシーを選択する場合は、[**一般**] - [**ログ**] ビューで、接続ブローカーのログが有効になっていることを確認してください (「ログ設定の定義」 を参照)。そのようにしておくと、ホスト鍵が変更された接続があればログで検出できます。

  警告

  [**自動的に保存**] ポリシーを有効にすると、サーバ認証が実質的に無効になり、能動的な攻撃者に対して接続が脆弱になるため、あらかじめ慎重に検討してください。

ローテーション

サーバ・ホスト鍵に適用するローテーション・オプションを選択します。

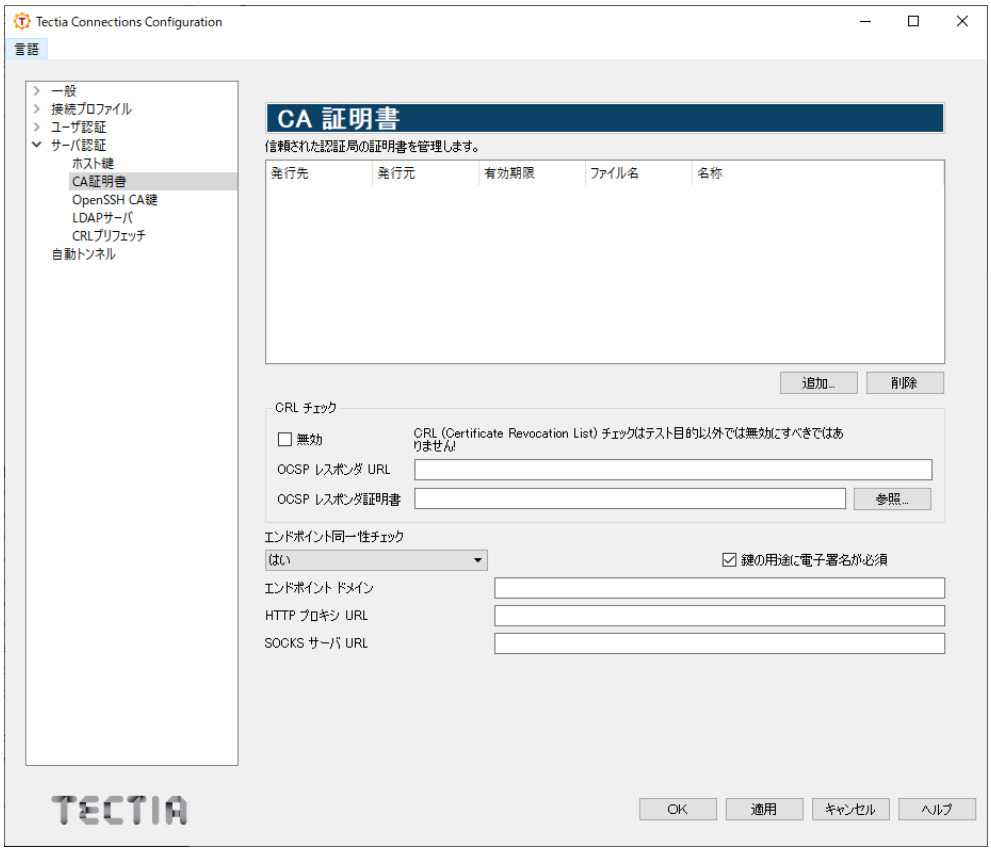
以下のオプションがあります。

- **いいえ** - 鍵ローテーションが無効になります。
- **はい** - 鍵ローテーションが有効になります。
- **新しい鍵の追加のみ** - 鍵ローテーションが有効になります。このオプションを選択すると、古い鍵は削除されずに、新しい鍵ファイルが鍵ファイルに追加されます。
- **Tectia Server のみ** - デフォルト - 鍵ローテーションが有効になりますが、Tectia Server に限られます。このオプションはサーバでも有効にする必要があります。

CA 証明書の管理

[**CA 証明書**] ページでは、信頼された CA 証明書を管理できます。

サーバ証明書認証の詳細については、4.3 を参照してください。



図A.39 CA 証明書の定義

CA 証明書を追加するには、[**追加**] ボタンをクリックし、追加する証明書を選択します。

X.509 証明書が追加できます。PKCS #7 パッケージ (`.p7b`) から証明書を追加するには、まず、コマンドラインで **ssh-keygen-g3** を使い `-7` オプションを指定して、パッケージから CA 証明書を抽出する必要があります。

```
> ssh-keygen-g3 -7 certfile.p7b
```

そうすることで、抽出した CA 証明書を追加できるようになります。

CA 証明書のリストには、以下のフィールドが表示されます。

- **発行先:** 証明書の発行先の認証局。
- **発行元:** CA 証明書を発行したエンティティ。
- **有効期限:** CA 証明書の有効期限が切れる日付。

- **ファイル名:** CA 証明書を含むファイル。

CRL チェック

証明書失効リスト (CRL) を使用しないようにするには、[**無効**] チェックボックスを選択します。CRL は、使用中のサーバ証明書に失効しているものがないか確認するために使用されます。

注意

CRL チェックを無効にすることはセキュリティ・リスクであるため、テスト目的以外では無効にしないでください。

OCSP レスポンダ URL

OCSP レスポンダ・サービスは、オンライン証明書状態プロトコル (OCSP) を使用して、証明書の有効状態に関する情報をリアルタイムに取得するための管理ポイントをクライアント・アプリケーションに提供します。

OCSP 検証が成功するためには、エンドエンティティ (= Secure Shell サーバ) 証明書と OCSP レスポンダ証明書の両方が同じ CA から発行される必要があります。証明書に `Authority Info Access` 拡張機能による OCSP レスポンダ URL が含まれている場合は、OCSP レスポンダが設定されていない場合にのみ使用されます。OCSP レスポンダが設定されている場合、この拡張機能は使用されません。

OCSP レスポンダが設定ファイルまたは証明書に定義されている場合、最初にそれが試され、それが失敗した場合にのみ、従来の CRL チェックが試され、さらにそれが失敗した場合、証明書検証は失敗を返します。

エンドポイント同一性チェック

クライアントがサーバのホスト名または IP アドレスを、サーバ・ホスト証明書で指定された [**サブジェクト名**] または [**サブジェクト代替名**] (DNS アドレス) に照らして検証するかどうかを指定します。デフォルトでは、[**エンドポイント同一性チェック**] は有効になっています (オプションは [はい])。その他に [いいえ] と [接続時に確認] のオプションがあります。

[いいえ] を選択した場合、サーバ・ホスト証明書のこのフィールドは検証されず、有効期間と CRL チェックのみに基づいて証明書が受け入れられます。

警告

クライアントでエンド・ポイント同一性チェックを無効にすることはセキュリティ・リスクです。そのようにすると、サーバ・ホスト証明書の発行元と同じ、信頼できる CA から発行された証明書を持っている人は誰でも、サーバに対して中間者攻撃を行えるようになります。

[接続時に確認] を選択した場合、証明書情報を検証し、接続を許可するか、またはキャンセルすることを求められます。

鍵の用途に電子署名が必須

アメリカ合衆国国防総省公開鍵基盤 (DoD PKI) のコンプライアンス要件の 1 つに、証明書の鍵用途でデジタル署名ビットを設定することがあります。鍵の用途に電子署名を必須にし、コンプライアンス要件を満たすには、このチェックボックスを選択します。

エンドポイント ドメイン

エンド・ポイント同一性チェックで使用されるデフォルト・ドメインを指定します。これはリモート・システム名のデフォルトのドメイン部分で、システム名のベース部分のみしか入手できない場合に使用されます。

デフォルト・ドメインが指定されていない場合でも、エンド・ポイント同一性チェックは短いホスト名で機能します。たとえば、ユーザが短いホスト名だけを指定してホスト "`rock`" への接続を試みた場合、証明書に完全な DNS アドレス "`rock.example.com`" が含まれていれば、接続は開かれ、Tectia Client は "`rock`" への接続を許可するという内容の警告を発します。

HTTP プロキシ URL

証明書の有効性を LDAP または OCSP で問い合わせる際に使用する HTTP プロキシを指定します。

アドレスのフォーマットは `"http://username@proxy_server:port/network/netmask,network/netmask... "` です。 `network/netmask` の部分はオプションで、(プロキシなしで) 直接接続されるネットワークを定義します。

SOCKS サーバ URL

証明書の有効性を LDAP または OCSP で問い合わせる際に使用する SOCKS サーバを指定します。

アドレスのフォーマットは `"socks://username@socks_server:port/network/netmask,network/netmask... "` です。 `network/netmask` の部分はオプションで、(SOCKS サーバなしで) 直接接続されるネットワークを定義します。

OpenSSH CA 鍵の管理

[OpenSSH CA 鍵] ページでは、OpenSSH 証明書を管理できます。

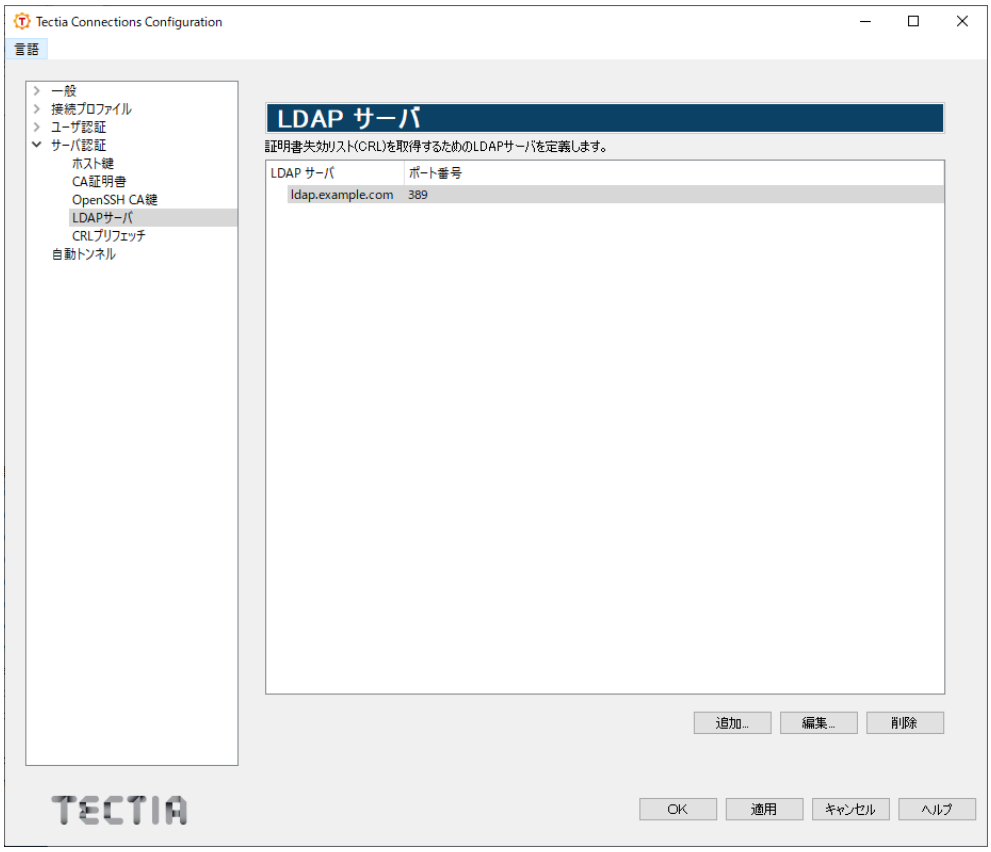
OpenSSH 証明書を追加するには、[**追加**] ボタンをクリックします。

OpenSSH 証明書を削除するには、リストから証明書を選択し、[**削除**] をクリックします。

LDAP サーバ設定の管理

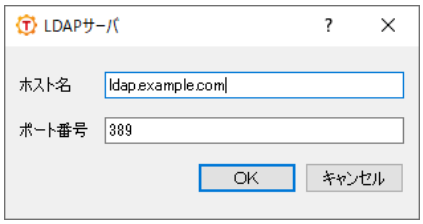
[**LDAP サーバ**] ページでは、検証中の証明書の発行者名に基づいて、CRL 及び/または下位の CA 証明書を取得するために使用する LDAP サーバを定義できます。

証明書に定義された CRL 配布ポイントが存在すれば、CRL はそこから自動的に取得され、検証されます。



図A.40 LDAP サーバの定義

LDAP サーバを追加するには、[**追加**] ボタンをクリックします。サーバのホスト名とポートを定義します。



図A.41 LDAP サーバの追加

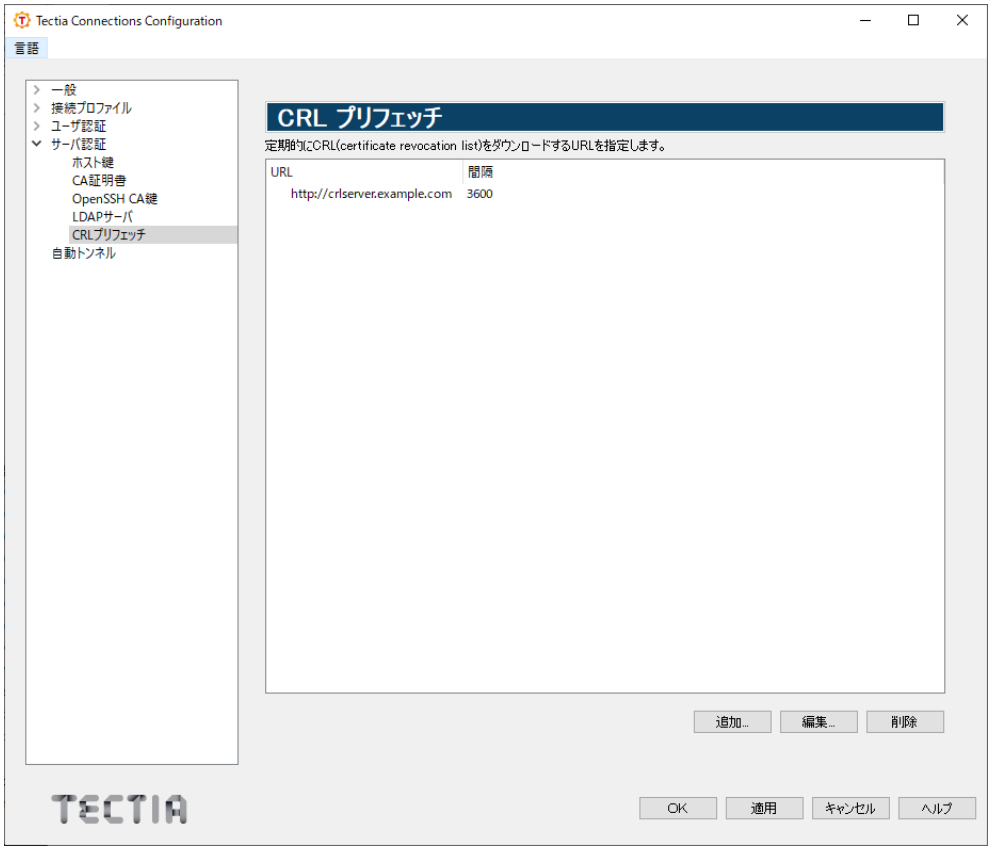
LDAP サーバを編集するには、リストからサーバを選択し、[**編集**] をクリックします。

LDAP サーバを削除するには、リストからサーバを選択し、[**削除**] をクリックします。

CRL プリフェッチ設定の管理

[**CRL プリフェッチ**] ページでは、証明書失効リスト (CRL) を定義された場所から一定の間隔で取得することを定義できます。CRL の配布ポイントには、標準形式の LDAP または HTTP の URL を指定するか、またはファイルを参照させることができます。ファイル形式はバイナリ DER または base64 でなければならず、PEM はサポートされていません。

証明書に定義された CRL 配布ポイントが存在すれば、CRL はそこから自動的に取得され、検証されます。



図A.42 CRL プリフェッチ設定の定義

CRL プリフェッチ アドレスを追加するには、[**追加**] をクリックします。[**CRL プリフェッチ**] ダイアログボックスが開きます。

図A.43 CRL プリフェッチ設定の追加

CRL の配布ポイントの [**URL**] と、CRL をダウンロードする [**間隔**] を入力し、[**OK**] をクリックします。デフォルトのダウンロード間隔は [`3600`] (秒) です。

CRL の配布ポイントがファイルを参照している場合は、ファイルの URL を以下の形式で入力します。

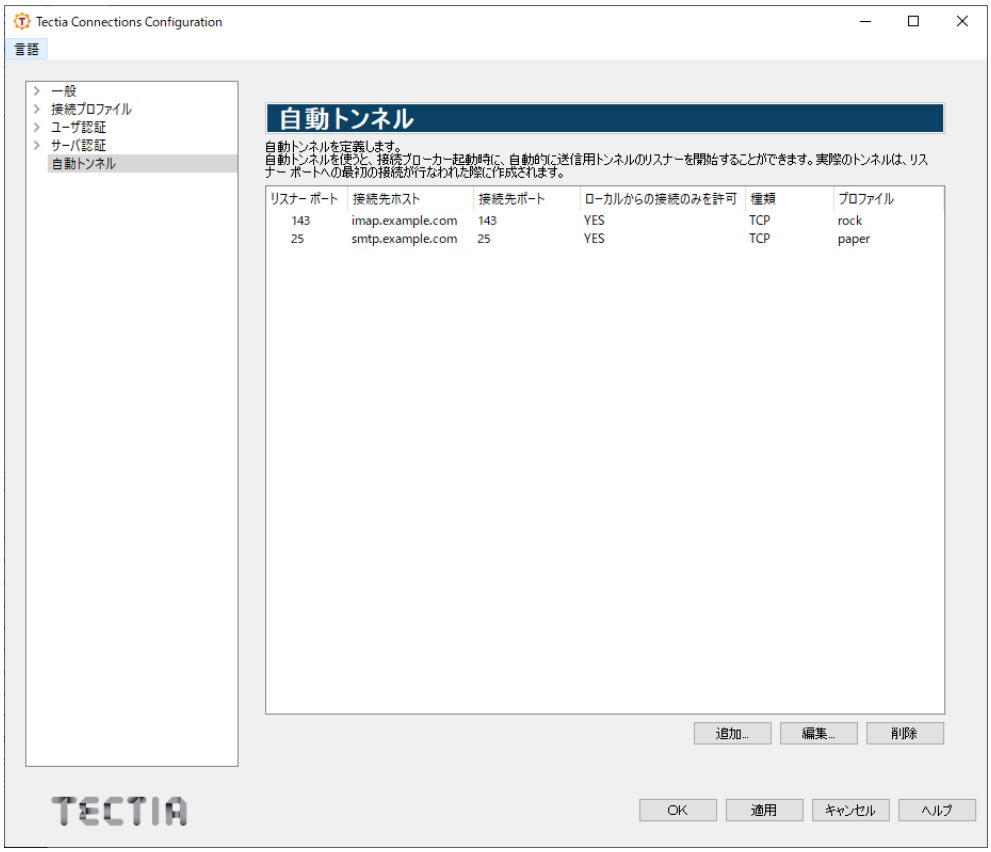
```
file:///absolute/path/name
```

既存の CRL プリフェッチ設定を編集するには、リストから設定を選択し、[**編集**] をクリックします。

既存の CRL プリフェッチ設定を削除するには、リストから設定を選択し、[**削除**] をクリックします。

A.1.6. 自動トンネルの定義

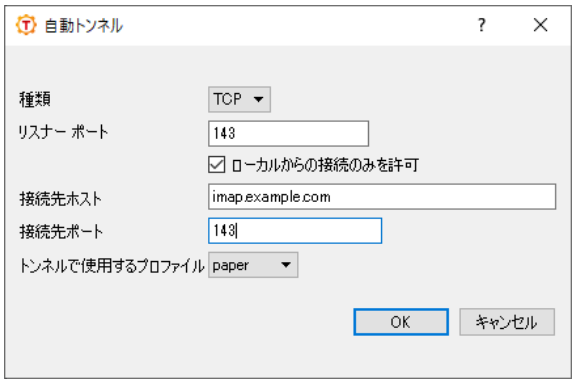
[**自動トンネル**] ページでは、接続ブローカーの起動時に自動的に開始されるローカル・トンネルのリスナを作成できます。実際のトンネルは、リスナ・ポートに初めて接続されたときに形成されます。その時点でサーバへの接続が開かれていない場合は、その接続も自動的に開かれます。



図A.44 自動トンネルの定義

接続ブローカーの起動時、自動トンネルのリストが読み込まれ、接続を開始するアプリケーションが、ここで定義されたルールに照合されます。

ツリー・メニューで [**自動トンネル**] を選択し、[**追加**] をクリックして [**自動トンネル**] ダイアログボックスを開きます。



図A.45 新しい自動トンネルの追加

- **種類:** トンネルの種類をドロップダウン・リストから選択します。TCP または FTP を選択できます。
- **リスナー　ポート:**　これは、トンネルがリッスンする、またはキャプチャするローカル・ポートの番号です。予約されたポート番号は使用しないでください。

  注意

  正常に接続するために、トンネルを作成する対象のプロトコルまたはアプリケーションには固定のポート番号 (IMAP の場合は 143 など) が必要な場合があります。その他のプロトコルまたはアプリケーションについても、オフセット (VNC の場合は 5900 など) を考慮に入れなければならない場合があります。

- **ローカルからの接続のみを許可:** ローカル接続のみを許可する場合は、このチェックボックスを選択したままにします。この場合、作成されたトンネルは他のコンピュータでは使用できません。デフォルトでは、ローカル接続のみが許可されます。これはほとんどの状況に適した選択です。外部からの接続も許可する場合は、それに伴うセキュリティへの影響も十分に検討してください。
- **接続先ホスト:** このフィールドでは、ポート転送の接続先ホストを定義します。デフォルト値は localhost です。

  注意

  ローカルホストの値は Secure Shell サーバによって解決されるため、この場合の `localhost` は接続先の Secure Shell ホストを指します。

- **接続先ポート:** 接続先ポートは、接続先ホストで転送される接続に使用されるポートを定義します。
- **トンネルで使用するプロファイル:** トンネルに使用するプロファイルを選択します。

自動トンネルを編集するには、リストからトンネルを選択し、[**編集**] をクリックします。

自動トンネルを削除するには、リストからトンネルを選択し、[**削除**] をクリックします。

トンネリングの詳細については、6.1 を参照してください。

A.2. 接続ブローカーの設定ファイル

XML ベースの接続ブローカー設定ファイル `ssh-broker-config.xml` のエレメントについては、ssh-broker-config(5) で説明されています。

ssh-broker-config

ssh-broker-config — Tectia 接続ブローカーの設定ファイルのフォーマット

接続ブローカーの設定ファイル `ssh-broker-config.xml` は、Unix と Windows 上の Tectia Client 及び ConnectSecure で使用されます。接続ブローカーの設定ファイルは、文書型定義 `ssh-broker-ng-config-1.dtd` に従った妥当なXML文書である必要があります。

接続ブローカーのファイル

接続ブローカーは、以下の 3 つの設定ファイルを読み込みます (すべて利用できる場合)。

1. 最初に `ssh-broker-config-default.xml` ファイルが読み込まれます。このファイルには工場出荷時の設定が保持されています。このファイルは編集せず、デフォルト設定を確認するためにのみ使用して下さい。。

   接続ブローカーが起動するためには、このファイルは使用可能な状態であり、正しくフォーマットが維持されている必要があります。

2. 次に接続ブローカーは、グローバル設定ファイルを読み込みます。グローバル設定ファイルの設定はデフォルト設定を上書きします。

   グローバル設定ファイルがない場合や、フォーマットが正しくない場合、接続ブローカーは通常通り起動し、代わりにユーザ固有の設定ファイルを読み込みます。フォーマットが正しくないグローバル設定ファイルは無視され、デフォルト設定またはユーザ固有の設定が存在する場合は、それが代わりに使用されます。

3. 最後に接続ブローカーは、ユーザ固有の設定ファイルがあれば、それを読み込みます。ユーザ固有の設定ファイルの設定は、以下の例外を除き、グローバル設定ファイルの設定を上書きします。

   - ユーザ固有の設定にある以下の設定は、グローバル設定ファイルの設定と組み合わされます。
     - `general` エレメントの `key-stores`、 `cert-validation`、及び `file-access-control` の設定
     - `profiles` エレメントのすべての設定
     - `static-tunnels` エレメントのすべての設定
   - グローバル設定ファイルとユーザ固有の設定ファイルの両方に同じ名前の接続プロファイルが定義されている場合、後者の接続プロファイルが使用されます。
   - `filter-engine` の設定がグローバル設定ファイルで定義されており、そのファイルが有効である (フォーマットが正しい) 場合、その設定が使用され、ユーザ固有の設定ファイルで行われた `filter-engine` の設定は無視されます。

ユーザ固有の設定ファイルがない場合、接続ブローカーは以前に読み込んだ設定ファイルを使用して起動します。ただし、ユーザ固有の設定が存在してもフォーマットが正しくない場合、接続ブローカーは全く起動しません。

Unix では、デフォルトの設定ファイルの場所は以下の通りです。

- デフォルトの設定:

  ```
  /opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml
  ```

- グローバル設定: `/etc/ssh2/ssh-broker-config.xml`
- ユーザ固有の設定: `$HOME/.ssh2/ssh-broker-config.xml`
- XML DTD:

  ```
  /opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd
  ```

注意

Unix 上の Tectia Client 6.1 以前では、デフォルトの補助データ・ディレクトリ `auxdata` は `/etc/ssh2/ssh-tectia/` でした。`ssh-broker-config.xml` ファイルが Tectia Client のバージョン 6.1 以前で作成されている場合は、接続ブローカー設定ファイルの DTD ディレクトリへの現在のパスが含まれるように、DOCTYPE 宣言を`/opt/tectia/share/auxdata/ssh-broker-ng/`に更新してください。

Windows の場合、デフォルトの設定ファイルの場所は以下の通りです (`<INSTALLDIR>` は Windows のデフォルトの Tectia インストール・ディレクトリを示します。1.1.2 を参照してください)。

- デフォルトの設定: "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml`"
- グローバル設定: "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml`"
- ユーザ固有の設定: "`%APPDATA%\SSH\ssh-broker-config.xml`"
- XML DTD: "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-ng-config-1.dtd`"

以下の項では、接続ブローカーの設定ファイルで利用できるオプションについて説明します。設定ファイルの構文の詳細については、A.5 を参照してください。

環境変数

接続ブローカー設定ファイルでは 2 種類の環境変数を使用できます。システム・レベルの環境変数に加えて、Tectia 固有の特別な変数を使用できます。環境変数は特殊な変数より優先されます。そのため、環境変数と特殊な変数が同じ名前の場合、環境変数が使用されます。

環境変数には、すべての英数字とアンダースコア '_' 記号が使用できます。変数名は、許可される最後の文字までです。

環境変数でファイルやディレクトリのパスなどを定義すると、以下に説明するように、その値に展開されます。

%VARIABLENAME%

環境変数が定義されている場合、その値に置き換えられます。この変数は大文字と小文字を区別せずにマッチングされます。変数が定義されていない場合、文字列 '%VARIABLENAME%' が結果となります。

$VARIABLENAME

環境変数が定義されている場合、その値に置き換えられます。この変数は、Unix では大文字と小文字を区別して、Windows では大文字と小文字を区別せずにマッチングされます。変数が定義されていない場合、空の文字列に置き換えられます。

${VARIABLENAME}text

'$VARIABLENAME' に定義された値に 'text' を付加したものに置き換えられます。

${VARIABLENAME:-default_value}

'$VARIABLENAME' に定義された値に置き換えられます。変数が設定されていない場合は 'default_value' に置き換えられます。

Tectia 固有の特殊な変数は以下の通りです。

%U または %username%

現在ログインしているユーザ名に置き換えられます。

%username-without-domain%

現在ログインしているユーザ名の短縮形、つまりドメイン部分を除いたものに置き換えられます。Windows で使用できます。

%G または %groupname%

現在ログインしているユーザのグループ名に置き換えられます。

%D または %homedir%

現在ログインしているユーザに定義されているホーム・ディレクトリに置き換えられます。

%IU または %userid%

現在ログインしているユーザに定義されているユーザ ID に置き換えられます。

%IG または %groupid%

現在ログインしているユーザに定義されているグループ ID に置き換えられます。

特殊な変数は、 $username のように Unix のフォーマットで入力することもできます。

文書型宣言とルート・エレメント

接続ブローカーの設定ファイルは妥当な XML ファイルであり、文書型宣言から始まります。

設定ファイルのルート・エレメントは secsh-broker です。これには general、 default-settings、 profiles、 static-tunnels、 gui、 及び logging のエレメントを含めることができます。

空の設定ファイルの例を以下に示します。

```
<!DOCTYPE secsh-broker SYSTEM "ssh-broker-ng-config-1.dtd">
<secsh-broker version="1.0">
  <general />
  <default-settings />
  <profiles />
  <static-tunnels />
  <gui />
  <logging />

</secsh-broker>
```

general エレメント

general エレメントには、使用する暗号化ライブラリや鍵ストアなどの設定が含まれます。

general エレメントには 0 個または 1 個の crypto-lib、 cert-validation、 key-stores、 user-config-directory、 protocol-parameters のエレメントのインスタンス、及び複数の known-hosts エレメントのインスタンスを含めることができます。

crypto-lib

このエレメントでは、使用する暗号化ライブラリ・モードを選択します。暗号化ライブラリは、標準バージョン (standard) または FIPS 140-2 認証バージョン (fips) のいずれかを使用できます。ライブラリ名は mode 属性の値として指定します。デフォルトでは標準の暗号化ライブラリが使用されます。FIPS モードでは、OpenSSL の暗号化ライブラリが使用されます。

FIPS モードは、グローバルまたはユーザ設定ファイル (またはその両方) で指定されている場合に使用されます。

```
<crypto-lib mode="fips" />
```

FIPS ライブラリが検証またはテストされたプラットフォームのリストについては、『Tectia Client/Server Product Description』を参照してください。

cert-validation

このエレメントでは、リモート・サーバの認証証明書を検証するために使用される、公開鍵基盤 (PKI) の設定を定義します。このエレメントには end-point-identity-check、 default-domain、 http-proxy-url、 socks-server-url、 cache-size、 max-crl-size、 external-search-timeout、 max-ldap-response-length、 ldap-idle-timeout、及び max-path-length の属性を指定できます。

`end-point-identity-check` 属性では、サーバ・ホスト証明書に指定されているサブジェクト名またはサブジェクト代替名 (DNS アドレス) に対して、クライアントがサーバのホスト名または IP アドレスを検証するかどうかを指定します。デフォルト値は `yes` です。`no` に設定した場合、サーバ・ホスト証明書のフィールドは検証されず、有効期間と CRL チェックのみに基づいて証明書が受け入れられます。

警告

`end-point-identity-check="no"` を設定することはセキュリティ・リスクです。これにより、サーバ・ホスト証明書の発行元と同じ、信頼できる認証局 (CA) から発行された証明書を持っている人は誰でも、サーバに対して中間者攻撃を行えるようになります。

その代わりに `ask` に設定すると、サーバのホスト名が証明書のホスト名と一致しない場合に、接続をキャンセルするか継続するかをユーザが決定できます。

`default-domain` 属性は、エンド・ポイント同一性チェックが有効な場合に使用できます。この属性ではリモート・システム名のデフォルトのドメイン部分を指定し、システム名のベース部分のみしか入手できない場合に使用されます。 `default-domain` は、システム名にドット (.) が含まれていない場合に付加されます。

デフォルト・ドメインが指定されていない場合でも、エンド・ポイント同一性チェックは短いホスト名で機能します。たとえば、ユーザが短いホスト名だけを指定してホスト "`rock`" への接続を試みた場合、証明書に完全な DNS アドレス "`rock.example.com`" が含まれていれば、接続は開かれ、Tectia Client は "`rock`" への接続を許可しているという内容の警告を発します。

証明書の有効性に関する LDAP または OCSP での問い合わせのために、 `http-proxy-url` 属性では HTTP プロキシを、 `socks-server-url` 属性では SOCKS サーバを定義します。

属性の値としてサーバのアドレスが与えられます。アドレスのフォーマットは `socks://username@socks_server:port/network/netmask,network/netmask ...` (SOCKS サーバの場合) または `http://username@proxy_server:port/network/netmask,network/netmask ...` (HTTP プロキシの場合) です。

たとえば、 `192.196.0.0` (16 ビット・ドメイン) 及び `10.100.23.0` (8 ビット・ドメイン) のネットワークには直接接続し、それ以外の接続では、ホスト `socks.ssh.com` とポート `1080` の SOCKS サーバを使用するには、 `socks-server-url` を以下のように設定します。

```
"socks://mylogin@socks.ssh.com:1080/192.196.0.0/16,10.100.23.0/24"
```

`cache-size` 属性では、証明書及び CRL 用のメモリ内・キャッシュの最大サイズ (メガバイト単位) を定義します。許容値の範囲は 1 ～ 512 で、デフォルト値は 300 MB です。

`max-crl-size` 属性では CRL の最大許容サイズ (メガバイト単位) を定義します。大きな CRL を処理すると、かなりの量のメモリと処理能力を消費するので、環境によってはサイズを制限することが賢明です。許容値の範囲は 1 ～ 512 で、デフォルト値は 50 MB です。

external-search-timeout 属性では、CRL や証明書を外部 HTTP 及び LDAP から検索する時間制限 (秒単位) を定義します。許容値の範囲は 1 ～ 3600 秒で、デフォルト値は 60 秒です。

max-ldap-response-length 属性では LDAP 応答の最大許容サイズ (メガバイト単位) を定義します。許容値の範囲は 1 ～ 512 で、デフォルト値は 50 MB です。

ldap-idle-timeout 属性では LDAP 接続のアイドル・タイムアウトを定義します。検証エンジンは LDAP 接続を保持し、次回の検索に再利用します。接続はLDAP アイドル・タイムアウトに達した後のみ、閉じられます。許容値の範囲は 1 ～ 3600 秒で、デフォルトのアイドル・タイムアウトは 30 秒です。

max-path-length 属性は、証明書を検証するときの認証パスの長さを制限します。この属性を使用すると、階層が深い PKI や、その PKI と他の PKI との相互認証が多い場合に、パスを保護したり、パスが長くなりすぎないように最適化したりできます。属性を使用するには、証明書の検証で使用されるパスの上限を把握しておく必要があります。例:

```
<cert-validation max-path-length="6">
  <ldap-server address="ldap://myldap.com" port="389" />
  <dod-pki enable="yes" />
  <ca-certificate name="CA 1" file="ca-certificate1.crt" />
</cert-validation>
```

この例では、パスはエンドエンティティ証明書とルート CA 証明書を含む 6 つの証明書に制限されています。指定しない場合、デフォルト値は 10 です。証明書の検証で遭遇するパスの最大長がわかっている場合は、値を小さくして検証を最適化します。

cert-validation エレメントは、複数の ldap-server、 ocsp-responder、 crl-prefetch エレメント、1 つの dod-pki エレメント、及び複数の ca-certificate 及び key-store エレメントを含めることができます。エレメントは一覧の順序でなければなりません。

ldap-server

このエレメントでは、検証中の証明書の発行者名に基づいて、CRL 及び/または下位の CA 証明書を取得するために使用する LDAP サーバの address 及び port を指定します。複数の LDAP サーバを指定するには、複数の ldap-server エレメントを使用します。

証明書に定義された CRL 配布ポイントが存在すれば、CRL はそこから自動的に取得され、検証されます。

port のデフォルト値は 389 です。

ocsp-responder

このエレメントでは URL 形式の OCSP (オンライン証明書状態プロトコル) レスポンダ・サービスのアドレスを url 属性で指定します。複数の OCSP レスポンダを指定するには、複数の ocsp-responder エレメントを使用します。

証明書に有効な Authority Info Access 拡張と OCSP レスポンダの URL がある場合、この設定の代わりにそれが使用されます。OCSP 検証が成功するためには、エンドエンティティ証明書と OCSP レスポンダ証明書の両方が同じ CA から発行されている必要があります。

オプションで `validity-period` (秒単位) を設定できます。この間、同じ証明書に対する新たな OCSP の問い合わせは行われず、古い結果が使用されます。デフォルトの有効期限は `0` です (毎回新しい問い合わせが行われます)。

crl-prefetch

このエレメントは、指定された URL から定期的に CRL をダウンロードするように Tectia Client に指示します。 `url` の値には、LDAP または HTTP の URL を指定することも、ローカル・ファイルを参照させることもできます。ファイル形式はバイナリ DER または base64 でなければならず、PEM はサポートされていません。

ローカル・ファイル・システムから CRL をダウンロードするには、以下のフォーマットでファイルの URL を定義します。

```
file:///absolute/path/name
```

LDAP サーバから CRL をダウンロードするには、以下のフォーマットで LDAP の URL を定義します。

```
ldap://ldap.server.com:389/CN=Root%20CA,
       OU=certification%20authorities,DC=company,
       DC=com?certificaterevocationlist
```

CRL をダウンロードする頻度を指定するには、 `interval` 属性を使用します。デフォルトは `3600` 秒です。

dod-pki

アメリカ合衆国国防総省公開鍵基盤 (DoD PKI) のコンプライアンス要件の 1 つに、証明書の鍵用途でデジタル署名ビットを設定することがあります。鍵の用途に電子署名を必須にするには、 `enable` 属性の値を `yes` に設定します。デフォルトは `no` です。

ca-certificate

このエレメントでは、サーバ認証に使用される認証局 (CA) を定義します。指定できる属性は `name`、 `file`、 `disable-crls`、及び `use-expired-crls` の 4 つです。

`name` 属性には CA の名前を含める必要があります。

このエレメントには、 `file` 属性の値として X.509 CA 証明書ファイルのパスを含めるか、または base64 エンコードされたアスキー・ブロックとして証明書を含める必要があります。

`disable-crls` 属性を `yes` に設定することにより、CRL チェックを無効にできます。デフォルトは `no` です。

失効した CRL は、 `use-expired-crls` 属性に数値 (秒単位) を設定することで使用できます。デフォルトは `0` です (失効した CRL を使用しません)。

key-store

このエレメントでは、サーバ認証のために外部鍵ストアに保存される CA 証明書を定義します。現在は z/OS 上で、System Authorization Facility (SAF) に保存されている CA 証明書にのみ使用されています。

証明書検証の設定例を以下に示します。

```
<cert-validation  end-point-identity-check="yes"
                  default-domain="example.com"
                  http-proxy-url="http://proxy.example.com:8080">
  <ldap-server    address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090"
                  validity-period="0" />
  <crl-prefetch   url="file:///full.path.to.crlfile"
                  interval="1800" />
  <dod-pki        enable="no" />
  <ca-certificate name="ssh_ca1"
                  file="ssh_ca1.crt"
                  disable-crls="no"
                  use-expired-crls="100" />
</cert-validation>
```

key-stores

このエレメントでは、ユーザ公開鍵及び証明書認証に関する設定を定義します。

`<general>` エレメントの下に 1 つの `<key-stores>` インスタンスがあり、そこに任意の数の `<key-store>`、 `<user-keys>`、及び `<identification>` エレメントを指定できます。エレメントの順番は自由です。

エレメントの値を定義するときは、特殊な変数や環境変数を使用できます。使用できる変数と、それぞれどのように展開されるのかについて、以下に示します。

- `%U` = `%USERNAME%` = ユーザ名
- `%USERNAME-WITHOUT-DOMAIN%` = ドメイン部を除いたユーザ名
- `%IU` = `%USERID%` = ユーザ ID (Windows 以外の場合)
- `%IG` = `%GROUPID%` = ユーザ・グループ ID (Windows 以外の場合)
- `%D` = `%HOMEDIR%` = ユーザのホーム・ディレクトリ
- `%G` = `%GROUPNAME%` = ユーザのデフォルト・グループの名前

環境変数も、それぞれの現在の値で置き換えられます。たとえば、 `$HOME` または `%HOME%` という文字列を使用して、ユーザのホーム・ディレクトリに展開できます (環境変数 `HOME` が設定されている場合)。

注意

短いエイリアス名 (例: `%U`) は大文字と小文字が区別され、長いエイリアス名 (例: `%USERNAME%`) は大文字と小文字が区別されません。

key-store

key-store エレメントはそれぞれ、1 つの鍵ストア・プロバイダを設定します。 key-stores/key-store エレメントでは type 及び init の属性を指定できます。

type 属性は鍵ストア・タイプです。現在サポートされているタイプは "mscapi"、"pkcs11"、 "software"、及び "zos-saf" です。

init 属性は key-store-provider 固有の初期化情報です。初期化文字列には、前述の **key-stores** で説明した特殊文字列を含めることができます。

鍵ストアの設定例については、「鍵ストアの設定例」tを参照してください。

user-keys

user-keys エレメントを使用すると、ユーザ鍵のデフォルト・ディレクトリを上書きできます。 user-keys エレメントでは次の属性を指定できます。

directory 属性では、ユーザの秘密鍵が保存されるディレクトリを定義します。フル・パスを入力します。

passphrase-timeout で属性は、パスフレーズで保護された秘密鍵がタイムアウトし、ユーザが再びパスフレーズを入力しなければならない時間 (秒単位) を定義します。デフォルトは 0 です。これは、パスフレーズがタイムアウトしないことを意味します。このエレメントの値は、 passphrase-idle-timeout の値よりも長くなければなりません。

デフォルトでは、ユーザがパスフレーズの入力に成功すると、接続ブローカーはパスフレーズで保護された秘密鍵を開放したままにします。これは、パスフレーズのタイムアウト・オプションで変更できます。 passphrase-timeout が設定されている場合、秘密鍵はタイムアウトが終了するまで開放された状態 (パスフレーズのプロンプトが表示されずに使用できる状態) になります。 passphrase-timeout 属性は、ハード・タイムアウトを設定します。このタイムアウトは、鍵が開放されたときに一度だけ設定され、鍵が複数回使用された場合でもリセットされません。

passphrase-idle-timeout 属性では、パスフレーズで保護された秘密鍵が、ユーザがその鍵にアクセスしたり、使用したりしない限りタイムアウトするまでの時間 (秒単位) を定義します。鍵にアクセスするたび、 passphrase-idle-timeout はリセットされます。デフォルトは 0 です。これは、パスフレーズが絶対にタイムアウトしないことを意味します。

両方のタイムアウトを同時に設定できますが、アイドル・タイムアウトがハード・タイムアウトより長く設定されている場合、アイドル・タイムアウトには全く効果がないことに注意してください。

rotation-period 属性では、鍵がローテートされるまでの時間 (秒単位) を定義します。期間の定義には m、 minutes、 h、 hours、 d、及び days のサフィックスを使用できます。

identification

identification エレメントを使用すると、ユーザ鍵を定義する identification ファイルのデフォルトの場所を上書きできます。 identification エレメントでは次の属性を指定できます。

`file` 属性では identification ファイルの場所を指定します。フル・パスを入力します。

`base-path` 属性では、identification ファイルがユーザの秘密鍵の保存場所であると期待するディレクトリを定義します。このエレメントを使用すると、identification ファイルのデフォルトの相対パスの解釈 (identification ファイルのディレクトリからの相対パス) を上書きできます。

`passphrase-timeout` 属性では、ユーザがパスフレーズを再入力しなければならない時間 (秒単位) を定義します。デフォルトは `0` です。これは、パスフレーズが再度要求されないことを意味します。

`passphrase-idle-timeout` 属性では、ユーザによる操作がない場合にパスフレーズがタイムアウトするまでの時間 (秒単位) を定義します。デフォルトは `0` です。これは、パスフレーズがタイムアウトしないことを意味します。

タイムアウトの設定は、 identification ファイルにリストアップされている秘密鍵にのみ影響します。

strict-host-key-checking

注意

このエレメントは Tectia Client バージョン 6.1.4 から非推奨です。

このエレメントは下位互換性のために設定でサポートされており、 `default-settings` または `profiles/profile` にある `server-authentication-methods/auth-server-publickey` エレメントの `policy` 属性が定義されていない場合にのみ使用されます。この場合、ホスト鍵ポリシーは、このオプションと `host-key-always-ask` 及び `accept-unknown-host-keys` オプションの値に基づいて解釈されます。詳細については、 **auth-server-publickey** を参照してください。

host-key-always-ask

注意

このエレメントは Tectia Client バージョン 6.1.4 から非推奨です。

このエレメントは下位互換性のために設定でサポートされており、 `default-settings` または `profiles/profile` にある `server-authentication-methods/auth-server-publickey` エレメントの `policy` 属性が定義されていない場合にのみ使用されます。この場合、ホスト鍵ポリシーは、このオプションと `strict-host-key-checking` 及び `accept-unknown-host-keys` オプションの値に基づいて解釈されます。詳細については、 **auth-server-publickey** を参照してください。

accept-unknown-host-keys

注意

このエレメントは Tectia Client バージョン 6.1.4 から非推奨です。

このエレメントは下位互換性のために設定でサポートされており、 `default-settings` または `profiles/profile` にある `server-authentication-methods/auth-server-publickey` エレメントの `policy` 属性が定義されていない場合にのみ使用されます。この場合、ホスト鍵ポリシーは、このオプションと `strict-host-key-checking` 及び `host-key-always-ask` オプションの値に基づいて解釈されます。詳細については、 **auth-server-publickey** を参照してください。

警告

このオプションを有効にする前に、慎重に検討してください。ホスト鍵のチェックを無効にすると、中間者攻撃に対して脆弱になります。

user-config-directory

このエレメントを使用すると、ユーザ固有の設定ファイルの保存場所をデフォルト以外の場所に変更できます (デフォルトの場所は、 Unix では `$HOME/.ssh2/`、 Windows では "`%APPDATA%\SSH`")。たとえば、クライアント側の設定をすべて一元管理する場合に使用できます。

このエレメントがグローバル設定ファイルに追加されると、接続ブローカーは、定義された場所にある以下のユーザ固有のファイルを読み込みます。

- ユーザの鍵ファイル
- ユーザ自身の設定ファイル
- ユーザの既知のホスト鍵
- ユーザの random_seed ファイル
- Windows GUI プロファイル・ファイル: `1.ssh2, 2.ssh2`
- **sftpg3** クライアントの起動バッチ・ファイル: `ssh_sftp_batch_file`

注意

接続ブローカーの設定で `user-config-directory` の設定を変更する前に、既存の SSH アプリケーションをすべて停止してください。

`user-config-directory` の設定は、Tectia Client や Tectia ConnectSecure など、同じホスト上で動作するすべての Tectia 製品に影響します。

`user-config-directory` オプションには `path` 属性があります。その値はディレクトリ・パスであるか、または以下のいずれかの変数です。

- `%U`: ユーザ名
- `%username%`: ユーザ名
- `%username-without-domain%`: ドメイン定義のないユーザ名
- `%D`: ユーザのホーム・ディレクトリ
- `%homedir%`: ユーザのホーム・ディレクトリ

- %USER_CONFIG_DIRECTORY%: ユーザ固有の設定ディレクトリ
- %IU: ユーザの ID、Unix の場合のみ
- %userid%: ユーザの ID、Unix の場合のみ
- %IG: グループ ID、Unix の場合のみ
- %groupid%: グループ ID、Unix の場合のみ

デフォルトは %USER_CONFIG_DIRECTORY% です。この変数はユーザ固有の設定ディレクトリ (Unix では $HOME/.ssh2、 Windows では %APPDATA%\SSH) を参照します。その他の設定では %USER_CONFIG_DIRECTORY% 変数は使用できません。

file-access-control

Unix でこのエレメントを使用すると、グローバル設定ファイルとユーザ固有の設定ファイル、及び秘密鍵ファイルに定義されたファイルのアクセス・パーミッションのチェックを有効にできます。パーミッションが期待通りでない場合、接続ブローカーは起動を拒否するか、または特定の秘密鍵の使用を拒否します。

デフォルトでは、この設定は無効になっています。Windows では、このエレメントは効果がありません。

グローバル設定ファイルとユーザの設定ファイルの両方に file-access-control エレメントが設定されている場合と、どちらか一方だけに設定されている場合では、ファイル・パーミッションのチェックの方法が異なります。詳細については、以下の表を参照してください。

表A.1 file-access-control の効果の違い

設定の場所:		パーミッションがチェックされる場所:		
グローバル設定	ユーザの設定	グローバル設定	ユーザの設定	秘密鍵ファイル
yes	yes / -	チェックされる	チェックされる	チェックされる
yes	no	チェックされる	チェックされる	チェックされない
no / -	yes	チェックされない	チェックされる	チェックされる
no / -	no / -	チェックされない	チェックされない	チェックされない

表中で、 "no" は file-access-control enable="no" を意味します。 "-" 記号は、その設定がファイル内で全く定義されていないことを意味します。

ファイルのアクセス・パーミッションをチェックする場合、以下の制御が適用されます。

- グローバル設定ファイルに期待されるパーミッション: すべてに読み取り権限、ユーザとグループにのみ書き込み権限。パーミッションがこれ以上広がると、接続ブローカーは起動しません。

- ユーザ設定ファイルに期待されるパーミッション: ユーザのみが読み取りと書き込みの権限を持ちます。パーミッションがこれ以上広がると、接続ブローカーは起動しません。

- 秘密鍵ファイルに期待されるパーミッション: ユーザのみが読み取りと書き込みの権限を持ちます。パーミッションがこれ以上広がると、チェックを通らない鍵は無視されます。

protocol-parameters

このエレメントには、パフォーマンスの調整に使用できる、プロトコル固有の値が含まれます。このエレメントは、非常に特殊な環境でのみ使用する必要があります。通常の場合は、デフォルト値を使用してください。

`threads` 属性を使用すると、プロトコル・ライブラリが使用するスレッド数（ファスト・パス・ディスパッチャ・スレッド）を定義できます。この属性を使用すると、5 つ以上の CPU を持つシステムにおいて、プロトコル内で同時に行う暗号変換を増やすことができます。値が 0 に設定されている場合、デフォルト値が使用されます。

`threads` 属性の例を以下に示します。

```
<protocol-parameters threads="8" />
```

known-hosts

このエレメントを使用すると、既知のサーバ・ホストのホスト鍵を保存する場所を指定し、ホスト鍵ファイルの保存フォーマットを定義できます。`known-hosts` ディレクトリが指定されていない場合、既知のホスト鍵はデフォルトのディレクトリに保存されます。デフォルトの場所については、「ファイル」を参照してください。z/OS の場合 (に限り)、このエレメントには `key-store` エレメントを含めることができます。

このエレメントは以下の目的で使用できます。

- 既知のサーバ・ホストの公開鍵データまたは公開鍵ファイルを含むデフォルトでないディレクトリを指定する。

- 既知のサーバ・ホストの公開鍵データを含む OpenSSH 形式の known_hosts ファイルにの、デフォルトでない場所を指定する。

- (z/OS の場合) 既知のサーバ・ホストの証明書を含む SAF 鍵ストアを指定する。

サーバ・ホスト鍵は、設定で指定されている順番に従って、`known-hosts` パスで検索されます。新しいホスト鍵を保存するときには、最後に定義された `known-hosts` エレメントの設定が使用されます。

`known-hosts` ファイルの設定を定義すると、デフォルトの OpenSSH ファイルが上書きされます。そのため、デフォルトの OpenSSH の場所と、設定で指定されている他の場所の両方を接続ブローカーで使用する場合は、すべての場所を個別に指定する必要があります。

known-hosts エレメントは複数定義でき、それぞれに 1 つまたは複数の path、 directory、file、 及び filename-format 属性を含めることができます。

path 属性には、値として known-hosts ファイルまたはディレクトリのフル・パスが必要です。例:

```
<known-hosts path="/u/username/.ssh/known_hosts" />
<known-hosts path="/etc/ssh2/hostkeys" />
<known-hosts path="/u/username/.ssh2/hostkeys" />
<known-hosts path="/h/username/hostkeys" filename-format="plain" />
```

directory 属性は、既知のホスト鍵がデフォルトでないディレクトリに保存されることを定義するために使用します。ディレクトリの完全なパスを値として入力します。定義されたディレクトリが存在しない場合、最初の接続の試みの際に作成されます。その場所に同じ名前のファイルが見つかった場合、接続は行われますが、ホスト鍵は保存されず、それについての警告がユーザに与えられます。 filename-format 属性を directory の設定と一緒に使用すると、ホスト鍵ファイルを保存するフォーマットを定義できます。
directory 属性の例を以下に示します。

```
<known-hosts directory="<path_to_dir>/MyKEYS"
             filename-format="plain" />
```

新しい既知のホスト鍵を保存するときには、設定ファイルの最後の known-hosts エレメントで定義されている path または directory (どちらか存在する方) が使用されます。最後の known-hosts エレメントに両方の属性が存在する場合、 directory 属性で指定されている場所が使用されます。

file 属性は、OpenSSH 形式の known_hosts ファイルを指すために使用します。ファイルの完全なパスを値として入力します。その場所に同じ名前のディレクトリが見つかった場合はエラーとみなされ、接続の試みは失敗します。 known-hosts エレメントに file 属性しか含まれておらず、定義された OpenSSH の known_hosts ファイルが存在する場合、受信したホスト鍵はまず、定義されているファイル内で検索され、そこで見つからない場合は、Tectia 固有のデフォルトの場所で検索が継続されます。

file 属性の例を以下に示します。

```
<known-hosts file="<path_to_file>/.ssh2/openSSH_keys" />
```

以下のように file または path 属性を空にすると、OpenSSH の known_hosts ファイルの処理が無効になります。

```
<known-hosts file="" />
or
<known-hosts path="" />
```

filename-format 属性では、新しいホスト鍵ファイルを保存するフォーマットを定義します。 filename-format 属性は、最後に指定された known-hosts エレメントとデフォルトのディレクトリにのみ関係します。

filename-format 属性では hash (デフォルト)、 plain、及び default (ハッシュと等しい) の値を指定します。

値を `hash` にすると、ホスト鍵ファイルが`keys_<hash>`のフォーマットで保存されます (例: "`keys_182166d2efe5a134d3fb948646e0b48f780bff6c`")。

値を `plain` にすると、ファイル名のフォーマットは `key_<port>_<hostname>.pub` になります。ここで、 `<port>` は Secure Shell サーバが動作しているポート、 `<hostname>` はそのサーバに接続するときに使用するホスト名です (例: "`key_22_my.example.com.pub`")。

`<known-hosts filename-format="plain" />` を設定すると、次の `known-hosts` エレメントの、またはその他の `known-hosts` エレメントが存在しない場合はデフォルトの保存場所の、ホスト鍵ファイルの保存フォーマットが変更されます。

同じ `known-hosts` エレメントを使用してホスト鍵の場所を複数定義しており、そのうちのいくつかはプレーン・フォーマットで鍵を保存する場合は、 `filename-format="default"` を最後のオプションとして代用できます。

ホスト鍵の保存フォーマットの詳細については、4.2.1 を参照してください。

key-store

このエレメントでは、既知のサーバ・ホストの証明書のための外部鍵ストアを定義します。現在は z/OS 上で、System Authorization Facility (SAF) に保存されているサーバ証明書にのみ使用されています。

extended

このエレメントは将来の使用のために予約されています。

鍵ストアの設定例

ソフトウェア・プロバイダの例

ソフトウェア・プロバイダでは、標準的な Secure Shell v2 またはレガシーの OpenSSH フォーマットでディスクに保存された鍵ペアと、ネイティブの X.509、 PKCS #7、 及び PKCS #12 フォーマットで保存された X.509 証明書を扱えます。

1 つの鍵ファイル (例: `/u/exa/keys/enigma` や `/etc/my_key`) を追加するには、秘密鍵ファイルと公開鍵ファイルの両方を指定します。

```
<key-stores>
  <key-store type="software"
             init="key_files(/u/exa/keys/enigma.pub,/u/exa/keys/enigma)" />
  <key-store type="software"
             init="key_files(/etc/my_key.pub,/etc/my_key)" />
</key-stores>
```

特定のディレクトリにあるすべての鍵 (例: `/u/exa/keys` と `/etc/keys` にあるすべての鍵) を追加するには、以下のように設定します。

```
<key-stores>
  <key-store type="software"
             init="directory(path(/u/exa/keys))" />
  <key-store type="software"
             init="directory(path(/etc/keys))" />
```

```
</key-stores>
```

PKCS #11 プロバイダの例

PKCS #11 プロバイダは、PKCS #11 トークン (スマート・カードや USB トークンなど) に保存されている鍵と証明書を扱います。

PKCS プロバイダの DLL パスと、すべてまたは特定のスロットを指定します。すべてのスロットを指定する場合の例:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/lib/pkcs.so),slots(all)" />
</key-stores>
```

sesam という名称のスロット 1 つを指定する場合の例:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/local/lib/pkcs.so),slots(sesam)" />
</key-stores>
```

default-settings エレメント

default-settings エレメントでは、デフォルト接続に関連する設定を定義します。これらの設定は、プロファイル固有の設定で上書きできます。「 profiles エレメント」を参照してください。

default-settings エレメントには以下のエレメントのインスタンスを 0 または 1 つ、一覧の順に含むことができます: ciphers、 macs、 kexs、 hostkey-algorithms、 rekey、 authentication-methods、 hostbased-default-domain、 compression、 proxy、 idle-timeout、 tcp-connect-timeout、 keepalive-interval、 exclusive-connection、 server-banners、 forwards、 extended、 remote-environment、 server-authentication-methods、 authentication-success-message、 sftpg3-mode、 terminal-selection、 terminal-bell、 close-window-on-disconnect、 quiet-mode、 checksum、 及び address-family

default-settings エレメントでは以下の属性を 1 つ指定できます。

user 属性を使用すると、リモート・サーバに接続するときに使用するデフォルトのユーザ名を定義できます。 user 属性の値には、以下のいずれかを指定できます。

- 接続プロファイルの設定または接続の試行で別のユーザ名が指定されていない限り、接続時に使用される一般的なユーザ名。ユーザ名は大文字と小文字を区別して扱われます。
- "%USERNAME%" を使用すると、現在ログインしているユーザのユーザ名を適用できます。
- このオプションを使用しても、空のままにすると、接続ブローカーはユーザにユーザ名の入力を求めます。

default-settings エレメントには以下のエレメントを含めることができます。

ciphers

このエレメントでは、クライアントがサーバに提示する暗号を定義します。 ciphers エレメントには複数の cipher エレメントを含めることができます。

暗号は、指定された順に試行されます。

cipher

このエレメントでは、クライアントがデータ暗号化のために要求する暗号を `name` で選択します。

サポートされる暗号の一覧は F.1 を参照してください。

```
<ciphers>
  <cipher name="aes128-cbc" />
  <cipher name="AEAD_AES_256_GCM" />
</ciphers>
```

macs

このエレメントでは、クライアントがサーバに提示する MAC を定義します。 `macs` エレメントには複数の `mac` エレメントを含めることができます。

MAC は、指定された順に試行されます。

mac

このエレメントでは、クライアントがデータ完全性の検証のために要求する MAC を `name` で選択します。

サポートされる MAC の一覧は F.3 を参照してください。

```
<macs>
  <mac name="hmac-sha2-512" />
</macs>
```

kexs

このエレメントでは、クライアントがサーバに提示する鍵交換方式 (KEX) を定義します。 `kexs` エレメントには複数の `kex` エレメントを含めることができます。

鍵交換は、指定された順に試行されます。

kex

このエレメントでは、クライアントが鍵交換方式のために要求する鍵交換を `name` で選択します。

サポートされるクラシック及び PQC 鍵交換方式の一覧は F.2 を参照してください。

```
<kexs>
   <kex name="ecdh-nistp521-kyber1024-sha512@ssh.com" />
   <kex name="ecdh-sha2-nistp256" />
</kexs>
```

hostkey-algorithms

このエレメントでは、サーバ認証に使用されるホスト鍵署名アルゴリズムを定義します。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方の設定ファイル

で定義されているものです。このようにすることで、SHA-2　のような特定のアルゴリズムのみを使用するようにサーバから強制できます。 hostkey-algorithms エレメントには複数の hostkey-algorithm エレメントを含めることができます。

ホスト鍵アルゴリズムは、指定された順に試行されます。例外: サーバのホスト鍵がすでにクライアントのホスト鍵ストアに存在する場合は、そのアルゴリズムが優先されます。

hostkey-algorithm

このエレメントでは、ホスト鍵または証明書によるサーバ認証に使用されるホスト鍵署名アルゴリズムを name で選択します。

サポートされるホスト鍵署名アリゴリズムは F.4 を参照してください。

```
<hostkey-algorithms>
   <hostkey-algorithm name="rsa-sha2-256" />
   <hostkey-algorithm name="ssh-dss-sha512@ssh.com" />
</hostkey-algorithms>
```

rekey

このエレメントでは、鍵交換が再び行われるまでに転送される bytes 数を指定します。値を "0" にすると、鍵更新の要求はオフになります。ただし、これによってサーバによる鍵更新の要求が妨げられることはありません。デフォルトは 1000000000 (1 GB) です。

```
<rekey bytes="1000000000" />
```

authentication-methods

このエレメントでは、クライアント側コンポーネントが要求する認証方法を指定します。 authentication-methods エレメントには、 auth-hostbased、 auth-password、 auth-publickey、 auth-gssapi、 及び auth-keyboard-interactive をそれぞれ 1 つ含めることができます。また、複数の authentication-method エレメントを指定することもできます。これらのエレメントの並び順は自由です。

認証方法は、 auth-* または authentication-method エレメントの記述順に試行されます。つまり、最も非対話的な方法を最初に配置する必要があります。

複数の対話型認証方法が許可されるように定義されていて、前の方法が失敗した場合、Tectia Client は、それらの方法を順番にサーバに提示します。

authentication-method

このエレメントでは、認証方法を name で指定します。これは下位互換性のために含まれています。代わりに auth-* エレメントを使用して下さい。

許可されている認証方法の名前は gssapi-with-mic、 publickey、 keyboard-interactive、 password、 及び hostbased です。

Tectia Client は Unix プラットフォームでのみホストベースの認証をサポートしています。

```
<authentication-methods>
  <authentication-method name="hostbased" />
  <authentication-method name="gssapi-with-mic" />
  <authentication-method name="publickey" />
  <authentication-method name="keyboard-interactive" />
  <authentication-method name="password" />
</authentication-methods>
```

auth-hostbased

このエレメントでは、ホストベースの認証が使用されることを指定します。

`auth-hostbased` エレメントには `local-hostname` エレメントを含めることができます。

local-hostname

このエレメントでは、ホストベースの認証の際にリモート・サーバに通知されるローカル・ホスト名を、 `name` 属性の値として指定します。

リモート・サーバはクライアント・ホストの公開鍵を特定する際に、クライアント・ホスト名をヒントとして使用できます。この情報は認証結果には大きく影響しませんが、サーバに膨大な数のホスト ID が保存されており、それらをすべて試すことが不可能な場合は、関連するクライアントのホスト鍵を見つけるまでの時間を短縮します。

auth-password

このエレメントでは、パスワード認証が使用されることを指定します。

auth-publickey

このエレメントでは、公開鍵認証が使用されることを指定します。

`auth-publickey` エレメントには `key-selection` エレメントを含めることができます。

`auth-publickey` エレメントには `signature-algorithms` 属性を含めることができます。この属性では、クライアント認証に使用される公開鍵署名アルゴリズムを定義し、カンマ区切りのリストとして指定します。使用されるアルゴリズムは、Tectia Server と接続ブローカーの両方の設定ファイルで定義されているものです。このようにすることで、SHA-2 のような特定のアルゴリズムのみを使用するように強制できます。サポートされているアルゴリズムのリストについては、 **hostkey-algorithm** を参照してください。

```
<authentication-methods>
  <auth-publickey signature-algorithms="ssh-ed25519,ssh-rsa-sha256@ssh.com"/>
</authentication-methods>
```

key-selection

このエレメントでは、ユーザ公開鍵をサーバに提示する際にクライアントが使用する鍵選択ポリシーを指定します。 `policy` 属性では `automatic` (デフォルト) 及び `interactive-shy` の値を指定できます。

`automatic` モードでは、クライアントは以下の順に鍵を試行します。

1. 公開鍵が使用可能で、秘密鍵にパスフレーズがない鍵 (ユーザの操作はなし)
2. 公開鍵が使用可能であるが、秘密鍵にパスフレーズが設定されている鍵 (1 回のパスフレーズの問い合わせ)
3. 公開鍵の取得にはパスフレーズが必要であるが、秘密鍵にはパスフレーズが不要な鍵 (サーバへ提示しようとしている各鍵に対して 1 回のユーザへの問い合わせがあるが、実際の公開鍵ログインではユーザの操作はなし)
4. 残りの鍵。つまり、公開鍵の取得と秘密鍵の取得の両方においてパスフレーズが必要な鍵

`interactive-shy` モードでは、クライアントは自動的には鍵を試行せず、使用可能な鍵の一覧から鍵を選択するようユーザに求めます。選択した鍵による認証に失敗した場合、クライアントは、すでに試行した鍵を一覧から除いて、ユーザに鍵の選択を再度求めます。使用可能な鍵の候補が 1 つしかない場合、ユーザに尋ねることなく自動的にその鍵が試行されます。

`key-selection` エレメントには `public-key` 及び `issuer-name` エレメントを含めることができます。

`key-selection` エレメントには以下を含めることができます。

```
altname-email
altname-upn
extended-key-usage
issuer-name
public-key
subject-name
validity
```

Example key-selection element:

```
<key-selection>
  <issuer-name name="CN=issuingcaname" pattern=".*testca.*" />
  <subject-name name="CN=username" pattern=".*username.*" />
  <extended-key-usage oid="ssh-client" explicit="yes" />
  <extended-key-usage oid="1.3.6.1.4.1.4449.1.2.4.1.3" />
  <validity valid-for="3600" />
  <altname-email name="user@something.com" pattern="user@.*" />
  <altname-upn name="user@something.com" pattern="user@.*" />
</key-selection>
```

altname-email

このエレメントを使用すると、公開鍵認証の際に、プレーンな公開鍵のみを試すか、証明書のみを試すかを指定できます。 `type` 属性では `plain` 及び `certificate` の値を指定できます。デフォルトでは、プレーンな公開鍵と証明書の両方を試します。

subjectAltName の email でユーザ証明書をフィルタリングします。 厳密な一致のためには `name` オプションを、正規表現による一致のためには `pattern` を使用します。

後述の変数による置き換えをサポートします。

altname-upn

subjectAltName の UPN でユーザ証明書をフィルタリングします。 厳密な一致のためには `name` オプションを、正規表現による一致のためには `pattern` を使用します。

後述の変数による置き換えをサポートします。

extended-key-usage

拡張鍵使用法によってユーザ証明書をフィルタリングします。 `oid` オプションを使用して名称またはOIDで指定します。 `explicit` が指定された場合は拡張鍵使用法が証明書に存在しなければなりません。 `explicit` が指定されない場合は拡張鍵使用法は含まれない証明書は一致したとみなされます。

issuer-name

このエレメントを使用すると、ユーザに提示されるユーザ証明書の一覧をフィルタリングできます。クライアント側のユーザ証明書は、その発行者名がサーバが要求したまたは受け入れた証明書発行者と比較されることでフィルタリングされます。 `match-server-certificate` 属性では `yes` 及び `no` の値を指定します。値を `yes` にすると、接続ブローカーはユーザ証明書の発行者名とサーバ証明書の発行者名の一致を試みます。 `no` のオプションは、発行者名をフィルタとして使用しないことを意味します。デフォルトでは、フィルタリングは行われません。

`issuer-name` は、テスト用ロールと管理者ロールなど、ユーザが同じサーバに対して異なるアクセス権を持つ複数の証明書を有している場合に便利です。接続ブローカーは、リモート・ホストで適用可能な適切な証明書を選別するので、ユーザは絞り込まれた証明書の中から正しい証明書を選択できます。

後述の変数による置き換えをサポートします。

public-key

このエレメントを使用すると、公開鍵認証の際に、プレーンな公開鍵のみを試すか、証明書のみを試すかを指定できます。 `type` 属性では `plain` 及び `certificate` の値を指定できます。デフォルトでは、プレーンな公開鍵と証明書の両方を試します。

subject-name

サブジェクト名でユーザ証明書をフィルタリングします。 厳密な一致のためには `name` オプションを、正規表現による一致のためには `pattern` を使用します。

後述の変数による置き換えをサポートします。

validity

ユーザ証明書を残存有効期限でフィルタリングします。 残存有効期限が `valid-for` を超える証明書のみが一致します。 `valid-for` は `3d 18h` のように、's'、 'm'、 'h' 及び 'd' (秒、 分、 時間及び日) の単位をサポートします。

auth-keyboard-interactive

このエレメントでは、認証においてキーボード・インタラクティブ認証方法が使用されることを指定します。

auth-gssapi

このエレメントでは、認証において GSSAPI が使用されることを指定します。

`auth-gssapi` エレメントでは次の属性を指定できます。

`dll-path` 属性では、必要な GSSAPI ライブラリの場所を指定します。この属性が指定されていない場合、ライブラリは複数の共通の場所で検索されます。ライブラリのフル・パスを指定する必要があります (例: `"/usr/lib/libkrb5.so,/usr/lib/libgssapi_krb5.so"`)。

AIX では `dll-path` にアーカイブ・ファイルを含める必要があります (例: `"<path>/libgssapi_krb5.a(libgssapi_krb5.a.so)"`)。ライブラリが共有オブジェクトである場合、または共有オブジェクトから抽出された場合、 `archive(shared_object)` 構文は必要ありません。

Windows では `dll-path` 属性は無視されます。Tectia Client は正しい DLL の場所を自動的に特定します。

`allow-ticket-forwarding` 属性では、Tectia Client が複数の接続にわたって Kerberos チケットを転送することを許可するかどうかを定義します。この属性には `yes` または `no` の値を指定できます。デフォルトは `no` です。

authentication-methods の設定例を以下に示します。

```
<authentication-methods>
  <auth-hostbased>
    <local-hostname name="host.example.com" />
  </auth-hostbased>
  <auth-gssapi allow-ticket-forwarding="yes"/>
  <auth-publickey>
    <key-selection policy="interactive-shy">
      <public-key type="plain" />
    </key-selection>
  </auth-publickey>
  <auth-keyboard-interactive />
  <auth-password>
    <password file="/path/filename" />
  </auth-password>
</authentication-methods>
```

hostbased-default-domain

このエレメントでは、ホストのデフォルトのドメイン名を (`name` として) 指定します。このエレメントは、ホストベースのユーザ認証を使用する際に、クライアント・ホストの完全修飾ドメイン名 (FQDN) がサーバに送信されるようにするために使用します。

デフォルトのドメイン名は、サーバに送信する前に、短いホスト名に付加されます。このような措置が必要なのは、一部のプラットフォーム (Solaris など) では短縮形のホスト名が使用されており、それでは署名が作成できないからです。

デフォルトのドメイン名で許容されるフォーマットは `.example.com` 及び `example.com` (先頭のドットなし) です。例:

```
<hostbased-default-domain name=".example.com" />
```

compression

このエレメントでは、クライアントがデータを圧縮して送信するかどうか (PUT 操作) を指定します。有効にすると、接続を通じて送信されるすべてのデータとその中のすべてのチャネルに、その場で圧縮が適用されます。

圧縮アルゴリズムの名前と圧縮レベルを属性として指定できます。 `name` 属性には `none` (圧縮を使用しない) または現在唯一サポートされているアルゴリズムの `zlib` を定義できます。デフォルトでは圧縮は使用されません。

zlib 圧縮の場合、 `level` 属性には `0` から `9` までの整数を指定できます。圧縮が有効になっていてもレベルが指定されていない (またはレベルが `0` に設定されている) 場合、デフォルトの圧縮レベルは `6` です。

Example: 送信データの最大レベルの圧縮を有効にする場合は、以下のように設定します。

```
<compression name="zlib" level="9" />
```

コマンドライン・ツールでは接続ごとに圧縮を有効にすることもできます。詳細については、sshg3(1)、sftpg3(1)、及び scpg3(1) の man ページを参照してください。

この `compression` 設定は、受信データ (GET 操作) には影響しません。受信データの圧縮は Secure Shell サーバで定義されることに注意してください。Tectia Server は常に圧縮レベル 6 を使用します。

proxy

このエレメントでは、クライアントが接続に使用する HTTP プロキシまたは SOCKS サーバのルールを定義します。属性は`ruleset` の 1 つだけです。

この属性値のフォーマットは、セミコロン (;) で区切られた一連のルールです。各ルールのフォーマットは URL フォーマットに似ています。1 つのルールの中で、接続の種類を最初に指定します。種類は `direct`、 `socks`、 `socks4`、 `socks5`、または `http-connect` (`socks` は `socks4` の同義語) です。この後に、サーバのアドレスとポートを続けます。ポートが指定されていない場合、デフォルトのポート(SOCKS では 1080、HTTP では 80) が使用されます。

アドレスの後に、カンマ (,) で区切られた 0 個以上の条件を指定します。条件では IP アドレスまたは DNS 名を指定できます。

```
direct:///[cond[,cond]...];
socks://server/[cond[,cond]...];
socks4://server/[cond[,cond]...];
socks5://server/[cond[,cond]...];
http-connect://server/[cond[,cond]...]
```

IP アドレス/ポートの条件には、以下のようにアドレス・パターンとオプションのポート範囲を指定します。

```
ip_pattern[:port_range]
```

`ip_pattern` は、以下のいずれかの形式にすることができます。

- 単一の IP アドレス `x.x.x.x`
- `x.x.x.x-y.y.y.y` の形式による IP アドレス範囲
- `x.x.x.x/y` の形式による IP サブネットワーク・マスク

DNS 名の条件は、以下のようにホスト名 ("*" と "?" を含む正規表現が使えます) とポート範囲から構成されます。

```
name_pattern[:port_range]
```

以下に `proxy` エレメントの例を示します。これにより、サーバはループバック・アドレスと `ssh.com` ドメインに直接アクセスし、`*.example` に HTTP CONNECT でアクセスし、その他のすべての接続先に SOCKS4 でアクセスするようになります。

```
<proxy ruleset="direct:///127.0.0.0/8,*.ssh.com;
                http-connect://http-proxy.ssh.com:8080/*.example;
                socks://fw.ssh.com:1080/" />
```

idle-timeout

このエレメントでは、接続を自動的に閉じるまでに許可されるアイドル時間 (すべての接続チャネルが閉じた後) の長さを指定します。`time` は秒単位で指定します。`type` は常に `connection` です。

デフォルトの設定は 5 秒です。長い時間を設定すると、セッション (**sshg3** など) を閉じた後でもサーバへの接続を維持できます。この時間の間は、再認証を行わずにサーバとの新しいセッションを開始できます。時間を 0 (ゼロ) に設定すると、サーバとの最後のチャネルが閉じられたときに、接続がすぐに終了します。

```
<idle-timeout time="5" />
```

tcp-connect-timeout

このエレメントでは TCP 接続のタイムアウトを指定します。この設定を行うと、リモート・ホストがダウンしている場合や到達できない場合、Secure Shell サーバへの接続試行が定義された時間後に停止されます。このタイムアウトはデフォルトのシステム TCP タイムアウトを上書きします。このタイムアウトの設定は、接続プロファイルごとに

(profiles の設定内) または接続ごとに (コマンドラインで) tcp-connect-timeout を定義することで上書きできます。

time は秒単位で指定します。工場出荷時の設定は 5 秒です。値を 0 (ゼロ) にすると、この機能が無効になり、デフォルトのシステム TCP タイムアウトが使用されます。

```
<tcp-connect-timeout time="5" />
```

keepalive-interval

このエレメントでは、Secure Shell サーバにキープアライブ・メッセージを送信する間隔を指定します。 time の値は秒単位で指定します。デフォルトの設定は 0 です。これは、キープアライブ・メッセージが無効であることを意味します。

```
<keepalive-interval time="0" />
```

exclusive-connection

exclusive-connection エレメントを使用すると、新しいチャネルごとに新しい接続が開かれるように指定できます。この設定には 1 つの属性 enable があり、値には yes または no を指定します。デフォルトは no です。これは、クライアントから要求された新しいチャネルに、すでに開いている接続が再利用されることを意味します。

server-banners

このエレメントでは、サーバ・バナー・メッセージ・ファイル (存在する場合) をログイン前のユーザに表示するかどうかを定義します。 visible の属性値として yes または no という単語を指定します。デフォルトは yes です。

サーバ・バナー表示しないようにするには以下のように設定します。

```
<server-banners visible="no" />
```

forwards

このエレメントには、クライアント側で X11 またはエージェント転送 (トンネリング) が許可されるかを定義する forward エレメントが含まれます。

forward

このエレメントでは X11 またはエージェント転送の設定を定義します。

type 属性では転送タイプ (x11 または agent) を定義します。 state 属性では転送を on、off、または denied に設定します。転送を denied に設定すると、ユーザはコマンドラインで転送を有効にできません。

X11 転送を禁止し、エージェント転送をグローバルに許可する転送の設定例を以下に示します。

```
<forwards>
  <forward type="x11" state="denied" />
  <forward type="agent" state="on" />
</forwards>
```

X11 転送とエージェント転送の使用に関する詳細については、6.3 及び6.4 を参照してください。

extended

このエレメントは将来の使用のために予約されています。

remote-environment

このエレメントは、クライアント側からサーバ側に渡される環境変数を定義する `environment` エレメントを含みます。その定義に基づいて、コマンド、シェル、またはサブシステムを要求する際にサーバ上で環境変数が設定されます。

サーバは環境変数の設定を制限できます。

environment

このエレメントでは、環境変数の名前と値、及び接続ブローカーがその値を処理すべきかどうかを定義します。指定できる属性は `name`、 `value`、及び `format` です。

リモート環境の設定例を以下に示します。

```
<remote-environment>
  <environment name="FOO" value="bar" />
  <environment name="QUX" value="%Ubaz" format="yes" />
  <environment name="ZAPPA" value="%Ubaz" />
</remote-environment>
```

`value` に `%U` を使用すると、ユーザ名を示すことができます。 `format="yes"` も定義すると、接続ブローカーは `%U` を実際のユーザ名に処理してからサーバに送信します。

上記の例でユーザ名を `joedoe` とします。サーバが環境変数の設定を許可している場合、この設定例では以下の環境変数がサーバ側で設定されることになります。

```
FOO=bar
QUX=joedoebaz
ZAPPA=%Ubaz
```

設定ファイルにあるリモート環境の設定を上書きするには、 **sshg3** コマンドを使用し、`--remote-environment` または `--remote-environment-format` の引数をコマンドライン・クライアントで指定します。

コマンドライン・オプションの詳細については、sshg3(1) を参照してください。

server-authentication-methods

この `server-authentication-methods` エレメントを使用すると、接続ブローカーがサーバ認証において特定の方法のみを使用することを強制できます。このエレメントには `auth-server-publickey` 及び `auth-server-certificate` エレメントを (それぞれ 1 つずつ) 含めることができます。これらのエレメントの並び順は自由です。

`auth-server-certificate` のみが指定された場合、サーバ証明書が必要です。サーバ証明書を受信しない場合、接続に失敗します。

auth-server-publickey のみを指定する場合、(プレーンの) サーバ公開鍵が必要です。サーバ公開鍵を受信しない場合、接続に失敗します。

auth-server-certificate と auth-server-publickey の両方が指定された場合、サーバ証明書があればそれが使用されます。サーバ証明書がなければ、サーバ公開鍵が使用されます。

auth-server-certificate

auth-server-certificate エレメントでは、サーバ認証に証明書が使用されることを指定します。

auth-server-publickey

auth-server-publickey エレメントでは、サーバ認証にホスト公開鍵が使用されることを指定します。

注意

バージョン 6.1.4 でホスト鍵ポリシーの設定が変更され、 auth-server-publickey エレメントで定義されるようになりました。

このエレメントには、未知のサーバ・ホスト鍵をどのように処理するかを定義する、 policy 属性があります。この属性には以下の値を指定できます。

- strict: ホスト鍵ストアからホスト鍵が見つかり、一致した場合にのみサーバに接続します。

  ポリシーが strict に設定されている場合、接続ブローカーは接続時にユーザの .ssh2/hostkeys ディレクトリにホスト鍵を追加せず、鍵が変更されたホストへの接続を拒否します。これによって、中間者攻撃に対する最大の防御を発揮します。ただし、常に別の手段でホスト鍵を取得しなければならないことになるので、新しいホストに頻繁に接続する場合は問題になることがあります。

- ask (デフォルト): ホスト鍵ストアからサーバ・ホスト鍵が見つからない場合、ユーザはホスト鍵を受け入れるかどうか問われます。ホスト鍵が変更されている場合は、その旨の警告がユーザに向けて表示され、どのように処理するか問われます。クライアント・アプリケーションがユーザに問い合わせることができない場合 (**sftpg3** がバッチ・モード、 -B など)、接続は切断されます。

- trust-on-first-use または tofu: サーバ・ホスト鍵が見つからない場合、ユーザの .ssh2/hostkeys ディレクトリに鍵が保存されます。ホスト鍵が変更された場合、接続は切断されます。

- advisory: この設定を使用すると、サーバ認証が事実上無効になり、積極的な攻撃者に対して接続が脆弱になります。

  サーバ・ホスト鍵がホスト鍵ストアに見つからない場合、ユーザの操作なしにユーザの .ssh2/hostkeys ディレクトリに鍵が追加されます。ホスト鍵が変更されている場合、ユーザの操作なしに接続が継続されます。ログが有効な場合、インシデントは監査されます。

ポリシーが advisory に設定されている場合、新しいホストからの鍵は、ユーザに承諾を求めることなく自動的に受け入れられ、ホスト鍵データベースに保存されます。ただし、変更されたホスト鍵 (すでに鍵がデータベースに存在するホストからのもの) は保存されず、その接続に対してのみ受け入れられます。

この設定は、接続ブローカーのログが有効になっている場合にのみ使用してください。

警告

ポリシーを advisory に設定する前に、慎重に検討してください。ホスト鍵のチェックを無効にすると、中間者攻撃に対して脆弱になります。

strict 以外のポリシー・モードでは、接続ブローカーに対してログが有効になっていると、Tectia Client は、変更されたホスト公開鍵や新規のホスト公開鍵に関する情報をそのフィンガープリントとともに syslog (Unix の場合) またはイベント・ビューア (Windows の場合) に記録します。

注意

FTP-SFTP 変換を使用する場合、ホスト鍵の受け入れをユーザから要求することはできません。ポリシーを tofu にするか、Secure Shell トンネリング・サーバ及び SFTP サーバのホスト鍵を事前に取得し、サーバの IP アドレスに基づいて保存しておく必要があります。

policy 属性が定義されていない場合、以下の表 A.2 に示すように、旧来の strict-host-key-checking、 host-key-always-ask、及び accept-unknown-host-keys オプションの値に基づいてホスト鍵ポリシーが解釈されます。

注意

バージョン 6.1.4 以降では、ユーザ固有の設定ファイルのホスト鍵ポリシーの設定が、常にグローバル設定ファイルの設定より優先されます。

表A.2 新しいホスト鍵ポリシー (Tectia Client 6.1.4 以降) に合わせた古いホスト鍵ポリシー (Tectia Client 5.0.0 ～ 6.1.3) の解釈

strict-host-key-checking	accept-unknown-host-keys	host-key-always-ask	Policy
-	-	-	ask (default)
有効	-	-	strict
有効	有効	-	strict
有効	有効	有効	ask
有効	-	有効	ask
-	有効	-	trust on first use
-	有効	有効	ask

strict-host-key-checking	accept-unknown-host-keys	host-key-always-ask	Policy
-	-	有効	**ask**

authentication-method

`server-authentication-methods/authentication-method` エレメントでは認証方法を `name` で指定します。このエレメントは下位互換性を保つために含まれています。代わりに `auth-server-*` エレメントを使用して下さい。

```
<server-authentication-methods>
  <authentication-method name="publickey" />
  <authentication-method name="certificate" />
</server-authentication-methods>
```

以下に `server-authentication-methods` エレメントの例を示します。

```
<server-authentication-methods>
  <auth-server-publickey policy="ask" />
  <auth-server-certificate />
</server-authentication-methods>
```

authentication-success-message

この設定では、 `AuthenticationSuccessMsg` メッセージが表示されるかどうかを定義します。 `authentication-success-message` エレメントには `enable` または `dekay` 属性があります。 属性があり、値には `yes` または `no` を指定します。デフォルトは `yes` です。これは、メッセージが出力され、ログに記録されることを意味します。

`Enable` には `yes` または `no` を指定します。 デフォルトは `yes` で、メッセージが表示されログに記録されることを意味します。

`Delay` にはどのくらいの期間認証成功のメッセージが表示されるのかに対応する数値を指定します。デフォルト値は `2` です。`0` を指定すると、メッセージはログに記録されるだけで表示はされません。

disconnect-message

接続を切断したときに表示されるメッセージです。切断メッセージの値は文字列です。このメッセージには以下の変数を複数指定することができます。

- `time`: 切断の時刻
- `random`: 16桁のランダムな16進数
- `random4`: 4桁のランダムな16進数
- `random8`: 8桁のランダムな16進数
- `random16`: `random` と同じです
- `pid`: sshg3 のプロセスID

- broker_pid: ブローカのプロセスID
- conn_id: 接続ID
- session_id: セッションID
- target_host: ターゲット・サーバ名
- target_port: ターゲット・サーバ・ポート

ランダムな値のいずれかが切断メッセージに使用されると、その値は認証成功メッセージの前にユーザに表示されます。接続の前後で値が異なる場合は、接続が何者かにスプーフィングされている可能性があります。

切断メッセージはデフォルトでは無効です。

keyboard-interactive

この設定は文字列の値をとる prefix 属性を含みます。 prefic の値が設定されるとキーボード・インタラクティブのプロンプトの前に表示されます。 prefix 属性の値には以下の変数を含めることができます。

- time
- host
- port
- user
- connid

デフォルトでは prefix の値は ${host}> です。

sftpg3-mode

この設定では、ファイルを転送する際の **sftpg3** クライアントの動作を定義します。 sftpg3-mode エレメントには compatibility-mode 属性があり、値には以下を指定します。

- tectia (デフォルト) - **sftpg3** は再帰的にファイルを転送します。これは、カレント・ディレクトリとそのすべてのサブディレクトリからファイルが転送されることを意味します。
- ftp - **get/put** コマンドは **sget/sput** として実行されます。これは、1 つのファイルを転送することを意味します。 **mget/mput** コマンドは再帰の深さが 1 に設定されています。これは、指定されたディレクトリからのファイルのみが転送され、サブディレクトリからのファイルは転送されないことを意味します。
- openssh - **get/put/mget/mput** コマンドは同じように動作し、再帰の深さは 1 に設定されています。これは、指定されたディレクトリのファイルのみが転送され、サブディレクトリのファイルは転送されないことを意味します。

再帰の深さは、 **sftpg3** クライアントのコマンド **get/put/mget/mput** でコマンドライン・オプション `--max-depth="LEVEL"` を指定して上書きできます。詳細については、sftpg3(1) を参照してください。

terminal-selection

このエレメントでは、ユーザがダブルクリックでテキストを選択したときの Tectia ターミナルの動作を定義します。このエレメントの属性は`selection-type` だけで、以下の値を指定できます。

`select-words` - ダブルクリックで 1 単語ずつ選択され、スペースとすべての句読文字が区切り文字として使用されます。これがデフォルトです。

`select-paths` - スペースに挟まれた文字列が選択されます。つまり、選択範囲が `\/.-_` の文字まで広がります。たとえば、ファイルのパス内の任意の場所をダブルクリックすると、そのパス全体が選択されます。

terminal-bell

このエレメントでは、Tectia ターミナルが接続先サーバからの音声通知を繰り返すかどうかを定義します。このオプションは Unix サーバとの接続にのみ適用されます。このエレメントの属性は `bell-style` だけで、以下の値を指定できます。

`none` - 音声通知は使用されません。

`pc-speaker` - 接続先サーバから音声通知があった場合、ユーザの PC スピーカからビープ音を鳴らします。

`system-default` - Tectia ターミナルは、接続先サーバのシステムで定義されているデフォルトのアラートを鳴らします。これがデフォルトです。

close-window-on-disconnect

このエレメントでは、 **CTRL+D** キーを押してサーバ・セッションから切断するときに、Tectia ターミナルのウィンドウも閉じるように定義します。このエレメントの属性は `enable` だけで、 `yes` または `no` の値を指定できます。デフォルトは `no` です。これは、**CTRL+D** キーを押すとサーバ接続だけが閉じられ、Tectia ターミナルのウィンドウは開いたままであることを意味します。

quiet-mode

この設定では、コマンドライン・クライアントが警告、エラー・メッセージ、及び認証成功メッセージを抑制するかどうかを定義します。 `quiet-mode` エレメントには `enable` 属性があり、値には `yes` または `no` を指定します。デフォルトは `no` です。これは、エラー及びメッセージが出力され、ログに記録されることを意味します。

`quiet-mode` エレメントはコマンドライン・ツール **scpg3**、**sshg3**、及び **sftpg3** に影響します。ここで `quiet-mode enable="yes"` と設定してサイレント・モードを有効にすることは、これらのクライアントを `-q` オプションで実行するのと同じです。 `-q` コマンドラインのパラメータは、この設定ファイルで設定されている `quiet-mode` エレメントよりも優先されることに注意してください。

checksum

`checksum` エレメントを使用すると、チェックサムの比較に関するデフォルトの設定を定義できます。このデフォルトは、32kB 未満のファイルではチェックサムをチェックしないという工場出荷時の設定を上書きします。

`checksum` エレメントには `type` 属性があり、以下の値を指定できます。

`yes|YES` - 32kB を超えるファイルで、MD5 チェックサムがチェックされます。これがデフォルト値です。

`no|NO` - チェックサムは使用されません。

`md5|MD5` - 32kB を超えるファイルで、MD5 チェックサムだけがチェックされます。コマンドライン・クライアント **scpg3** 及び **sftpg3** で `--fips` パラメータが設定されている場合、このハッシュは使用されません。

`sha1|SHA1` - 32kB を超えるファイルで、SHA1 チェックサムだけがチェックされます。コマンドライン・クライアント **scpg3** 及び **sftpg3** で `--fips` パラメータが設定されている場合、このハッシュが使用されます。

`md5-force|MD5-FORCE` - コマンドライン・ツール **scpg3** 及び **sftpg3** で `--fips` パラメータが設定されている場合を除き、MD5 チェックサムが強制されます。

`sha1-force|SHA1-FORCE` - すべてのファイルで SHA1 チェックサムが強制されます。

`checkpoint|CHECKPOINT` - 1 つずつ転送される大きなファイルでチェックポイントが強制されます。

注意

接続ブローカーが FIPS モードで起動され、 `md5` 属性が設定ファイルで定義されていても、 **scpg3** または **sftpg3** が `--fips` パラメータで起動されていない場合には、 `md5` が使用されます。

チェックサムは、コマンドライン・クライアント **scpg3** 及び **sftpg3** でも、環境変数でも定義できます。3 つのチェックサムの設定の優先順位は以下の通りです (それぞれが異なる場合)。後の設定が常に前の設定を上書きします。

- 設定ファイルでの `checksum` 設定
- `SSH_SFTP_CHECKSUM_MODE` 環境変数
- コマンドライン引数

address-family

`address-family` エレメントでは IP アドレス・ファミリを定義します。 `type` 属性の値としてアドレス・ファミリを指定します。Tectia Client は IPv4 (`inet`) アドレス指定、IPv6 (`inet6`) アドレス指定、またはその両方 (`any`) を使用して動作します。 `type` のデフォルト値は `any` です。

profiles エレメント

profiles エレメントでは、指定されたサーバに接続するための接続プロファイルを定義します。 profiles エレメントには複数の profile エレメントを含めることができます。基本的には、プロファイルごとに 1 つのサーバへの接続ルールを定義します。 要求されるアルゴリズム等を指定する、一般的なプロファイルは、コマンド・ライン・オプションの --template-profile を使って **sshg3**、 **scpg3** または **sftpg3** でサーバに接続するために使用できます。profile エレメントでの設定はデフォルトの接続設定を上書きします。

接続にプロファイルが使用されている場合、プロファイルの設定はデフォルトの設定を上書きします。「default-settings エレメント」を参照してください。

profile

profile エレメントでは接続プロファイルを定義します。このエレメントには id、 name、 host、 port、 protocol、 host-type、 connect-on-startup、 user、 及び gateway-profile 属性があります。

プロファイルの id は、プロファイルの存続期間を通して変更されない一意の識別子でなければなりません。

プロファイルには、追加の name を指定できます。これは自由形式のテキスト文字列です。この名前は、コマンドラインでプロファイルを使って接続するために使用できるので、各プロファイルには一意の名前を定義してください。

host 属性は必須の設定で、Secure Shell サーバ・ホストのアドレスを定義します。アドレスには IP アドレスまたはドメイン名を指定できます。 host="*" の値を使用すると、セッションを開始するときにホスト・アドレスの入力をユーザに求めることができます。

port は必須の設定です。この属性では Secure Shell サーバ・リスナのポート番号を定義します。デフォルトのポートは 22 です。

protocol は必須の設定です。この属性では、使用する通信プロトコルを定義します。現在、許可されている値は secsh2 だけです。

接続ブローカーの起動時に、プロファイルで指定されている接続を自動的に行うには、connect-on-startup 属性の値を yes に設定します。この場合、 user 属性 (接続する相手のユーザ名) も指定します。また、接続のために何らかの形態の非対話型認証を設定する必要があります。

host-type 属性ではアスキー (テキスト) ファイル転送のためのサーバの種類を設定します。この属性では、アスキー・ファイルで使用される改行規則を指定します。デフォルト値は default です。これは、ローカル・プラットフォームによって改行規則が決定されることを意味します。クライアントが Windows 上で動作している場合、Windows 互換の改行 (CR + LF、'\r\n') が使用されます。クライアントが他のプラットフォームで動作している場合、Unix 互換の改行 (LF、'\n') が使用されます。 host-type に指定できるその他の値には、 windows (Windows リモート・ホスト用) と unix (Unix リモート・ホスト用) があります。この値は、Tectia Server 以外のサーバを使用している場合に定義してください。

`user` 属性では、接続を開くためのユーザ名を指定します。"`%USERNAME%`" の値を使用すると、現在ログインしているユーザのユーザ名を適用できます。 `user="*"` の値を使用すると、ログイン時にユーザ名を入力するようにユーザに求めることができます。 `user` 属性が定義されていない場合、デフォルトの接続設定で定義されているユーザ名が使用されます。

`gateway-profile` 属性を使用すると、ネストされたトンネルを作成できます。ネストできるトンネルは、プロファイルの `gateway-profile` を参照する `local-tunnel` エレメントで定義されているトンネルと、 `filter-engine` 及び `static-tunnels` で定義されているトンネルです。接続を通過させるプロファイル名を属性の値として指定します。最初のトンネルはゲートウェイ・ホスト・プロファイルを使用して作成され、そこから、このプロファイルで定義されているホストへの第 2 のトンネルが作成されます。

hostkey

このエレメントでは、 `file` 属性の値としてリモート・サーバ・ホストの公開鍵ファイルのパスを指定します。

代わりに、公開鍵を base64 エンコードされたアスキー・ブロックとして含めることもできます。

ciphers

このエレメントでは、このプロファイルで使用される暗号を定義します。詳細については、 **ciphers** を参照してください。

macs

このエレメントでは、このプロファイルで使用される MAC を定義します。詳細については、 **macs** を参照してください。

kexs

このエレメントでは、このプロファイルで使用される鍵交換を定義します。詳細については、 **kexs** を参照してください。

hostkey-algorithms

このエレメントでは、このプロファイルで使用されるホスト鍵署名アルゴリズムを定義します。詳細については、 **hostkey-algorithms** を参照してください。

rekey

このエレメントでは、このプロファイルで使用される鍵更新の設定を定義します。詳細については、 **rekey** を参照してください。

authentication-methods

このエレメントでは、このプロファイルで使用される認証方法を定義します。詳細については、 **authentication-methods** を参照してください。

user-identities

このエレメントでは、ユーザ公開鍵認証で使用される ID を指定します。接続ブローカーで使用できるすべての鍵を指定する `key-stores` エレメントとは対照的に、このエレメントを使用すると、この接続プロファイルが使用されるときに認証で試行される鍵を制御し、試行される順序を指定できます。

`user-identities` エレメントには複数の `identity` エレメントを含めることができます。複数の `identity` エレメントを使用すると、記述されている順に試行されます。

identity

`identity` エレメントでは `file`、 `hash`、 `identity-file`、 `id`、 及び `data` 属性を指定します。

`file` 属性では、公開鍵ファイル (主として) または証明書のパスを指定します。フル・パスとファイル名を値として入力します。

`hash` 属性は、関連する秘密鍵を識別するために使用される公開鍵のハッシュを入力するために使用します。この鍵は接続ブローカーで使用できなければなりません。使用できる鍵の公開鍵ハッシュは **ssh-broker-ctl** ツールで確認できます。ssh-broker-ctl(1) も参照してください。

`identity-file` 属性は将来の使用のために予約されています。

`id` 属性は将来の使用のために予約されています。

`data` 属性は将来の使用のために予約されています。

以下に `user-identities` エレメントの例を示します。

```
<user-identities>
  <identity file="$HOME/user/.ssh2/id_rsa_3072_a" />
  <identity file="C:\%username-without-domain%\private_keys\id_rsa_4096_a" />
  <identity hash="#a8edd3845005931aaa658b5573609e7d31e23afd" />
</user-identities>
```

compression

このエレメントでは、このプロファイルで使用される圧縮の設定を定義します。詳細については、 **compression** を参照してください。

proxy

このエレメントでは、このプロファイルで使用される HTTP プロキシ 及び SOCKS サーバの設定を定義します。詳細については、 **proxy** を参照してください。

このプロファイルに `gateway-profile` が定義されている場合、プロキシ設定は無視され、デフォルトのプロキシ設定またはゲートウェイ・プロファイルのプロキシ設定が代わりに使用されます。

idle-timeout

このエレメントでは、このプロファイルで使用されるアイドル・タイムアウトの設定を定義します。詳細については、 **idle-timeout** を参照してください。

tcp-connect-timeout

このエレメントでは、このプロファイルの TCP 接続タイムアウトを定義します。このタイムアウトは、ダウンしている、または到達できない Secure Shell サーバへの接続試行を終了するために使用されます。デフォルト値は 5 秒です。詳細については、 **tcp-connect-timeout** を参照してください。

keepalive-interval

このエレメントでは、Secure Shell サーバにキープアライブ・メッセージを送信する間隔を定義します。この設定はこのプロファイルに適用されます。デフォルト値は 0 です。これは、キープアライブ・メッセージが送信されないことを意味します。詳細については、 **keepalive-interval** を参照してください。

exclusive-connection

このエレメントでは、このプロファイルで接続が行われたときに、新しいチャネルごとに新しい接続を開くかどうかを定義します。この設定には 1 つの属性 `enable` があり、値には `yes` または `no` を指定します。デフォルトは `no` です。これは、クライアントから要求された新しいチャネルに対して、開いている接続が再利用されることを意味します。 **exclusive-connection** も参照してください。

server-banners

このエレメントでは、このプロファイルで使用されるサーバ・バナーの設定を定義します。詳細については、 **server-banners** を参照してください。

forwards

このエレメントでは、このプロファイルで許可される転送を定義します。詳細については、 **forwards** を参照してください。

tunnels

`tunnels` エレメントでは、このプロファイルによる接続が行われたときに開かれるトンネルを定義します。このエレメントには複数の `local-tunnel` 及び `remote-tunnel` エレメントを含めることができます。

local-tunnel

このエレメントでは、接続プロファイルを使用して接続されたときに自動的に開かれるローカル・トンネル (ポート転送) を定義します。このエレメントには `type`、 `listen-port`、 `listen-address`、 `dst-host`、 `dst-port`、 及び `allow-relay` の 6 つの属性があります。

`type` 属性ではトンネルの種類を定義します。この属性には `tcp` (デフォルト、特別な処理は行わない)、 `ftp` (FTP データ・チャネル用に一時的な転送を作成し、FTP セッション全体を効果的に保護する)、または `socks` (Tectia Client は他のアプリ

ケーションの SOCKS サーバとして動作し、SOCKS トランザクションの要求に応じて転送が作成される) を指定できます。

`listen-port` 属性ではローカル・クライアントのリスナ・ポート番号を定義します。

`listen-address` 属性を使用すると、クライアント上のどのネットワーク・インターフェイスでリッスンするのかを定義できます。この値には、ローカル・ホスト上のインターフェイスに属する IP アドレスを指定できます。値を `0.0.0.0` にすると、すべてのインターフェイスでリッスンします。デフォルトは `127.0.0.1` (クライアントの localhost ループバック・アドレス) です。その他の値を設定する場合は、`allow-relay="yes"` を設定する必要があります。

`address-family` のオプション `inet6` では、デフォルトのリッスン・アドレスは `::1` です。すべてのインターフェイスでリッスンするには、`::` を指定します。`address-family` のオプション `any` では、デフォルトではリッスン・アドレスは `127.0.0.1` と `::1` の両方です。すべてのインターフェイスでリッスンするには、`::` を指定します。

指定されたリスナへの接続が行われるたびに、接続は Secure Shell を介してリモート・サーバにトンネルされ、指定された接続先ホスト及びポート (`dst-host`、`dst-port`) にサーバからもう一つの接続が行われます。サーバからのその先の接続はセキュアではなく、通常の TCP 接続になります。

`dst-host` 及び `dst-port` 属性では接続先ホストのアドレスとポートを定義します。`dst-host` の値には IP アドレスまたはドメイン名を指定できます。デフォルトは `127.0.0.1` (localhost = サーバ・ホスト) です。

`allow-relay` 属性では、クライアント・ホストの外部から、リッスンされるポートへの接続を許可するかどうかを定義します。デフォルトは `no` です。`allow-relay="yes"` を使用すると、`listen-address` の設定もチェックします。

ローカル・トンネルの使用の詳細については、6.1 を参照してください。

remote-tunnel

このエレメントでは、接続プロファイルを使用して接続されたときに自動的に開かれるリモート・トンネル (ポート転送) を定義します。このエレメントには`type`、`listen-port`、`listen-address`、`dst-host`、`dst-port`、及び `allow-relay` の 6 つの属性があります。

`type` 属性ではトンネルの種類を定義します。この属性には `tcp` (デフォルト、特別な処理は行わない) または `ftp` (FTP データ・チャネル用に一時的な転送を作成し、クライアントとサーバ間の FTP セッションを効果的に保護する) を指定できます。

`listen-port` 属性ではリモート・サーバのリスナ・ポート番号を定義します。

`listen-address` 属性を使用すると、サーバ上のどのネットワーク・インターフェイスでリッスンするのかを定義できます。この値には、サーバ・ホスト上のイン

ターフェイスに属する IP アドレスを指定できます。値を `0.0.0.0` にすると、すべてのインターフェイスでリッスンします。デフォルトは `127.0.0.1` (サーバの localhost ループバック・アドレス) です。その他の値を設定する場合は、`allow-relay="yes"` とする必要があります。

`address-family` のオプション `inet6` では、デフォルトのリッスン・アドレスは `::1` です。すべてのインターフェイスでリッスンするには、`::` を指定します。`address-family` のオプション `any` では、デフォルトではリッスン・アドレスは `127.0.0.1` と `::1` の両方です。すべてのインターフェイスでリッスンするには、`::` を指定します。

このリスナへの接続が行われるたびに、接続は Secure Shell を介してローカル・クライアントにトンネルされ、指定された接続先ホスト及びポート (`dst-host`、`dst-port`) にクライアントからもう一つの接続が行われます。クライアントからのその先の接続はセキュアではなく、通常の TCP 接続になります。

`dst-host` 及び `dst-port` 属性では接続先ホストのアドレスとポートを定義します。`dst-host` の値には IP アドレスまたはドメイン名を指定できます。デフォルトは `127.0.0.1` (localhost = クライアント・ホスト) です。

`allow-relay` 属性では、サーバ・ホストの外部から、リスナ・ポートへの接続を許可するかどうかを定義します。デフォルトは `no` です。

リモート・トンネルの使用の詳細については、6.2 を参照してください。

extended

このエレメントは将来の使用のために予約されています。

remote-environment

このエレメントでは、このプロファイルで使用されるリモート環境の設定を定義します。`remote-environment` エレメント内で、サーバに渡す環境変数ごとに `environment` エレメントを定義します。詳細については、**remote-environment** を参照してください。

server-authentication-methods

このエレメントでは、このプロファイルで許可されるサーバ認証方法を定義します。詳細については、**server-authentication-methods** を参照してください。

password

このエレメントを使用すると、パスワード認証の応答としてクライアントが送信するユーザ・パスワードを指定できます。

パスワードは、`string` 属性で直接指定するか、`file` 属性でパスワードを含むファイルのパスを指定するか、または `command` 属性でパスワードを出力するプログラムまたはスクリプトのパスを指定できます。

`command` オプションを使用してシェル・スクリプトを参照する場合は、そのスクリプトがユーザのシェルを定義し、実際のパスワードを出力することも確認してください。そのようになっていない場合、シェル・スクリプトに使用するシェルを特定できないため、実行されたプログラムは失敗します。たとえば、パスワードの文字列が `my_password.txt` という名前のファイルに定義されていて、bash シェルを使用する場合は、以下の行をスクリプトに含めます。

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

警告

このオプションを使用してパスワードを指定する場合は、意図するユーザ以外は `ssh-broker-config.xml` ファイル、パスワード・ファイル、またはプログラムにアクセスできないようにすることが非常に重要です。

注意

コマンドライン・オプションで指定したパスワードがある場合、この設定を上書きします。

以下に接続プロファイルの例を示します。

```
<profile name="rock"
         id="id1"
         host="rock.example.com"
         port="22"
         connect-on-startup="no"
         user="doct">

  <hostkey file="key_22_rock.pub">
  </hostkey>

  <authentication-methods>
    <auth-publickey />
    <auth-password />
  </authentication-methods>

  <server-authentication-methods>
    <auth-server-publickey policy="strict" />
  </server-authentication-methods>

  <server-banners visible="yes" />

  <forwards>
    <forward type="agent" state="on" />
    <forward type="x11" state="on" />
  </forwards>

  <tunnels>
    <local-tunnel type="tcp"
                  listen-port="143"
```

```
                dst-host="imap.example.com"
                dst-port="143"
                allow-relay="no" />
  </tunnels>

  <remote-environment>
    <environment name="FOO" value="bar" />
    <environment name="QUX" value="%Ubaz" format="yes" />
    <environment name="ZAPPA" value="%Ubaz" />
  </remote-environment>

</profile>
```

static-tunnels エレメント

static-tunnels の設定は、自動トンネルの動作の設定に使用します。接続ブローカーの起動時にローカル・トンネルのリスナを自動的に作成できます。実際のトンネルは、リスナ・ポートに初めて接続されたときに形成されます。その時点でサーバへの接続が開かれていない場合は、その接続も自動的に開かれます。

static-tunnels エレメントには任意の数の tunnel エレメントを含めることができます。

tunnel

tunnel エレメントでは静的トンネルを指定します。このエレメントには以下の属性があります: type、 listen-port、 listen-address、 dst-host、 dst-port、 allow-relay、及び profile

type 属性ではトンネルの種類を定義します。トンネルの種類には tcp と ftp があります。

- tcp は一般的な TCP トンネリング用のリスナを指定します。
- ftp はFTP トンネリング用のリスナを指定します (FTP データ・チャネルもトンネリングされます)。

listen-port 属性では、ローカル・クライアントのリスナ・ポート番号を定義します。

listen-address 属性を使用すると、クライアント上のどのネットワーク・インターフェイスでリッスンするのかを定義できます。この値には、ローカル・ホスト上のインターフェイスに属する IP アドレスを指定できます。値を 0.0.0.0 にすると、すべてのインターフェイスでリッスンします。デフォルトは 127.0.0.1 (クライアントの localhost ループバック・アドレス) です。その他の値を設定する場合は、 allow-relay="yes" とする必要があります。

address-family のオプション inet6 では、デフォルトのリッスン・アドレスは ::1 です。すべてのインターフェイスでリッスンするには、 :: を指定します。 address-family のオプション any では、デフォルトではリッスン・アドレスは 127.0.0.1 と ::1 の両方です。すべてのインターフェイスでリッスンするには、 :: を指定します。

dst-host 及び dst-port 属性では接続先ホストのアドレスとポートを定義します。 dst-host の値には IP アドレスまたはドメイン名を指定できます。デフォルトは 127.0.0.1 (localhost = サーバ・ホスト) です。

allow-relay 属性では、クライアント・ホストの外部から、リッスンされるポートへの接続を許可するかどうかを定義します。デフォルトは no です。

profile 属性では、トンネルに使用される接続プロファイルの ID を指定します。

```
<static-tunnels>
  <tunnel type="tcp"
          listen-address="127.0.0.1"
          listen-port="9000"
          dst-host="st.example.com"
          dst-port="9000"
          allow-relay="no"
          profile="id1" />
</static-tunnels>
```

gui エレメント

gui エレメントは Tectia ターミナルの GUI 設定の調整に使用します。 gui エレメントにはhide-tray-icon、 show-exit-button、及び show-admin 属性があります。これらはすべて、値として yes または no を指定する必要があります。

hide-tray-icon 属性では、Windows タスクバー (システム・トレイとも呼ばれる) の通知領域に Tectia アイコンを表示させるかどうかを制御します。デフォルトは no です (トレイ・アイコンが表示されます)。

show-exit-button 属性では、Tectia アイコンのショートカット・メニューに **終了** コマンドを表示させるかどうかを制御します。デフォルトは yes です。

show-admin 属性では、Tectia アイコンのショートカット・メニューに **設定** コマンドを表示させるかどうかを定義します。デフォルトは yes です。ボタンが表示されない場合に Tectia コネクション設定 GUI を起動させるには、 ssh-tectia-configuration.exe (デフォルトで "<INSTALLDIR>\SSH Tectia Broker" ディレクトリにあります) を実行します。

```
<gui hide-tray-icon="no"
     show-exit-button="yes"
     show-admin="yes"
```

logging エレメント

logging エレメントでは、ログ・イベントの重大性とログのファシリティを定義するログ設定を変更します。このエレメントは 1 つ以上の log-target 及び log-events エレメントを含みます。

log-target

このエレメントでは、監査のためのログ・ターゲットを指定します。デフォルトでは、ブローカは何もログに記録しません。このエレメントを使用すると、ログ・データをファイルまたは syslog に移すことができます。

log-target エレメントには file 及び type を属性として指定できます。

`type` 属性では、監査データが出力されるログ機能を指定します。指定できる値は `file`、`syslog`、または `discard` です。

`file` 属性では、監査データの書き込み先であるファイル・システムのパスを設定します。`type` 属性に `syslog` または `discard` が設定されている場合、`file` 属性は使用できません。

log-events

このエレメントでは、それぞれのログ・イベントの重大性とファシリティを設定します。イベントには適切なデフォルト値があり、明示的なログ設定が行われない場合に使用されます。この設定により、デフォルト値をカスタマイズできます。

このエレメントには、1 つまたは複数の `log-target` エレメントを含めることもできます。ここで定義すると、`log-target` エレメントは `logging/log-target` で指定されている定義を上書きします。

イベントには、属性として `facility` と `severity` を設定できます。イベント自体は `log-events` エレメント内に記述されている必要があります。

ファシリティには `normal`、`daemon`、`user`、`auth`、`local0`、`local1`、`local2`、`local3`、`local4`、`local5`、`local6`、`local7`、または `discard` を指定できます。ファシリティを `discard` に設定すると、サーバは指定されたログ・イベントを無視するようになります。

Windows では、`normal` と `discard` のファシリティのみが使用されます。

重大性には `informational`、`notice`、`warning`、`error`、`critical`、`security-success`、または `security-failure` を指定できます。

設定ファイルで特に定義されていないイベントには、デフォルト値が使用されます。その他すべての定義の後に空の `log-events` エレメントを指定し、そのエレメントに重大性の値を設定することで、残りのすべてのイベントについてデフォルトを上書きできます。

ログ・イベントの名前には、'*' と '?' の文字をワイルドカードとして使用できます。

ログ・イベントの完全なリストについては、付録E を参照してください。

デフォルトでログを記録するようにプログラムされているすべてのイベントのログを、`/tmp/foo` と `syslog` の両方に記録する場合のログ設定例を以下に示します。

```
<logging>
  <log-target file="/tmp/foo" />
  <log-target type="syslog" />
</logging>
```

イベント名が "`Key_store_*`" に一致するイベントを除き (それらは破棄されます)、イベントのログを `/tmp/foo` に記録する場合のログ設定例を以下に示します。

```
<logging>
  <log-target file="/tmp/foo" />
  <log-events facility="discard">
    Key_store_*
```

```
  </log-events>
</logging>
```

A.3. 設定ファイルのバックアップ

アップグレードやテスト設定の作成を始める前に、変更を加えた接続ブローカーの設定ファイルのバックアップがあることを確認してください。

ユーザ固有の接続ブローカーの設定ファイルは、デフォルトでは Unix の場合は `$HOME/.ssh2/ssh-broker-config.xml` に、Windows の場合は `%APPDATA%\SSH\ssh-broker-config.xml` にあります。接続ブローカーの設定ファイルを保存するたびに、古い設定ファイルのバックアップ (`ssh-broker-config.xml.bak`) が同じディレクトリに保存されます。バックアップしたファイルに戻す必要がある場合は、元の場所に元の名前でコピーしてください。

Windows で Tectia をアップグレードすると、以前の接続ブローカーの設定ファイルのバックアップ・コピーが自動的に作成され、以下のユーザ固有のディレクトリに保存されます。

"`%APPDATA%\SSH\backup-<version>-<date>`"

ここで

`<version>` は Tectia のリリース・バージョン

`<date>` はアップグレードの日付です。

A.4. 接続ブローカー設定ファイルのクイック・リファレンス

この付録では、接続ブローカーの設定ファイル (`ssh-broker-config.xml`) のエレメントに関するクイック・リファレンスをまとめています。 クイック・リファレンスは以下の 4 つの表に分かれています。

- 表 A.3: `general` エレメント
- 表 A.4: `default-settings` エレメント
- 表 A.5: `profiles` エレメント
- 表 A.6: その他のエレメント (`static-tunnels`、`gui`、 及び `logging`)

それぞれの表には、使用できる設定ファイルのエレメントとその属性、属性値 (デフォルト値がある場合は太字で表示)、及び説明が記載されています。表中のエレメント名には、ssh-broker-config(5) に記載されているエレメントの詳細説明へのリンクが設定されています。

エレメントの階層は、親エレメントと子エレメントの間にスラッシュ ('/') を入れて表されています。

表A.3 ssh-broker-config.xml クイック・リファレンス - general エレメント

エレメント	属性とその値	説明
crypto-lib	mode = "standard|fips"	暗号ライブラリ・モード: 標準または FIPS 140-2 認証
cert-validation	end-point-identity-check = "yes|no|ask"	クライアントはサーバのホスト名または IP アドレスをサーバのホスト証明書に照らして検証します
	default-domain = domain_name	リモート・システム名のデフォルトのドメイン部分
	http-proxy-url = HTTP_proxy	証明書の有効性を問い合わせるための HTTP プロキシ
	socks-server-url = SOCKS_server	証明書の有効性を問い合わせるための SOCKS サーバ
	cache-size = [1 to 512] (デフォルト: "300")	証明書及び CRL のためのメモリ・キャッシュの最大サイズ (MB)
	max-crl-size = [1 to 512] (デフォルト: "50")	受け入れる CRL の最大サイズ (MB)
	external-search-timeout = [1 to 3600] (デフォルト: "60")	CRL や証明書を外部 HTTP 及び LDAP から検索する時間制限 (秒)
	max-ldap-response-length = [1 to 512] (デフォルト: "50")	受け入れる LDAP 応答の最大サイズ (MB)
	ldap-idle-timeout = [1 to 3600] (デフォルト: "30")	LDAP 接続のアイドル・タイムアウト (秒)
	max-path-length = number (デフォルト: "10")	証明書を検証するときの証明書パスの最大長
cert-validation / ldap-server	address = LDAP_server_address	CRL 及び/または下位の CA 証明書を取得するための LDAP サーバ・アドレス
	port = port_number (デフォルト: "389")	CRL 及び/または下位の CA 証明書を取得するための LDAP サーバ・ポート
cert-validation / ocsp-responder	url = URL_address	OCSP (オンライン証明書状態プロトコル) レスポンダ・サービスのアドレス
	validity-period = seconds (デフォルト: "0")	同じ証明書に対する新たな OCSP 問い合わせが行われない (古い結果が使用される) 期間

エレメント	属性とその値	説明
cert-validation / crl-prefetch	url = LDAP_URL|HTTP_URL|file_URL	Tectia Client は定期的にこの URL から CRL をダウンロードします
	interval = seconds (デフォルト: "3600")	CRL がダウンロードされる間隔
cert-validation / dod-pki	enable = "yes|no"	鍵の用途に電子署名を強制します
cert-validation / ca-certificate	name = CA_name	サーバ認証に使用する認証局 (CA) の名称
	file = path	X.509 CA 証明書ファイルのパス
	disable-crls = "yes|no"	CRL チェックを無効にします
	use-expired-crls = seconds (デフォルト: "0")	失効した CRL を使用する期間
key-stores / key-store	type = "mscapi|pkcs11|software|zos-saf"	鍵ストアの種類
	init = init_info\	鍵ストア・プロバイダ固有の初期化情報
key-stores / user-keys	directory = path	ユーザの秘密鍵が保存されているディレクトリ
	passphrase-timeout = seconds (デフォルト: "0")	パスフレーズで保護された秘密鍵がタイムアウトするまでの時間
	passphrase-idle-timeout = seconds (デフォルト: "0")	パスフレーズで保護された秘密鍵が、ユーザがその鍵にアクセスしたり、使用したりしないことでタイムアウトするまでの時間
key-stores / identification	file = path	ユーザ鍵を定義する identification ファイルの場所
	base-path = path	identification ファイルがユーザの秘密鍵の保存場所であると期待するディレクトリ
	passphrase-timeout = seconds (デフォルト: "0")	ユーザによるパスフレーズの再入力が必要になるまでの時間
	passphrase-idle-timeout = seconds (デフォルト: "0")	ユーザによる操作がないためにパスフレーズがタイムアウトするまでの時間
user-config-directory	path = path (デフォルト: "%USER_CONFIG_DIRECTORY%")	ユーザ固有設定ファイルの非デフォルトの場所
file-access-control (Unix のみ)	enable = "yes|no"	グローバル設定ファイル、ユーザ固有の設定ファイル、及び秘密鍵ファイルに定義されている

エレメント	属性とその値	説明
		ファイル・アクセス・パーミッションのチェックを有効にします
protocol-parameters	threads = number (0 に設定すると、デフォルト値が使用されます)	プロトコル・ライブラリが使用するスレッド (ファスト・パス・ディスパッチャ・スレッド) の数
known-hosts	path = path	known-hosts ファイルまたはディレクトリの非デフォルトの場所
	file = path	OpenSSH 形式の known_hosts ファイルの場所
	directory = path	既知のホスト鍵を保存するための非デフォルトのディレクトリ
	filename-format = "hash|plain|default" ("default" = "hash")	新しいホスト鍵ファイルを保存するフォーマット

表A.4 ssh-broker-config.xml クイック・リファレンス - default-settings エレメント

エレメント	属性とその値	説明
	user = user_name	リモート・サーバに接続するときに使用するデフォルトのユーザ名
ciphers / cipher	name = cipher_name	クライアントがデータ暗号化のために要求する暗号
macs / mac	name = MAC_name	クライアントがデータ完全性の検証のために要求する MAC
kexs / kex	name = KEX_name	クライアントが鍵交換のために要求する鍵交換方法
hostkey-algorithms / hostkey-algorithm	name = hostkey-algorithm_name	ホスト鍵または証明書によるサーバ認証に使用されるホスト鍵署名アルゴリズム
rekey	bytes = number (デフォルト: "1000000000" (1 GB))	鍵交換が再度行われるまでに転送されるバイト数
authentication-methods / auth-hostbased	-	ホストベースの認証が使用されます
authentication-methods / auth-hostbased / local-hostname	name = host_name	ホストベースの認証時にリモート・サーバに通知されるローカル・ホスト名

エレメント	属性とその値	説明
authentication-methods / auth-password	-	パスワード認証が使用されます
authentication-methods / auth-publickey	-	公開鍵認証が使用されます
	signature-algorithms = comma-separated_list	クライアント認証に使用される公開鍵署名アルゴリズム
authentication-methods / auth-publickey / key-selection	policy = "automatic|interactive-shy"	ユーザ公開鍵をサーバに提示する際にクライアントが使用する鍵の選択ポリシー
authentication-methods / auth-publickey / key-selection / public-key	type = "plain|certificate" (デフォルトでは両方が試行されます)	公開鍵認証の際に、プレーンな公開鍵のみ、または証明書のみが試されます
authentication-methods / auth-publickey / key-selection / issuer-name	name = certificate_issuer_name	この名前と、サーバが要求した、または受け入れた証明書発行者を比較することによって、クライアント側のユーザ証明書をフィルタリングできます
	match-server-certificate = "yes|no"	接続ブローカーはユーザ証明書の発行者名とサーバ証明書の発行者名のマッチングを試みます
authentication-methods / auth-gssapi	-	認証に GSSAPI が使用されます
	dll-path = path (Windows では無視されます)	必要な GSSAPI ライブラリの場所
	allow-ticket-forwarding = "yes|no"	複数の接続にまたがる Kerberos チケットの転送を許可します
authentication-methods / auth-keyboard-interactive	-	認証にキーボード・インタラクティブ方法が使用されます
hostbased-default-domain	name = domain_name	サーバに送信する前に短いホスト名に付加される、ホストのデフォルトのドメイン名
compression	name = "none|zlib"	クライアントが送信するデータを圧縮します
	level = [0 to 9] (デフォルト: "0" (= レベル 6))	zlib の場合の圧縮レベル
proxy	ruleset = rule_sequence	クライアントが接続に使用する HTTP プロキシまたは SOCKS サーバのルール

エレメント	属性とその値	説明
idle-timeout	type = "connection"	アイドル・タイムアウトは常に接続に対して定義されます
	time = seconds (デフォルト: "5")	接続を自動的に閉じるまでに許可されるアイドル時間 (すべての接続チャネルが閉じた後)
tcp-connect-timeout	time = seconds (デフォルト: "5")	TCP 接続のタイムアウト (リモート・ホストがダウンしている場合や到達できない場合に Secure Shell サーバへの接続試行が停止されるまでの時間)
keepalive-interval	time = seconds (デフォルト: "0")	Secure Shell サーバにキープアライブ・メッセージを送信する時間間隔
exclusive-connection	enable = "yes|no"	新しいチャネルごとに新しい接続が開かれます
server-banners	visible = "yes|no"	サーバ・バナー・メッセージ・ファイル (存在する場合) をログイン前のユーザに表示します
forwards / forward	type = "x11|agent"	転送の種類
	state = "on|off|denied"	転送のオン / オフを設定するか、または転送を拒否します
remote-environment / environment	name = env_var_name	クライアント側からサーバに渡される環境変数の名前
	value = string	環境変数の値
	format = "yes|no"	接続ブローカーは、 value で指定された Tectia 固有の特殊変数 (例: %U%) を処理します
server-authentication-methods / auth-server-certificate	-	サーバ認証に証明書を使用します
server-authentication-methods / auth-server-publickey	-	サーバ認証に公開ホスト鍵を使用します
	policy = "strict|ask|tofu|advisory"	未知のサーバ・ホスト鍵の取り扱いに関するポリシー
authentication-success-message	enable = "yes|no"	AuthenticationSuccessMsg メッセージを出力してログに記録します
	delay =seconds (default: "2")	期間認証成功のメッセージが表示される時間

エレメント	属性とその値	説明
disconnect-message	message = string	接続の切断時に表示されるメッセージを指定します。
keyboard-interactive	prefix = string	キーボード・インタラクティブのプロンプトの前に表示されるメッセージを指定します。
sftpg3-mode	compatibility-mode = "tectia|ftp|openssh"	ファイル転送時の **sftpg3** の動作
terminal-selection	selection-type = "select-words|select-paths"	ユーザがダブルクリックでテキストを選択したときの Tectia ターミナルの動作
terminal-bell	bell-style = "none |pc-speaker|system-default"	Tectia ターミナルが接続先サーバ (Unix) から音声通知を繰り返します
close-window-on-disconnect	enable = "yes|no"	**CTRL+D** キーを押してサーバ・セッションから切断するときに、Tectia ターミナルのウィンドウが閉じます:
quiet-mode	enable = "yes|no"	**scpg3**、 **sshg3**、及び **sftpg3** に警告、エラー・メッセージ、及び認証成功メッセージを抑制させます。
checksum	type = "yes|no|md5|sha1|md5-force| sha1-force|checkpoint"	チェックサムの比較のデフォルト設定
address-family	type = "any|inet|inet6"	IP アドレス・ファミリ: 両方、IPv4、または IPv6

表A.5 ssh-broker-config.xml クイック・リファレンス - profiles エレメント

エレメント	属性とその値	説明
profile	id = ID	プロファイルの存続期間を通して変更されない一意の識別子
	name = string	プロファイルで接続するためにコマンドラインで使用できる一意の名前 (自由形式のテキスト文字列)
	host =IP_address|FQDN |short_hostname	Secure Shell サーバのホスト・アドレス
	port = port_number (デフォルト: "22")	Secure Shell サーバのリスナ・ポート番号
	protocol = "secsh2"	プロファイルが使用する通信プロトコル
	host-type = "default|windows|unix"	アスキー (テキスト) ファイル転送のサーバ・タイプ

エレメント	属性とその値	説明
	connect-on-startup = "yes|no"	接続ブローカーの起動時にこのプロファイルで自動的に接続します
	user = user_name	接続を開くためのユーザ名
	gateway-profile = profile_name	ネストされたトンネルを作成します
profile / **hostkey**	file = path	リモート・サーバのホスト公開鍵ファイルのパス
profile / **ciphers** / **cipher**	name = cipher_name	このプロファイルで使用される暗号
profile / **macs** / **mac**	name = MAC_name	このプロファイルで使用されるMAC
profile / **kexs** / **kex**	name = KEX_name	このプロファイルで使用される鍵交換
profile / **hostkey-algorithms** / **hostkey-algorithm**	name = hostkey-algorithm_name	このプロファイルで使用されるホスト鍵署名アルゴリズム
profile / **rekey**	bytes = number (デフォルト: "1000000000" (1 GB))	このプロファイルを使用するときに、鍵交換が再度行われるまでに転送されるバイト数
profile / **authentication-methods**	default-settings / authentication-methods と同じ子エレメントを使用して、このプロファイルの認証方法を定義します (表 A.4 を参照)	
profile / **user-identities** / **identity**	file = path	(主として) 公開鍵ファイルまたは証明書のパス
	hash = hash	関連する秘密鍵を識別するために使用される公開鍵のハッシュ
	identity-file = path	将来の使用のために予約。
profile / **compression**	name = "none|zlib"	このプロファイルで使用される (クライアントが送信するデータの) 圧縮設定
	level = [0 to 9] (デフォルト: "0" (= レベル 6))	zlib の圧縮レベル
profile / **proxy**	ruleset = rule_sequence	このプロファイルを使用した接続にクライアントが使用するHTTP プロキシまたは SOCKS サーバのルール
profile / **idle-timeout**	type = "connection"	アイドル・タイムアウトは常に接続に対して定義されます
	time = seconds (デフォルト: "5")	このプロファイルを使用して開かれた接続を自動的に閉じるま

エレメント	属性とその値	説明
		でに許可されるアイドル時間 (すべての接続チャネルが閉じた後)
profile / **tcp-connect-timeout**	`time = seconds` (デフォルト: `"5"`)	このプロファイルを使用した TCP 接続のタイムアウト。リモート・ホストがダウンしている場合や到達できない場合、Secure Shell サーバへの接続試行が定義された時間後に停止されます
profile / **keepalive-interval**	`time = seconds` (デフォルト: `"0"`)	このプロファイルを使用して Secure Shell サーバにキープアライブ・メッセージを送信する時間間隔
profile / **exclusive-connection**	`enable = "yes|no"`	このプロファイルを使用して、新しいチャネルごとに新しい接続が開かれます
profile / **server-banners**	`visible = "yes|no"`	サーバ・バナー・メッセージ・ファイル (存在する場合) を、このプロファイルを使用してログインする前にユーザに表示します
profile / **forwards** / **forward**	`type = "x11|agent"`	このプロファイルの転送の種類
	`state = "on|off|denied"`	このプロファイルを使用して、転送をオン、オフ、または拒否 (ユーザがコマンドラインで転送を有効にできない状態) に設定します。
profile / **tunnels** / **local-tunnel**	`type = "tcp|ftp|socks"`	このプロファイルを使用して接続されたときに、自動的に開かれるローカル・トンネルの種類
	`listen-address = IP_address` (デフォルト: `127.0.0.1`)	クライアントでリッスンするネットワーク・インターフェイス
	`listen-port = port_number`	ローカル・クライアントのリスナ・ポート番号
	`dst-host = IP_address|domain_name` (デフォルト: `127.0.0.1`)	接続先ホストのアドレス
	`dst-port = port_number`	接続先ポート

エレメント	属性とその値	説明
	allow-relay = "yes|no"	クライアント・ホストの外部から、リッスンされるポートへの接続を許可します
profile / tunnels / remote-tunnel	type = "tcp|ftp"	このプロファイルを使用して接続されたときに、自動的に開かれるリモート・トンネルの種類
	listen-address = IP_address (デフォルト: 127.0.0.1)	サーバがリッスンするネットワーク・インターフェイス
	listen-port = port_number	リモート・サーバのリスナ・ポート番号
	dst-host = IP_address|domain_name (デフォルト: 127.0.0.1)	接続先ホストのアドレス
	dst-port = port_number	接続先ポート
	allow-relay = "yes|no"	サーバ・ホストの外部から、リスナ・ポートへの接続を許可します
profile / remote-environment / environment	name = env_var_name	クライアント側からサーバに渡される環境変数の名前
	value = string	環境変数の値
	format = "yes|no"	接続ブローカーは、 value で指定された Tectia 固有の特殊変数 (例: %U%) を処理します
profile / server-authentication-methods	default-settings / server-authentication-methods と同じ子エレメントを使用して、このプロファイルを使用して許可されるサーバ認証方法を定義します (表 A.4 を参照)	
profile / password	string = password	パスワード認証の応答としてクライアントが送信するユーザ・パスワード
	file = password_file	パスワードを含むファイル
	command = path	パスワードを出力するプログラムまたはスクリプトのパス

表A.6 ssh-broker-config.xml クイック・リファレンス - static-tunnels、gui、及び logging エレメント

エレメント	属性とその値	説明
static-tunnels / tunnel	type = "tcp|ftp"	静的トンネルの種類
	listen-address = IP_address (デフォルト: 127.0.0.1)	クライアントがリッスンするネットワーク・インターフェイス

エレメント	属性とその値	説明
	listen-port = port_number	ローカル・クライアントのリスナ・ポート番号
	dst-host = IP_address|domain_name (デフォルト: 127.0.0.1)	接続先ホストのアドレス
	dst-port = port_number	接続先ポート
	allow-relay = "yes|no"	クライアント・ホストの外部から、リッスンされるポートへの接続を許可します
	profile = ID	トンネルに使用される接続プロファイル ID
gui	hide-tray-icon = "yes|no"	Windows タスクバーの通知領域にある Tectia アイコンを非表示にします
	show-exit-button = "yes|no"	Tectia アイコンのショートカット・メニューに 終了 コマンドを表示します
	show-admin = "yes|no"	Tectia アイコンのショートカット・メニューに 設定 コマンドを表示します
logging / log-target	file = path	監査データが書き込まれるファイル
	type = "file|syslog|discard"	監査データの出力先となるログ機能
logging / log-events	facility = "normal|daemon |user|auth|local0|local1|local2 |local3|local4|local5|local6 |local7|discard" (Windows の場合: facility = "normal|discard")	イベントの種類
	severity = "informational |notice|warning|error| critical|security-success |security-failure"	イベントの重要度
logging / log-events / log-target	logging / log-target と同じ	

A.5. Broker 設定ファイルの構文

接続ブローカーの設定ファイルの DTD は以下の通りです。

```
<!-- secsh-broker.dtd                                              -->
<!--                                                               -->
<!-- Copyright (c) 2017 SSH Communications Security Corporation.   -->
<!-- This software is protected by international copyright laws.   -->
<!-- All rights reserved.                                          -->
<!--                                                               -->
<!-- Document type definition for the Connection Broker XML        -->
<!-- configuration files.                                          -->
<!--                                                               -->

<!-- Tunable parameters used in the policy. -->

<!-- Both ipv4 and ipv6 are enabled by default -->
<!ENTITY default-address-family-type        "any">


<!-- The top-level element -->
<!ELEMENT secsh-broker          (general?,default-settings?,profiles?,
                                 static-tunnels?,gui?,
                                 filter-engine?,logging?)>
<!ATTLIST secsh-broker
      version                   CDATA           #IMPLIED>

<!-- General element. Only "known-hosts" can appear multiple times. -->
<!ELEMENT general               (crypto-lib|cert-validation|key-stores|
                                 strict-host-key-checking|host-key-always-ask|
                                 accept-unknown-host-keys|known-hosts|
                                 user-config-directory|file-access-control|
                                 protocol-parameters)*>

<!-- Cryptographic library. -->
<!ELEMENT crypto-lib            EMPTY>
<!ATTLIST crypto-lib
      mode                      (fips|standard) "standard">

<!-- PKI settings. "dod-pki" element may appear only once, other elements -->
<!-- may be specified multiple times.                                    -->

<!ELEMENT cert-validation       (ldap-server|
                                 ocsp-responder|
                                 crl-prefetch|
                                 dod-pki|
                                 ca-certificate|
                                 key-store)*>

<!ATTLIST cert-validation
      end-point-identity-check  (yes|no|YES|NO|ask|ASK) "yes"
      default-domain            CDATA           #IMPLIED
      http-proxy-url            CDATA           #IMPLIED
      socks-server-url          CDATA           #IMPLIED
      max-path-length           CDATA           "10"
      cache-size                CDATA           "300"
      max-crl-size              CDATA           "50"
      external-search-timeout   CDATA           "60"
```

```
      max-ldap-response-length  CDATA           "50"
      ldap-idle-timeout         CDATA           "30">

<!ELEMENT ldap-server           EMPTY>
<!ATTLIST ldap-server
      address                   CDATA           #REQUIRED
      port                      CDATA           "389">

<!ELEMENT ocsp-responder        (#PCDATA)>
<!ATTLIST ocsp-responder
      url                       CDATA           #REQUIRED
      validity-period           CDATA           "0"
      responder-certificate     CDATA           #IMPLIED>

<!-- CRL prefetch. -->
<!ELEMENT crl-prefetch          EMPTY>
<!ATTLIST crl-prefetch
      interval                  CDATA           "3600"
      url                       CDATA           #REQUIRED>

<!-- CA certificates. -->
<!ELEMENT ca-certificate        (#PCDATA)>
<!ATTLIST ca-certificate
      name                      CDATA           #REQUIRED
      file                      CDATA           #IMPLIED
      disable-crls              (yes|no|YES|NO) "no"
      use-expired-crls          CDATA           "0" >

<!-- Enforce digital signature in key usage. -->
<!ELEMENT dod-pki               EMPTY>
<!ATTLIST dod-pki
      enable                    (yes|no|YES|NO) "no" >

<!ELEMENT key-stores            ((key-store|user-keys|identification)*)>

<!ELEMENT key-store             EMPTY>
<!ATTLIST key-store
      type                      CDATA           #REQUIRED
      init                      CDATA           #IMPLIED
      disable-crls              (yes|no|YES|NO) "no"
      use-expired-crls          CDATA           "0" >

<!ELEMENT user-keys             EMPTY>
<!ATTLIST user-keys
      directory                 CDATA           #IMPLIED
      poll-interval             CDATA           "10"
      passphrase-timeout        CDATA           "0"
      passphrase-idle-timeout   CDATA           "0">

<!ELEMENT identification        EMPTY>
<!ATTLIST identification
      file                      CDATA           #REQUIRED
      base-path                 CDATA           #IMPLIED
      passphrase-timeout        CDATA           "0"
```

```
      passphrase-idle-timeout   CDATA           "0">

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT strict-host-key-checking EMPTY>
<!ATTLIST strict-host-key-checking
      enable                    (yes|no|YES|NO) #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT host-key-always-ask   EMPTY>
<!ATTLIST host-key-always-ask
      enable                    (yes|no|YES|NO) #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT accept-unknown-host-keys EMPTY>
<!ATTLIST accept-unknown-host-keys
      enable                    (yes|no|YES|NO) #REQUIRED>

<!ELEMENT exclusive-connection  EMPTY>
<!ATTLIST exclusive-connection
      enable                    (yes|no|YES|NO) #REQUIRED>

<!ELEMENT known-hosts           (key-store*)>
<!ATTLIST known-hosts
      path                      CDATA           #IMPLIED
      file                      CDATA           #IMPLIED
      directory                 CDATA           #IMPLIED
      filename-format           (hash|plain|default) "default" >

<!-- Extended plugin configuration -->
<!ELEMENT extended              (ext)*>

<!ELEMENT ext                   (#PCDATA | EMPTY | ext)*>
<!ATTLIST ext
      name                      CDATA           #REQUIRED>

<!-- Default settings element.  No element may appear multiple times. -->
<!ELEMENT default-settings      (ciphers|macs|kexs|hostkey-algorithms|
                                 transport-distribution|rekey|
                                 authentication-methods|
                                 hostbased-default-domain|
                                 compression|proxy|idle-timeout|
                                 tcp-connect-timeout|keepalive-interval|
                                 exclusive-connection|server-banners|
                                 forwards|extended|remote-environment|
                                 server-authentication-methods|
                                 authentication-success-message|
                                 sftpg3-mode|terminal-selection|terminal-bell|
                                 close-window-on-disconnect|quiet-mode|
                                 checksum|address-family)*>

<!ATTLIST default-settings
      user                      CDATA           #IMPLIED>

<!-- Server banners. -->
```

```
<!ELEMENT server-banners        EMPTY>
<!ATTLIST server-banners
      visible                   (yes|no|YES|NO) "yes">

<!-- Ciphers element. -->
<!ELEMENT ciphers               (cipher*)>

<!-- Cipher. -->
<!ELEMENT cipher                EMPTY>
<!ATTLIST cipher
      name                      CDATA           #REQUIRED>

<!-- Macs element. -->
<!ELEMENT macs                  (mac*)>

<!-- Mac. -->
<!ELEMENT mac                   EMPTY>
<!ATTLIST mac
      name                      CDATA           #REQUIRED>

<!-- Kexs element. -->
<!ELEMENT kexs                  (kex*)>

<!-- Kex. -->
<!ELEMENT kex                   EMPTY>
<!ATTLIST kex
      name                      CDATA           #REQUIRED>

<!-- Hostkey algorithms element. -->
<!ELEMENT hostkey-algorithms    (hostkey-algorithm*)>

<!-- Hostkey algorithm. -->
<!ELEMENT hostkey-algorithm     EMPTY>
<!ATTLIST hostkey-algorithm
      name                      CDATA           #REQUIRED>

<!ELEMENT rekey                 EMPTY>
<!ATTLIST rekey
      bytes                     CDATA           "0">

<!-- Hostbased default domain. -->
<!ELEMENT hostbased-default-domain EMPTY>
<!ATTLIST hostbased-default-domain
      name                      CDATA           #REQUIRED>

<!-- Authentication methods element. -->
<!ELEMENT authentication-methods (authentication-method|auth-hostbased
                                |auth-password|auth-publickey|auth-gssapi
                                |auth-keyboard-interactive)*>
<!ELEMENT server-authentication-methods (authentication-method
                                |auth-server-publickey
                                |auth-server-certificate)*>

<!ELEMENT auth-server-publickey EMPTY>
```

```
<!ATTLIST auth-server-publickey
      policy                    CDATA #IMPLIED><!-- "strict", "ask", "tofu", -->
                                              <!-- "advisory" -->

<!ELEMENT auth-server-certificate   EMPTY>

<!ELEMENT remote-environment    (environment*)>

<!ELEMENT environment           EMPTY>
<!ATTLIST environment
      name                      CDATA           #REQUIRED
      value                     CDATA           #REQUIRED
      format                    (yes|no|YES|NO) "no">

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT transport-distribution EMPTY>
<!ATTLIST transport-distribution
      num-transports            CDATA           #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT authentication-method EMPTY>
<!ATTLIST authentication-method
      name                      CDATA           #REQUIRED>

<!ELEMENT auth-hostbased        (local-hostname?)>
<!ELEMENT local-hostname        EMPTY>
<!ATTLIST local-hostname
      name                      CDATA           #REQUIRED>

<!ELEMENT auth-password         EMPTY>

<!ELEMENT auth-publickey        (key-selection?)>
<!ATTLIST auth-publickey
      signature-algorithms      CDATA           #IMPLIED>

<!ELEMENT key-selection         (public-key|issuer-name|subject-name|
                                extended-key-usage|key-usage|policy-info)*>
<!ATTLIST key-selection
      policy                    CDATA           #IMPLIED
      exclude                   (yes|no|YES|NO) "no"
      require-all               (yes|no|YES|NO) "no">

<!ELEMENT public-key EMPTY>
<!ATTLIST public-key
      type                      CDATA           #REQUIRED>
<!ELEMENT issuer-name           EMPTY>
<!ATTLIST issuer-name
      name                      CDATA           #IMPLIED
      pattern                   CDATA           #IMPLIED
      match-server-certificate  (yes|no|YES|NO) "no">
<!ELEMENT subject-name          EMPTY>
<!ATTLIST subject-name
      name                      CDATA           #IMPLIED
      pattern                   CDATA           #IMPLIED>
```

```
<!ELEMENT extended-key-usage    (#PCDATA)>
<!ATTLIST extended-key-usage
      oid                       CDATA           #IMPLIED
      explicit                  (yes|no|YES|NO) "no">
<!ELEMENT key-usage             (#PCDATA)>
<!ATTLIST key-usage
      bit                       CDATA           #IMPLIED>
<!ELEMENT auth-keyboard-interactive EMPTY>
<!ELEMENT auth-gssapi           EMPTY>

<!-- Actually, the default for allow-ticket-forwarding is "no", but we
     don't want to override value if it is left undefined. -->
<!ATTLIST auth-gssapi
      dll-path                  CDATA           "/usr/lib/libgssapi_krb5.so,
                                                /usr/lib64/libgssapi_krb5.so,
                                                /usr/lib/libkrb5.so,
                                                /usr/lib/libgss.so,
                                                /usr/local/gss/gl/mech_krb5.so,
                                                /usr/local/lib/libgssapi_krb5.so,
                                                /usr/local/lib/libkrb5.so,
                                                /usr/kerberos/lib/libgssapi_krb5.so,
                                                /usr/kerberos/lib/libkrb5.so,
                                                /usr/lib/gss/libgssapi_krb5.so,
                                                /usr/kerberos/lib/libgssapi_krb5.so.2,
                                                /usr/lib/libgssapi_krb5.so.2,
                                                /usr/lib/amd64/gss/mech_krb5.so,
                                                /usr/lib/amd64/libgss.so"
      allow-ticket-forwarding   (yes|no)        #IMPLIED>

<!-- User identities. -->
<!ELEMENT user-identities       (identity*)>
<!ELEMENT identity              EMPTY>
<!ATTLIST identity
      identity-file             CDATA           #IMPLIED
      file                      CDATA           #IMPLIED
      hash                      CDATA           #IMPLIED
      id                        CDATA           #IMPLIED
      data                      CDATA           #IMPLIED>

<!-- Password. -->
<!ELEMENT password              (#PCDATA)>
<!ATTLIST password
      string                    CDATA           #IMPLIED
      file                      CDATA           #IMPLIED
      command                   CDATA           #IMPLIED>

<!-- Proxy rules. -->
<!ELEMENT proxy                 EMPTY>
<!ATTLIST proxy
      ruleset                   CDATA           #REQUIRED>

<!-- Idle timeout. -->
<!ELEMENT idle-timeout          EMPTY>
<!ATTLIST idle-timeout
```

```
      type                      (connection)    "connection"
      time                      CDATA           #IMPLIED>

<!-- Connect timeout. -->
<!ELEMENT tcp-connect-timeout   EMPTY>
<!ATTLIST tcp-connect-timeout
      time                      CDATA           #REQUIRED>

<!-- Keepalive interval. -->
<!ELEMENT keepalive-interval    EMPTY>
<!ATTLIST keepalive-interval
      time                      CDATA           #REQUIRED>

<!-- Forwards element. -->
<!ELEMENT forwards              (forward*)>

<!-- Forward. -->
<!ELEMENT forward               EMPTY>
<!ATTLIST forward
      type                      (x11|agent)     #REQUIRED
      state                     (on|off|denied) #REQUIRED>


<!-- Compression. -->
<!ELEMENT compression           EMPTY>
<!ATTLIST compression
      name                      CDATA           #IMPLIED
      level                     CDATA           #IMPLIED>

<!ELEMENT authentication-success-message EMPTY>
<!ATTLIST authentication-success-message
      enable                    (yes|no|YES|NO) "yes">

<!ELEMENT quiet-mode            EMPTY>
<!ATTLIST quiet-mode
      enable                    (yes|no|YES|NO) "no">

<!ELEMENT sftpg3-mode           EMPTY>
<!ATTLIST sftpg3-mode
      compatibility-mode        CDATA           "tectia">

<!ELEMENT terminal-selection    EMPTY>
<!ATTLIST terminal-selection
      selection-type            (select-words|select-paths)
                                                "select-words">

<!ELEMENT terminal-bell         EMPTY>
<!ATTLIST terminal-bell
      bell-style                (none|pc-speaker|system-default)
                                                "system-default">

<!ELEMENT close-window-on-disconnect EMPTY>
<!ATTLIST close-window-on-disconnect
      enable                    (yes|no)        "no">
```

```
<!ELEMENT checksum EMPTY>
<!ATTLIST checksum
      type          (yes|no|md5|sha1|md5-force|sha1-force|checkpoint|
                     YES|NO|MD5|SHA1|MD5-FORCE|SHA1-FORCE|CHECKPOINT) "yes">

<!ELEMENT user-config-directory EMPTY>
<!ATTLIST user-config-directory
      path                      CDATA           "%USER_CONFIG_DIRECTORY%">

<!ELEMENT file-access-control   EMPTY>
<!ATTLIST file-access-control
      enable                    (yes|no|YES|NO) "no">

<!-- address-family mode setting ipv4 & ipv6-->
<!ELEMENT address-family        EMPTY>
<!ATTLIST address-family
      type                      (any|inet|inet6)
                                              "&default-address-family-type;">

<!ELEMENT protocol-parameters   EMPTY>
<!ATTLIST protocol-parameters
      threads                   CDATA           #IMPLIED>

<!-- Profiles element. -->
<!ELEMENT profiles              (profile*)>

<!-- Connection profile.  No element may appear multiple times. -->
<!ELEMENT profile               (hostkey|ciphers|macs|kexs|hostkey-algorithms|
                                 transport-distribution|rekey|
                                 authentication-methods|
                                 user-identities|
                                 compression|proxy|idle-timeout|
                                 tcp-connect-timeout|keepalive-interval|
                                 exclusive-connection|server-banners|
                                 forwards|tunnels|extended|remote-environment|
                                 server-authentication-methods|password|
                                 profile-group)*>
<!ATTLIST profile
      id                        CDATA           #IMPLIED
      name                      CDATA           #IMPLIED
      host                      CDATA           #REQUIRED
      port                      CDATA           "22"
      protocol                  CDATA           "secsh2"
      host-type                 (unix|windows|default) "default"
      connect-on-startup        (yes|no|YES|NO) "no"
      user                      CDATA           #IMPLIED
      gateway-profile           CDATA           #IMPLIED>

<!ELEMENT profile-group EMPTY>
<!ATTLIST profile-group
      name                      CDATA           #REQUIRED>

<!-- Hostkey. -->
```

```
<!ELEMENT hostkey               (#PCDATA)>
<!ATTLIST hostkey
      file                      CDATA           #IMPLIED>

<!-- Tunnels element. -->
<!ELEMENT tunnels               (local-tunnel*,remote-tunnel*)>

<!-- Local tunnel. -->
<!ELEMENT local-tunnel          EMPTY>
<!ATTLIST local-tunnel
      type                      CDATA           "tcp"
      listen-address            CDATA           "127.0.0.1"
      listen-port               CDATA           #REQUIRED
      dst-host                  CDATA           "127.0.0.1"
      dst-port                  CDATA           #REQUIRED
      allow-relay               (yes|no|YES|NO) "no">

<!-- Remote tunnel. -->
<!ELEMENT remote-tunnel         EMPTY>
<!ATTLIST remote-tunnel
      type                      CDATA           "tcp"
      listen-address            CDATA           "127.0.0.1"
      listen-port               CDATA           #REQUIRED
      dst-host                  CDATA           "127.0.0.1"
      dst-port                  CDATA           #REQUIRED
      allow-relay               (yes|no|YES|NO) "no">

<!-- Static tunnels element. -->
<!ELEMENT static-tunnels        (tunnel*)>

<!-- Static tunnel. -->
<!ELEMENT tunnel                EMPTY>
<!ATTLIST tunnel
      type                      CDATA           "tcp"
      listen-address            CDATA           "127.0.0.1"
      listen-port               CDATA           #REQUIRED
      dst-host                  CDATA           "127.0.0.1"
      dst-port                  CDATA           #REQUIRED
      allow-relay               (yes|no|YES|NO) "no"
      profile                   CDATA           #REQUIRED>

<!-- GUI. -->
<!ELEMENT gui                   EMPTY>
<!ATTLIST gui
      hide-tray-icon            (yes|no|YES|NO) "no"
      show-exit-button          (yes|no|YES|NO) "yes"
      show-admin                (yes|no|YES|NO) "yes"
      enable-connector          (yes|no|YES|NO) "yes"
      show-security-notification (yes|no|YES|NO) "yes">

<!ELEMENT filter-engine         (network|dns|filter|rule)*>
<!ATTLIST filter-engine
      ip-generate-start         CDATA           "198.18.0.1"
      ip6-generate-start        CDATA           "2001:db8::ff00:42:8329"
```

```
      ftp-filter-at-signs       (yes|no|YES|NO) "no">

<!ELEMENT network               EMPTY>
<!ATTLIST network
      id                        ID              #REQUIRED
      address                   CDATA           #IMPLIED
      domain                    CDATA           #IMPLIED
      ip-generate-start         CDATA           #IMPLIED
      ip6-generate-start        CDATA           #IMPLIED>

<!ELEMENT dns                   EMPTY>
<!ATTLIST dns
      id                        ID              #REQUIRED
      network-id                IDREF           #IMPLIED
      application               CDATA           #IMPLIED
      host                      CDATA           #IMPLIED
      ip-address                CDATA           #IMPLIED
      pseudo-ip                 (yes|no|YES|NO) "no">

<!ELEMENT filter                EMPTY>
<!ATTLIST filter
      dns-id                    IDREF           #REQUIRED
      ports                     CDATA           #REQUIRED
      action                    (block|direct|tunnel|ftp-tunnel|ftp-proxy|
                                BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
                                                #REQUIRED
      profile-id                CDATA           #IMPLIED
      destination               CDATA           #IMPLIED
      destination-port          CDATA           #IMPLIED
      fallback-to-plain         (yes|no|YES|NO) "no">

<!ELEMENT rule                  EMPTY>
<!ATTLIST rule
      application               CDATA           #IMPLIED
      host                      CDATA           #IMPLIED
      ip-address                CDATA           #IMPLIED
      pseudo-ip                 (yes|no|YES|NO) "no"
      ports                     CDATA           #REQUIRED
      action                    (block|direct|tunnel|ftp-tunnel|ftp-proxy|
                                BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
                                                #REQUIRED
      profile-id                CDATA           #IMPLIED
      destination               CDATA           #IMPLIED
      destination-port          CDATA           #IMPLIED
      username                  CDATA           #IMPLIED
      hostname-from-app         (yes|no|YES|NO) "no"
      username-from-app         (yes|no|YES|NO) "no"
      fallback-to-plain         (yes|no|YES|NO) "no"
      show-sftp-server-banner   (yes|no|YES|NO) "no">


<!ELEMENT logging               (log-target*,log-events*)>

<!-- Log events. -->
```

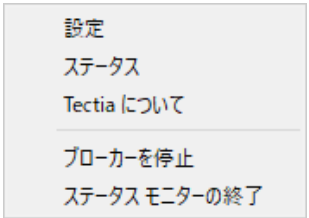
```
<!-- Log event facility. -->
<!ENTITY default-log-event-facility       "normal">

<!-- Log event severity. -->
<!ENTITY default-log-event-severity       "notice">

<!ELEMENT log-target            EMPTY>
<!ATTLIST log-target
      file                      CDATA                           #IMPLIED
      type                      (file|syslog|socket|discard)    "file"
      format                    (syslog|csv|xml)                "syslog" >

<!ELEMENT log-events            (log-target|#PCDATA)*>
<!ATTLIST log-events
      facility      (normal|daemon|user|auth|local0|local1|
                     local2|local3|local4|local5|local6|local7|discard)
                                        "&default-log-event-facility;"
      severity      (informational|notice|warning|error|critical|
                     security-success|security-failure)
                                        "&default-log-event-severity;">
```

A.6. Tectia のショートカット・メニュー (Windows 及び Linux)

Windows または Linux で Tectia Client (または接続ブローカー) が実行されているとき、Tectia コネクション設定 GUI の [一般] 設定で表示されるように選択されている場合は、Tectia のアイコン が Windows タスクバーの通知領域 (通常はデスクトップ下部の時刻表示の横) に表示されます。

ショートカット・メニューを開くには、Tectia のアイコンを右クリックします。

ショートカット・メニューの内容は、 [一般] ビューの Tectia コネクション設定 GUI で変更できます (A.1.2 を参照)。



図A.46 ステータス・ウィンドウのショートカット・メニュー

メニューには以下のオプションがあります。

- **[設定]** を選択すると Tectia コネクション設定　GUI が開きます。
- **[ステータス]** を選択すると **[Tectia 接続ステータス GUI]** が開き、接続ブローカーに関する情報を確認できます。詳細については、A.6.1 を参照してください。
- **[Tectia について]** を選択すると、インストールされている Tectia Client のバージョンが表示されます。
- **[ブローカーを停止]** を選択すると接続ブローカーが停止し、開いている接続がすべて閉じられます。
- **[ステータス モニターの終了]** を選択すると、Tectia 接続ステータス GUI が終了します。

A.6.1. Tectia 接続ステータス GUI

Tectia 接続ブローカーのステータスは **Tectia 接続ステータス GUI** で確認できます。

Tectia 接続ステータス GUI を開くには、Windows タスクバーの通知領域で Tectia アイコン をクリックするか、同アイコンを右クリックしてショートカット・メニューから **[ステータス]** を選択します。Tectia 接続ステータス GUI では **[接続]** ビュー、**[鍵]** ビュー、及び **[ログ]** ビューにアクセスできます。それぞれのビューを表示するには、ダイアログボックスの左にあるビュー・アイコンをクリックします。

接続ビュー

Tectia 接続ステータス GUI の **[接続]** ビューには、現在アクティブな、コンピュータとの間のセキュアな接続 (ターミナル、トンネル、または SFTP) が表示されます。

図A.47 ステータス・ウィンドウの接続ビュー

各接続について、以下の情報が表示されます。

- **接続:** `user@host#port` の形式で示された接続先

- **ID**: 接続しているプログラムの識別子 (接続ブローカーが発行する番号)
- **アップロード**: アップロードされたデータ量 (バイト単位)
- **ダウンロード**: ダウンロードされたデータ量 (バイト単位)
- **アップロード速度**: アップロード速度 (キロバイト/秒単位)
- **ダウンロード速度**: ダウンロード速度 (キロバイト/秒単位)

接続の詳細を表示するには、接続を右クリックし、[**接続ステータスを表示**] を選択します。このビューには、認証、トランスポートおよび鍵交換の方法、転送されたデータ量、及び接続中のチャネルに関する情報が表示されます。

チャネルの詳細を表示するには、チャネルを右クリックし、[**チャネル ステータスを表示**] を選択します。このビューには、チャネルの種類、接続ID とホスト、期間、及び転送されたデータ量に関する情報が表示されます。

鍵をホストにアップロードするには、接続またはチャネルを右クリックし、[**鍵をアップロード**] を選択します。[**公開鍵認証ウィザード**] が開き、そこでアップロードする鍵を選択してから、鍵をアップロードできます。アップロードに成功すると、アップロードのステータスや鍵のアップロード先が表示されます。詳細については、「公開鍵認証ウィザードの使用」 を参照してください。

開いている接続を閉じるには、接続を右クリックし、[**切断**] を選択します。

接続中のチャネルを閉じるには、チャネルを右クリックし、[**チャネルを終了**] を選択します。

鍵ビュー

Tectia 接続ステータス GUI の [**鍵**] ビューには、公開鍵認証に使用できる公開鍵と証明書が表示されます。

図A.48 ステータス・ウィンドウの 鍵ビュー

各鍵または証明書について、以下の情報が表示されます。

- **ID**: 鍵の識別子 (接続ブローカーが発行する番号)
- **プロバイダ**: 鍵プロバイダの名前
- **ファイル名**: 鍵ファイルの名前
- **ステータス**: 鍵のステータス。以下のステータスがあります。
  - **locked** - ファイルはパスフレーズまたは PIN で保護されており、接続ブローカーに提供されていません。[**PINを入力**] のショートカット・コマンドを使用してパスフレーズを入力すると、鍵のロックが解除されます。
  - **open** - パスフレーズまたは PIN は接続ブローカーに提供されています。
- **種類**: 鍵の種類 (RSA、DSA、ECDSA、または Ed25519) または証明書の種類 (X.509 証明書)
- **コメント**: 鍵の長さ、作成者、作成日時など、鍵に関する詳細

公開鍵の詳細を表示するには、公開鍵を右クリックし、[**鍵の情報を表示**] を選択します。このビューには、鍵の種類と長さ、鍵の名前、場所、及びフィンガープリントに関する情報、及びパスフレーズなどの認証コードが提供されているかどうかが表示されます。

キャッシュからパスフレーズまたは PIN を消去するには、鍵を右クリックし、[**キャッシュされたPINを消去**] を選択します。鍵のステータスがロック状態に変わります。

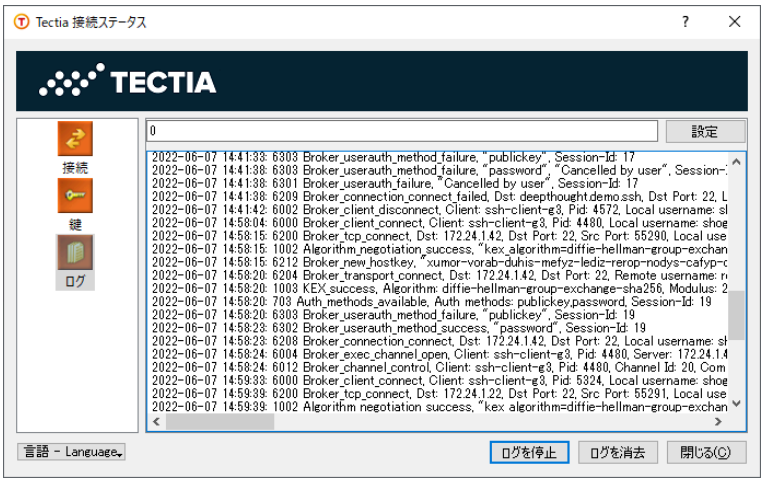
パスフレーズまたは PIN をキャッシュに保存するには、ロックされている公開鍵を右クリックし、[**PINを入力**] を選択します。パスフレーズまたは PIN の入力を求めるダイアログが開きます。鍵のステータスがロック解除状態に変わります。

公開鍵を保存するには、鍵を右クリックし、[**公開鍵を保存**] を選択します。[**公開鍵を保存**] ダイアログが開き、鍵を保存できます。

公開鍵をホストにアップロードするには、鍵を右クリックし、[**公開鍵をアップロード**] を選択します。[**公開鍵認証ウィザード**] が開き、そこでホストへの接続を定義してから、公開鍵をアップロードできます。アップロードに成功すると、アップロードのステータスや鍵のアップロード先が表示されます。詳細については、「公開鍵認証ウィザードの使用」 を参照してください。

ログ・ビュー

Tectia 接続ステータス GUI の [**ログ**] ビューには、現在のセキュアな接続のログ情報が表示されます。



図A.49 ステータス・ウィンドウのログ・ビュー

デバッグ・レベルを設定するには、フィールドに値を入力し、[**設定**] をクリックするか、または `Enter` キーを押します。

マウスを使ってログからテキストを選択し、 **Ctrl+C** キーでクリップボードにコピーできます。 **Ctrl+A** キーを押すと、ログのすべての内容が選択されます。これらのコマンドは、ショートカット・メニュー (ログを右クリックして表示) からも利用できます。

ログの作成を一時停止するには、[**ログを停止**] をクリックします。

ログの作成を再開するには、[**ログを再開**] をクリックします。

ログを消去するには、[**ログを消去**] をクリックします。

付録B Tectia SSHターミナル GUI 及び Tectia セキュア・ファイル転送 GUI の設定 (Windows)

Tectia のユーザ・インターフェイスを設定するには、ツールバーの [**ユーザインターフェースの設定**] ボタン をクリックするか、または [**編集**] → [**ユーザインターフェースの設定**] オプションを選択します。

[**設定**] ダイアログの左側に、さまざまな設定のカテゴリがツリー構造で表示されます。

ブランチをクリックすると、そのブランチに関連する設定が表示されます。設定を変更するには、[設定]　ダイアログの右側に表示される選択項目を変更します。一部の設定は、設定を保存してから新しいターミナル・ウィンドウまたはファイル転送ウィンドウを開くか、新しい接続を開始するまで有効にならないことに注意してください。

注意

Tectia Client には 2 つの設定ツールがあります。

- **Tectia コネクション設定　GUI** (アイコンをクリックして表示) は、接続の設定を編集する場合に使用します。
- **ユーザインターフェースの設定**ツール　(アイコンをクリックして表示) は、Tectia SSHターミナル GUI 及び Tectia セキュア・ファイル転送 GUI を編集する場合に使用します。

接続の設定方法については、A.1 を参照してください。

B.1. グローバル設定の定義

グローバル設定はリモート・ホスト・コンピュータへのすべての接続で共通です。グローバル設定はユーザ・プロファイルのディレクトリ ("%APPDATA%\SSH") に global.dat というファイル名で保存されます。

図B.1 [設定] ダイアログの [グローバル設定] ページ

B.1.1. 外観の定義

アプリケーション及びターミナル・ウィンドウの外観は [**設定**] ダイアログの [**外観**] ページで設定します。

図B.2 [設定] ダイアログの [外観] ページ

Office XP 風の外観

[Office XP 風の外観] チェックボックスを選択すると、メニュー・バーやツールバーの表示方法を Microsoft Office XP 風の外観に合わせて変更できます。

ターミナル設定

[ターミナル設定] オプションでは、ターミナル・ウィンドウの動作を定義できます。

右クリック時に選択範囲を貼り付け

このチェックボックスを選択すると、ターミナル・ディスプレイに表示されているテキストを簡単にコピーできます。このオプションを選択すると、テキストをを強調表示するだけコピーし、マウスの右ボタンをクリックしてテキストを貼り付けることができます。

出力時に最下行までスクロール

このチェックボックスを選択すると、新しいテキストが出力されたときは常に、ターミナル・ウィンドウが最下行までスクロールします。このオプションが選択されていない　と、新しいテキスト行が表示されてもウィンドウは最下行までスクロールしません。デフォルトでは、このオプションは選択されています。

ターミナルのバッファ行数

このテキスト・ボックスには、スクロールバック・バッファに集める行数を入力します。値が大きいほど、以前のターミナル出力を確認する際のターミナル・ディスプレイのスクロールバック量が大きくなります。デフォルト値は 500 行です。

クリックでURLを開く

このチェックボックスを選択すると、ターミナル・ウィンドウ内のリンクをクリック操作で開くことができます。このオプションはデフォルトで選択されています。

タイトル バー

[タイトル・バー] の設定は、ターミナル・ウィンドウとファイル転送ウィンドウのタイトル・バーに表示される内容に影響します。

[**プロファイル名またはホスト名を表示**] チェックボックスを選択すると、プロファイルを使用している場合に、現在接続しているリモート・ホスト・コンピュータのプロファイル名がタイトル・バーに表示されます。プロファイルを使用しない場合は、ホスト名が表示されます。

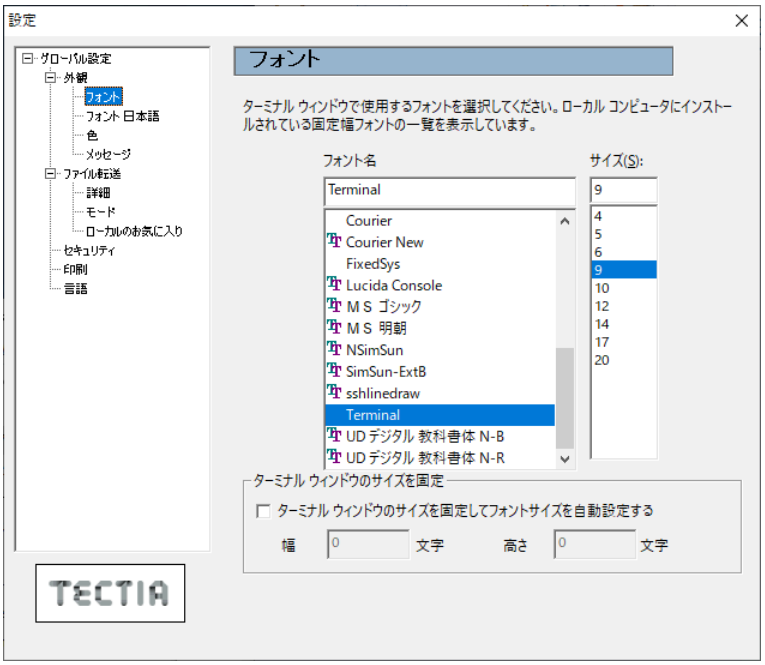
ウィンドウ レイアウト

複数のウィンドウを同時に開いた状態で接続プロファイルを作成し、レイアウトを保存した場合、そのプロファイルを選択すると、通常はプロファイルに関連するすべてのウィンドウが開きます。[ウィンドウ・レイアウト] オプションを使用すると、この動作を上書きできます。

[**プロファイルのすべてのウィンドウを開く**] チェックボックスを選択すると、プロファイルを選択したときに、そのプロファイルに関連するすべてのウィンドウが開きます。このオプションを選択解除すると、新しいウィンドウを開いたときに、他のウィンドウは設定された位置で開きます。デフォルトでは、このオプションは有効になっています。

B.1.2. フォントとターミナル・ウィンドウのサイズの選択

ターミナル・ウィンドウで使用するフォントは、[**設定**] ダイアログの [**フォント**] ページで選択できます。これと同じページで、ターミナル・ウィンドウのサイズを固定するか、フォント・サイズに応じて自動的に調整するかを選択することもできます。



図B.3 [設定] ダイアログの [フォント] ページ

フォント

使用するフォントとそのサイズを選択できます。新しいフォントの設定は、[**OK**]　をクリックするとすぐにターミナル・ウィンドウに反映されます。変更を取り消す場合は、[**キャンセル**] をクリックします。

フォント名

目的のフォントを [**フォント名**] リストから選択します。このリストには、ローカル・コンピュータにインストールされている等幅 (固定幅) のフォントが表示されます。プロポーショナル・フォントはターミナル・ウィンドウに適していないため選択できません。

サイズ

目的のフォント・サイズを [**サイズ**] リストから選択します。フォント・サイズはターミナル・ウィンドウのサイズに影響するため、選択したフォントが小さいほど、ターミナル・ウィンドウも小さくなることに注意してください。ただし、フォント・サイズを変更した後でもターミナル・ウィンドウのサイズは変更できますが、フォント・サイズは選択されたサイズのままになります。

ターミナル ウィンドウのサイズを固定

Tectia　ターミナルのサイズを固定できます。デフォルトではターミナルのサイズは固定されておらず、フォント・サイズに応じて調整されます。[ターミナル・ウィンドウのサイズを固定] の設定に加えた変更は、[**OK**] をクリックするとすぐに有効になります。変更を取り消す場合は、[**キャンセル**] をクリックします。

ターミナル ウィンドウのサイズを固定してフォントサイズを自動設定する

固定されたサイズのターミナル・ウィンドウを使用するには、このチェックボックスを選択します。ウィンドウの幅と高さを定義できます。フォント・サイズはウィンドウに合わせて自動で調整されます。フォント・サイズの選択は無効になりますが、使用するフォントを選択することはできます。

幅

固定サイズのターミナル・ウィンドウの幅を文字数で定義します。プログラムのデフォルト値は 80 文字です。

高さ

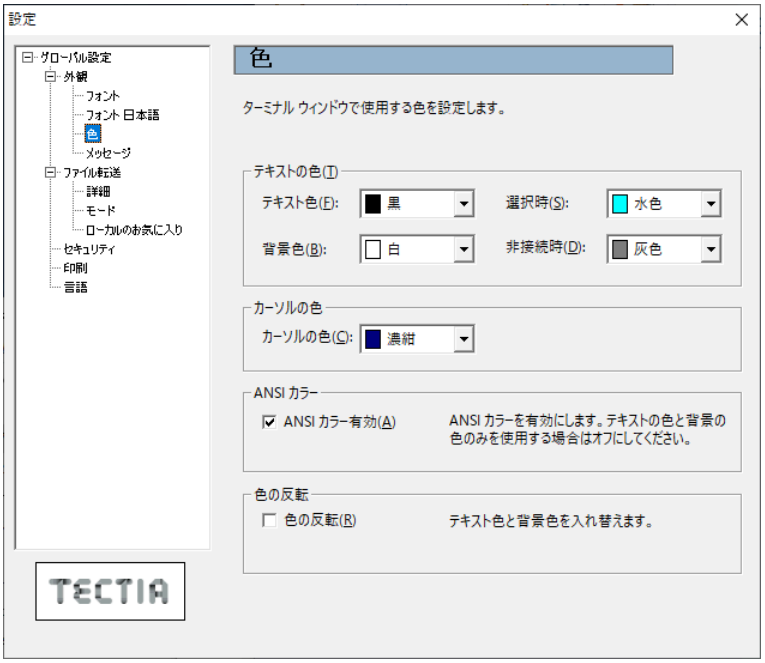
固定サイズのターミナル・ウィンドウの高さを文字数で定義します。プログラムのデフォルト値は 24 文字です。

B.1.3. 色の選択

ターミナル・ウィンドウで使用する色は、[**設定**] ダイアログの [**色**] ページで選択できます。新しい色の設定は、[**OK**] をクリックするとすぐに有効になります。

[グローバル設定] で定義されている色の設定は、すべての接続プロファイルに影響します。

ターミナルの色を変更しても、すでにターミナル・ウィンドウに表示されている内容には影響しませんが、この時点からは、選択した配色を使用してテキスト出力が行われることに注意してください。



図B.4 [設定] ダイアログの [色] ページ

テキストの色

テキストの色は、接続されているウィンドウと接続されていないウィンドウの両方で、ターミナル・ウィンドウの背景色及びテキストの色に影響します。

テキスト色

希望するテキスト色をドロップダウン・メニューから選択します。テキスト色は、リモート・ホスト・コンピュータと接続しているウィンドウのテキストに使用されます。色は 16 色から選択できます。デフォルトのテキスト色は黒色です。

背景色

希望する背景色をドロップダウン・メニューから選択します。色は 16 色から選択できます。デフォルトの背景色は白色です。

選択時

マウスでテキストを選択したときに、背景色として使用する色をドロップダウン・メニューで選択します。色は 16 色から選択できます。デフォルトの選択色は水色です。

非接続時

リモート・ホスト・コンピュータに接続されていないターミナル・ウィンドウでテキスト色として使用する色をドロップダウン・メニューで選択します。色は 16 色から選択できます。灰色は、接続されていないターミナル・ウィンドウのデフォルトのテキスト色です。

カーソルの色

希望するカーソルの色をドロップダウン・メニューから選択します。色は 16 色から選択できます。デフォルトのカーソルの色は濃紺色です。

ANSI カラー

ANSI 制御コードを使用して、ターミナル・ウィンドウのテキストの色を変更できます。ANSI カラーの設定では、この機能を使用するかどうかを選択できます。ANSI カラーを無効にしている場合でも、ターミナル・ウィンドウで使用するテキストの色や背景色を自由に選択できます。

ターミナル・ウィンドウで ANSI カラーを使用できるようにするには、[**ANSI カラー有効**] チェックボックスを選択します。デフォルトでは、ANSI カラーが選択されています。

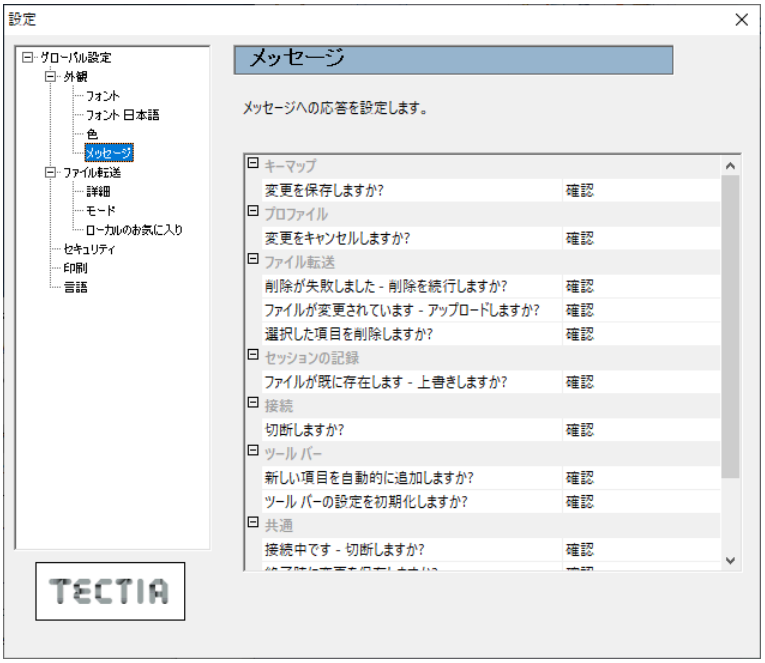
色の反転

表示色を反転させることで、ポジティブ (明に暗) 表示からネガティブ (暗に明) 表示へと素早く切り替えて、視認性を向上させることができます。

テキスト色と背景色を入れ替えるには、[**色の反転**] チェックボックスを選択します。この設定は、[**OK**] をクリックした時点でターミナル・ウィンドウ全体に影響します。

B.1.4. メッセージの定義

[**設定**] ダイアログの [**メッセージ**] ページでは、通常はユーザの確認を必要とする標準的なメッセージに対するデフォルトの返答を設定できます。メッセージはいくつかのカテゴリに分かれて表示されます。



図B.5 表示する確認ダイアログの指定

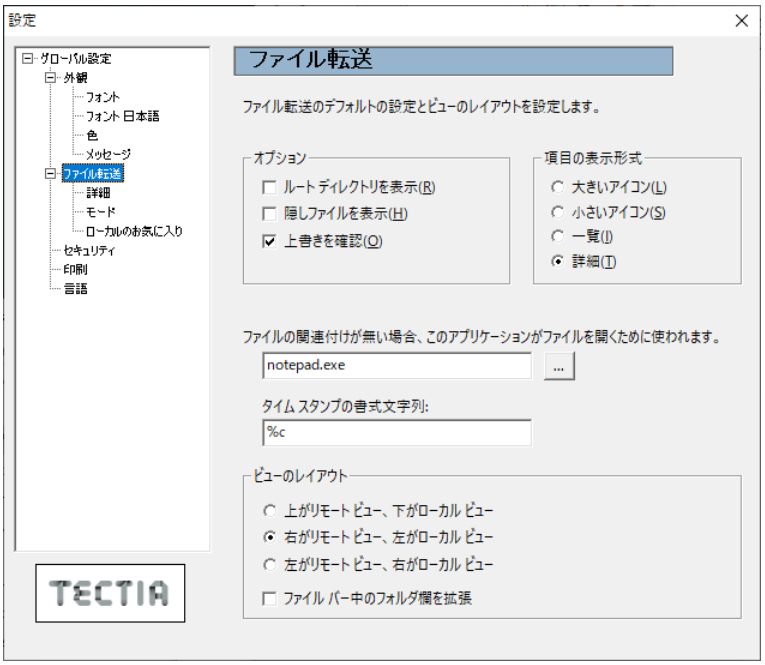
それぞれの確認は、アクションを自動的に受け入れる (はい) か拒否する (いいえ) か、またはユーザに確認を求める (確認) かを設定できます。デフォルトでは、すべてのメッセージはユーザにアクションを確認するよう求めます。

B.1.5. ファイル転送設定の定義

Tectia セキュア・ファイル転送 GUI のデフォルト設定は [**ユーザインターフェースの設定**] ダイアログの [**ファイル転送**] ページで行えます。新しい設定は、その後に起動するファイル転送ウィンドウに影響します。

注意

ここで行うファイル転送の設定は、Tectia のファイル転送 GUI にのみ影響し、SFTP 変換や、ファイル転送に関連するコマンドライン・ツールには影響しません。



図B.6 [設定] ダイアログのグローバルな [ファイル転送] ページ

オプション

以下のオプションがあります。

ルート ディレクトリを表示

このチェックボックスを選択すると、ファイル転送ウィンドウにルート・ディレクトリがデフォルトで表示されます。

隠しファイルを表示

このチェックボックスを選択すると、ファイル転送ウィンドウに隠しファイルがデフォルトで表示されます。

上書きを確認

ターゲット・システムにすでに存在するファイルを転送しようとするときに、ファイル転送ユーティリティが確認を求めるようにする場合は、このチェックボックスを選択します。

項目の表示形式

この設定では、ファイル転送ウィンドウのデフォルト表示形式を 4 種類の表示形式から選択できます。

大きいアイコン

このオプションを選択すると、ファイル転送のファイル表示が[大きいアイコン]で表示さ れます。各ファイルやフォルダが大きいアイコンで表示され、わかりやすい整然とした表示になります。

小さいアイコン

このオプションを選択すると、ファイル転送のファイル表示が[小さいアイコン]で表示されます。各ファイルやフォルダが小さいアイコンで表示されます。これにより、[大きいアイコン]の場合に比べて数倍の項目が表示されます。

一覧

このオプションを選択すると、ファイル転送のファイル表示が [一覧] で表示されます。各ファイルとフォルダは小さいで表示され、ファイルとフォルダが 1 列に表示されます。

詳細

このオプションを選択すると、ファイル転送フォルダ表示が [詳細] 表示で表示されます。ファイルとフォルダは小さいアイコンで表示され、ファイル名、ファイル・サイズ、ファイル・タイプ、最終更新日時、及び属性が表示されます。

[ファイル] ビューの上部にある [**名前**]、[**サイズ**]、[**種類**]、[**更新日時**]、及び [**属性**] のソート・バーをクリックすると、ファイル名、ファイル・サイズ、ファイル・タイプ、最終更新日時に基づいてファイルとフォルダが並べ替えられます。同じ並べ替えオプションをもう一度クリックすると、並び順が逆になります。

並べ替え機能では大文字と小文字は区別されません。大文字のテキストは小文字のテキストと一緒に並べ替えられます。

ファイル・タイプの関連付けはローカル・コンピュータの設定に基づきます。特定のファイル名拡張子を持つファイルに対して新しいファイル・タイプの説明を定義した場合、リモート・コンピュータ内のファイルもそのタイプのファイルとして表示されます。これにより、ホスト・コンピュータでも特定のファイル・タイプを認識しやすくなります。

ファイルの関連付けが無い場合、このアプリケーションがファイルを開くために使われます。

Tectia Client は、Windows エクスプローラと同じ方法でファイル・タイプを関連付けます。ファイル転送ウィンドウでファイルをダブルクリックすると、そのファイル・タイプが関連付けられているアプリケーションでファイルが開きます。

すべてのファイル・タイプにアプリケーションが関連付けられているわけではありません。このフィールドでは、ファイル・タイプの関連付けがないファイルを開くためのアプリケーションを定義できます。デフォルトのアプリケーションは、テキストを含むファイルに最適なメモ帳です。

不明なファイル・タイプのデフォルトの関連付けを変更するには、テキスト・フィールドの横にあるボタンをクリックします。[アプリケーションの選択] ダイアログが表示され、目的のアプリケーションを選択できます。

タイム スタンプの書式文字列

書式文字列のフィールドには、ファイル転送ウィンドウでのファイルのタイム・スタンプの表示形式を定義する文字列を入力できます。デフォルト値は %c で、Windows の国または地域の設定 (ロケール) で定義された形式で日付と時刻が表示されます。

タイム・スタンプの形式を変更するには、デフォルト値を、以下の文字の組み合わせで構成される文字列に置き換えます。

%a

短縮された曜日名

%A

完全な曜日名

%b

短縮された月名

%B

完全な月名

%c

ロケールに適した日付と時刻の表現

%d

数字で表した日付 (01 〜 31)

%H

24 時制の時 (00 〜 23)

%I

12 時制の時 (01 〜 12)

%j

数字で表した通日 (001 〜 366)

%m

数字で表した月 (01 〜 12)

%M

数字で表した分 (00 〜 59)

%p

現在のロケールの 12 時制の午前/午後表示

%S

数字で表した秒 (00 〜 59)

%U

日曜日を週の初日とする数字で表した週番号 (00 〜 53)

%w

数字で表した曜日 (0 〜 6、日曜日が 0)

%W

月曜日を週の初日とする数字で表した週番号 (00 〜 53)

%x

現在のロケールの日付表現

%X

現在のロケールの時刻表現

%y

数字で表した、世紀を除いた年 (00 〜 99)

%Y

数字で表した、世紀を含む年

%z, %Z

タイムゾーンの名称または略称。タイムゾーンが不明な場合は文字なし

%%

パーセント記号

ビューのレイアウト

ファイル転送ウィンドウでは、ローカル・ビュー・ペインとリモート・ビュー・ペインをどのように配置するかを選択できます。以下のオプションがあります。

- **上がリモート ビュー、下がローカル ビュー**

- **右がリモート ビュー、左がローカル ビュー**
- **左がリモート ビュー、右がローカル ビュー**

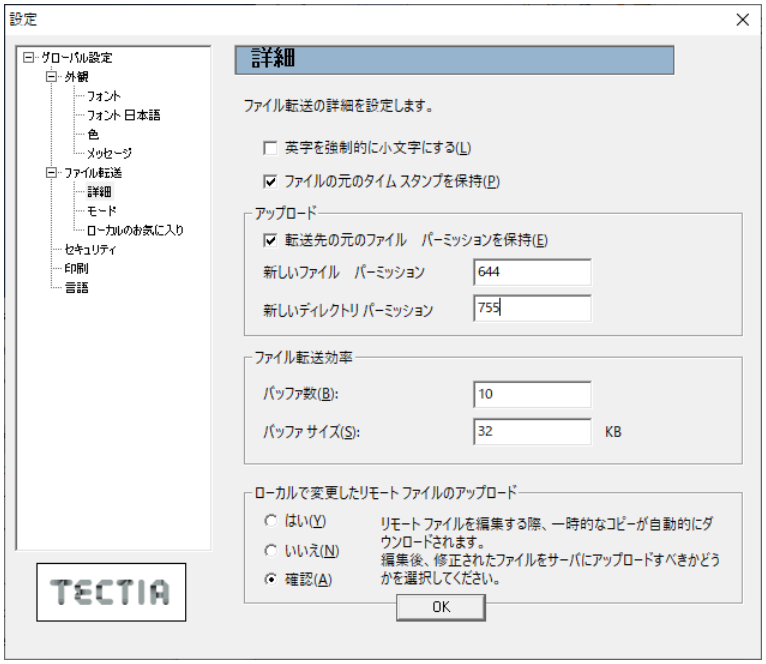
[**ファイル バー中のフォルダ欄を拡張**] チェックボックスを選択すると、ファイル・バーのボタンが少なくなり、お気に入りフォルダ・リストの表示領域が広くなります。

B.1.6. 詳細なファイル転送オプションの定義

[**設定**] ダイアログの [**詳細**] ページでは、ファイル転送の GUI オプションを追加で設定できます。新しい設定は、その後に起動するファイル転送ウィンドウに影響します。

注意

ここで行うファイル転送の設定は、Tectia のファイル転送 GUI にのみ影響し、SFTP 変換や、ファイル転送に関連するコマンドライン・ツールには影響しません。



図B.7 詳細なファイル転送オプション

ファイル転送の詳細を設定します。

以下の設定はファイル転送に影響します。

英字を強制的に小文字にする

このオプションを選択すると、ファイル転送時に小文字のファイル名が強制されます。

ファイルの元のタイム スタンプを保持

転送されたファイルでも元のタイム・スタンプの値を保持する場合は、このチェックボックスを選択します。このオプションを選択解除すると、転送されたファイルのタイム・スタンプは転送時の日時になります。

アップロード

以下の設定はアップロードに影響します。

転送先の元のファイル　パーミッションを保持

このチェックボックスを選択すると、サーバ上のファイル・パーミッションが保持されます。転送されたファイルが既存のファイルを上書きする場合は、元のファイルと同じファイル・パーミッションが使用されます。ファイルが新規の場合、サーバのターゲット・ディレクトリのデフォルトのパーミッション・マスクが使用されます。

このチェックボックスを選択解除すると、アップロードされたファイルに対して新しいファイル・パーミッションが強制的に適用されます。パーミッションは、以下の [**新しいファイル パーミッション**] 及び [**新しいディレクトリ パーミッション**] で定義します。

新しいファイル パーミッション

アップロードされたファイルの値として使用される、8 進数の Unix ファイル・パーミッション・マスク (Unix の `chmod` コマンドと同様) を入力します。

8 進数 (base-8) の表記法は 3 桁で構成されています。1 桁目はファイルの所有者に与えられるパーミッション、2 桁目はそのファイルのユーザ・グループに対するパーミッション、最後の 1 桁はそれ以外のすべてのユーザに与えられるパーミッションを指定します。

3 桁の数字はそれぞれ、以下のいずれかの値になります。

- **0**: パーミッションはありません。
- **1**: ファイルの実行が可能です。
- **2**: ファイルへの書き込みが可能です。
- **3**: ファイルへの書き込みと実行が可能です。(2 + 1 = 3)
- **4**: ファイルの読み取りが可能です。
- **5**: ファイルの読み取りと実行が可能です。(4 + 1 = 5)
- **6**: ファイルの読み取りと書き込みが可能です。(4 + 2 = 6)
- **7**: ファイルの読み取りと書き込み、実行が可能です。(4 + 2 + 1 = 7)

たとえば **644** (シンボル表記法で `-rw-r--r--`) は、ファイルの所有者にはファイルの読み取りと書き込みのパーミッションがあり、ユーザ・グループや他のユーザにはファイルの読み取りのパーミッションだけがあることを指定します。

新しいディレクトリ パーミッション

アップロードされたディレクトリの値として使用される、8 進数の Unix ディレクトリ・パーミッション・マスク (Unix の `chmod` コマンドと同様) を入力します。

表記法は [**新しいファイル パーミッション**] の場合と同じです。

ファイル転送効率

以下の設定はファイル転送プロセスに影響します。

バッファ数

ファイル転送で使用するバッファの数を入力します。デフォルト値は 10 です。

バッファ サイズ

デフォルトのバッファ・サイズ (キロバイト単位) を入力します。デフォルト値は 32 キロバイトです。

ローカルで変更したリモート ファイルのアップグレード

この選択は、リモート・ホスト・コンピュータに保存されているファイルをローカルで編集した場合の Tectia Client の動作に影響します。

はい

このオプションを選択すると、ローカルで変更されたファイルがリモート・ホスト・コンピュータにアップロードされます。

いいえ

このオプションを選択すると、ローカルで変更されたファイルはリモート・ホスト・コンピュータにアップロードされません。

確認

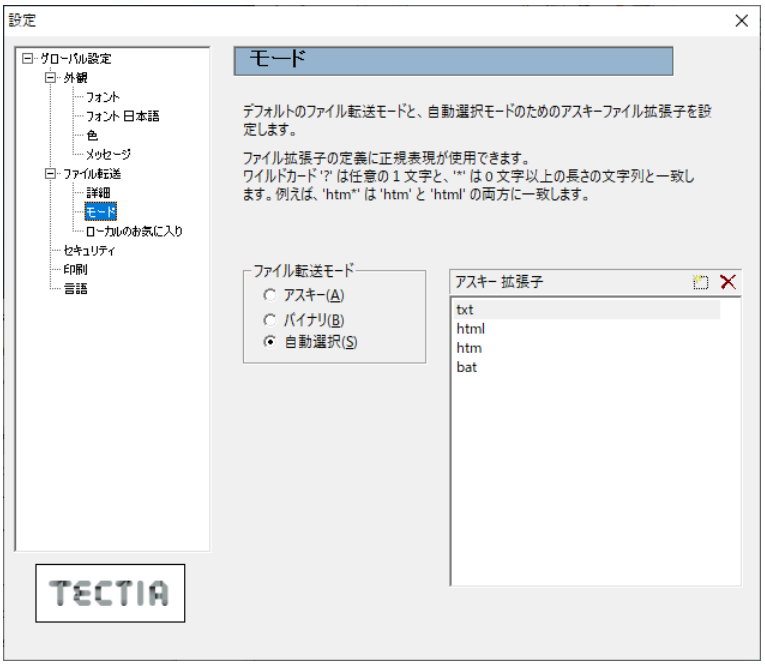
このオプションを選択すると、Tectia Client は、ローカルで変更されたファイルをアップロードするかどうかを決定するよう求めます。

B.1.7. ファイル転送モードの定義

[**設定**] ダイアログの [**モード**] ページは、どのファイルをアスキー・モードで転送するかに影響を与えます。

注意

ここで行うファイル転送の設定は、Tectia のファイル転送 GUI にのみ影響し、SFTP 変換や、ファイル転送に関連するコマンドライン・ツールには影響しません。



図B.8 ファイル転送モードの選択

ファイル転送モード

デフォルトのファイル転送モードを以下のオプションから選択します。

アスキー

デフォルトでは、すべてのファイルがアスキー・モードで転送されます。

バイナリ

デフォルトでは、すべてのファイルがバイナリ・モードで転送されます。

自動選択

[アスキー拡張子] リストで指定された拡張子を使用するファイルは、アスキー・モードで転送されます。それ以外のファイルはバイナリ・モードで転送されます。

アスキー 拡張子

[アスキー拡張子]　リストで指定された拡張子を使用するファイルは、アスキー・モードを使って転送されます。

新規作成

新しいファイル拡張子をリストに追加するには、[**新規作成**] アイコン ([**アスキー 拡張子**] リストの右上) をクリックします。[新規作成] アイコンのキーボード・ショートカットは `Ins` キーです。

ファイル拡張子はワイルドカード文字で指定できます。`?` は任意の 1 文字に一致し、`*` は任意の 0 またはそれ以上の文字に一致します。たとえば、`htm*` は `htm` と `html` の両方に一致します。

削除

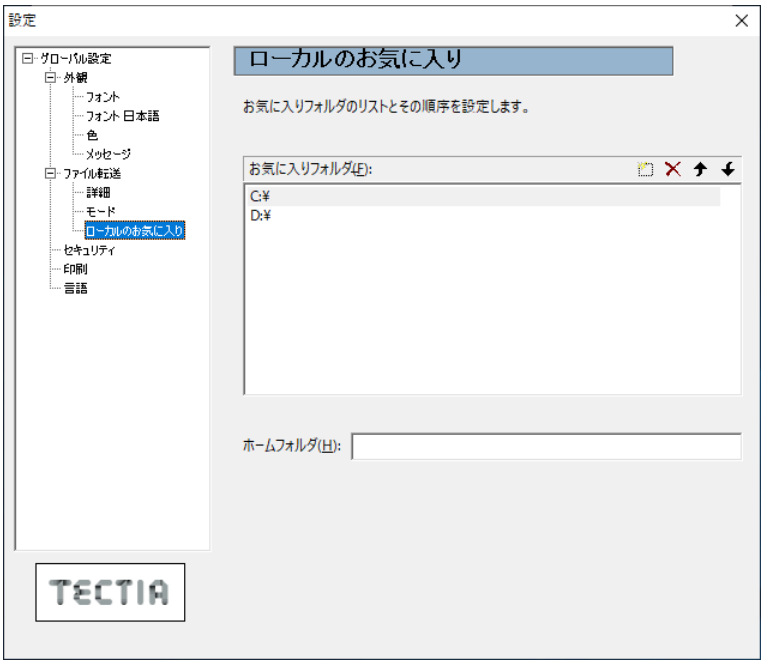
リストからファイル拡張子のエントリを選択し、[**削除**] アイコン ([**アスキー 拡張子**] リストの右上) をクリックすると、拡張子が削除されます。[削除] アイコンのキーボード・ショートカットは `Delete` キーです。

B.1.8. ローカルのお気に入りの定義

[**設定**] ダイアログの [**ローカルのお気に入り**] ページでは、ローカル・コンピュータでよく使われるディレクトリのリストを作成できます。[ファイル転送] ウィンドウのドロップダウン・メニューからこれらのお気に入りを簡単に選択できます。

注意

ここで行うファイル転送の設定は、Tectia のファイル転送 GUI にのみ影響し、SFTP 変換や、ファイル転送に関連するコマンドライン・ツールには影響しません。



図B.9 よく使うディレクトリのリストの作成

お気に入りフォルダ

このリストには、ローカル・コンピュータに定義されているお気に入りフォルダが含まれています。最初は、ローカルで使用可能なドライブがリストに含まれています。リストの上に表示される [**新規作成**]、[**削除**]、[**上**]、及び [**下**] アイコンを使って、お気に入りの追加、削除、及び並べ替えができます。

ホームフォルダ

[**ホームフォルダ**] フィールドには、ファイル転送ウィンドウのローカル・ビュー・ペインに最初に表示されるディレクトリを入力できます。

B.1.9. セキュリティ設定の定義

セキュリティ設定は [設定] ダイアログの [**セキュリティ**] ページで行えます。

図B.10 [設定] ダイアログの [セキュリティ] ページ

ターミナル接続

以下のオプションがあります。

終了時にクリップボードの内容を消去

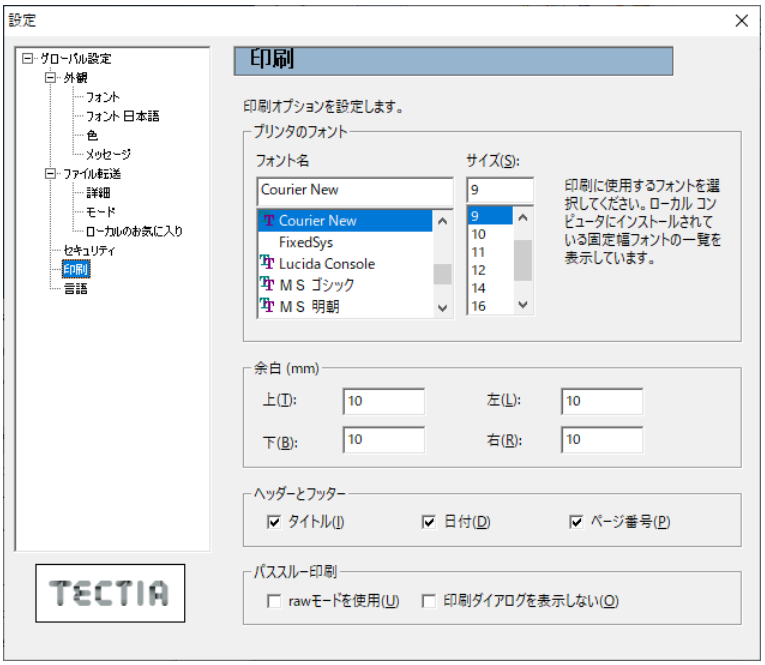
このチェックボックスを選択すると、切り取りと貼り付けの操作で最近コピーされたものがクリップボードから削除されます。

セッション終了時にバッファの内容を消去

このチェックボックスを選択すると、ターミナル出力に残っているすべての内容がスクロールバック・バッファから消去されます。

B.1.10. 印刷

印刷設定は [設定] ダイアログの [**印刷**] ページで行えます。



図B.11 [設定] ダイアログの [印刷] ページ

プリンタのフォント

印刷出力で使用する [**フォント名**] と [**サイズ**] を選択します。システムにインストールされている等幅フォントを選択できます。

余白 (mm)

印刷出力のページ周囲の余白の幅を選択します。ページの上下左右の余白をすべて個別に指定できます。すべての余白のデフォルト値は 10 ミリメートル (または 1 センチメートル) です。

ヘッダーとフッター

印刷ページに表示する追加情報を選択します。

タイトル はページの左上に表示され、ターミナル・ウィンドウのタイトルが表示されます (例: `remotehost - Tectia Client`)。

日付 はページの右上に表示され、そのページが印刷された日付と時刻が表示されます (例: `15 September 2003, 11:10`)。日付と時刻の形式は、Windows で使用されている形式と同じです。

ページ番号 ページの右下に表示されます (例: `Page 1 of 2`)。

パススルー印刷

パススルー印刷では、サーバがターミナル・エミュレーション・コードを使ってクライアントのプリンタで印刷できます。

raw モードでは、Tectia Client は印刷するデータをプレーンテキストとしてプリンタに送信します。このモードでは線のグラフィックなどの印刷はできません。

raw モードでない場合、Tectia Client は印刷するデータを画像としてプリンタに送信します。これはデフォルトの設定であり、印刷に問題がない場合に使用されます。ただし、一部の古いプリンタでは画像の印刷に対応していない場合があります。

rawモードを使用

印刷するデータを raw モードでプリンタに渡すには、このチェックボックスを選択します。印刷に問題がある場合は、このチェックボックスを適宜選択または選択解除してください。

B.2. コマンドライン・オプションの使用

目的によっては、コマンドライン (コマンド・プロンプト) から Tectia SSHターミナル GUI を操作することが便利な場合があります。

Tectia SSHターミナル GUI (`ssh-client-g3.exe`) のコマンドライン構文は以下の通りです。

```
ssh-client-g3 [-r] [-p port] [-u user] [-h host] [profile]
```

コマンドラインのパラメータには以下の意味があります。

`-r`

`-r` オプションは、ユーザ・インターフェイス (ツールバー及びメニュー) に対して行ったカスタマイズをすべてリセットします。確認ダイアログが表示されます。

`-p [port_number]`

`-p` オプションは、接続に使用するポート番号を指定します。このオプションが指定されていない場合、デフォルトのプロファイルで定義されているポート番号が使用されます。

`-u [user_name]`

`-u` オプションは接続で使用するユーザ名を指定します。このオプションが指定されていない場合、デフォルトのプロファイルで定義されているユーザ名が使用されます。

`-h [host_name]`

`-h` オプションは接続で使用するホスト名を指定します。このオプションが指定されていない場合、デフォルトのプロファイルで定義されているホスト名が使用されます。

`[profile]`

プロファイルを指定する場合、そのプロファイルはコマンドラインの最後のオプションである必要があります。コマンドラインのパラメータはプロファイル設定を上書きしま

す。指定されているプロファイルがない場合、デフォルトのプロファイルが使用されます。

`-f`

`-f` (または `/f`) オプションはデフォルトの SFTP ファイル転送プロファイルを開始します。

たとえば、以下のコマンドを実行すると、すぐに `remotehost` というホストへの接続が開始され、`guest` として接続されます。ポート番号が指定されていないので、デフォルトのプロファイルで指定されているポートが接続に使用されます。

```
ssh-client-g3 -h remotehost -u guest
```

以下のコマンドを実行すると、プロファイル・ファイル `custom.ssh2` で定義されている設定を使用して、`remotehost` への接続がすぐに開始されます。

```
ssh-client-g3 -h remotehost custom.ssh2
```

ホストが (`-h` オプションを使用して) 指定されておらず、プロファイルが指定されていない場合、コマンドラインで指定された値が自動的に入力されたログイン・ダイアログが開きます。

たとえば、以下のコマンドを実行すると、ポート番号には指定された 222 を使用し、ユーザ名には `guest` を使用してログイン・ダイアログが表示されます。

```
ssh-client-g3 -u guest -p 222
```

コマンドライン・クライアントの `sshg3.exe` も Windows 版 Tectia Client に含まれています。これは、特にスクリプトを作成する際に便利です。`sshg3.exe` の構文については、sshg3(1) を参照してください。

その他にも、いくつかのコマンドライン・ユーティリティが Tectia Client のインストールに含まれています。詳細については、付録C を参照してください。

B.3. ユーザ・インターフェイスのカスタマイズ

本項では、グラフィカル・ユーザ・インターフェイスを変更するためのオプションについて説明します。

B.3.1. 設定の保存

ユーザ・インターフェイスの設定を変更すると、Tectia GUI のタイトル・バーの現在の設定ファイル名の後ににアスタリスク (`*`) が表示されます (例: `default*`)。これは、変更した設定がまだ恒久的なものではないこと、つまりまだ保存されていないことを示しています。

変更を恒久的なものにし、後で使用するために保存しておくことができます。ツールバーの [**保存**] ボタンをクリックするか、または [**ファイル**] → [**設定の保存**] を選択して、現在の設定に加えた変更を保存します。

現在開いているターミナル・ウィンドウとファイル転送ウィンドウの位置は、[**ファイル**] → [**レイアウトの保存**] オプションでそれぞれ保存できます。ウィンドウの位置を整えて、レイアウト設定をデフォルト設定ファイルに保存しておくと、次回 Tectia SSHターミナル GUI を起動したときに、自動的に好みの位置にウィンドウが配置されるようになります。

デフォルトでは、すべてのウィンドウが一度に開きます。[**設定**] ダイアログの [**外観**] ページでこの設定を変更しておくと、新しいターミナル・ウィンドウやファイル転送ウィンドウを開くときに、定義したウィンドウが必要なときだけ開くようになります。B.1.1 を参照してください。

設定の指定に手間がかかる場合は、変更した設定ファイル (`ssh-broker-config.xml`、`global.dat`、及び `*.ssh2`) のバックアップ・コピーを作成し、安全な場所に保存しておくことをお勧めします。そのようにしておけば、(ハードウェアの故障などで) 設定ファイルを失ってしまっても、個人設定を再度作成する必要はありません。

複数の設定ファイル

設定ファイルは、リモート・ホスト・コンピュータごとに別々に保存できます。これを行うには、[**プロファイル**] を使用します。プロファイルの詳細な使用方法については、A.1.3 を参照してください。

B.3.2. 設定の読み込み

過去に保存したプロファイルは簡単に使用できます。[プロファイル] ツールバーの [**プロファイル**] オプションを選択するか、または [**ファイル**] → [**プロファイル**] を選択すると、過去に保存したプロファイルのメニューが表示されます。プロファイル名をクリックすると、そのプロファイルの設定を使用した接続がすぐに開始されます。

この操作は、すでにリモート・ホスト・コンピュータに接続している場合にも有効です。プロファイル名をクリックすると、新しい別の接続が開始されます。

特定の接続の設定を読み込む別の方法には、Windows エクスプローラなどで設定ファイル名をダブルクリックする方法があります。Tectia Client をインストールすると、拡張子が `.ssh2` のファイルが Tectia のソフトウェアに関連付けられます。つまり、設定ファイルをダブルクリックすることで、任意の設定ファイルが読み込まれた状態で Tectia Client を起動できます。

複数のリモート・ホスト・コンピュータに定期的に接続する場合は、Windows のデスクトップなどに、対応する設定ファイルへのショートカットを作成しておけます。これにより、デスクトップ上のアイコンをクリックするだけで、関連する設定がすでに定義された状態で、目的の接続を素早く開くことができます。

B.4. セッションの記録

Tectia Client はセッションのターミナル出力をテキスト・ファイルに記録できます。この機能は **Tectia SSHターミナル GUI** から有効にできます。

テキスト・ファイルへのセッションの記録を開始するには、[**ファイル**] → [**セッションの記録**] を選択します。開いた [**名前を付けて保存**] ダイアログボックスで、セッションを記録するファイルを定義します。新しいファイル名を入力することも、既存のファイルを選択することもできます。

既存のファイルにセッションを記録することを選択した場合は、[**ファイルが既に存在します**] というメッセージが表示されます。既存のファイルを上書きする場合は、[**はい**] をクリックします。セッションの記録を既存のファイルの最後に追加するには、[**いいえ**] をクリックします。

付録C コマンドライン・ツールと man ページ

Tectia Client には、いくつかのコマンドライン・ツールが同梱されています。それらのツールの機能について、以下の付録で簡単に説明します。

Unix の場合、以下のマニュアル・ページでも同様の情報を確認できます。

- ssh-broker-g3(1): 接続ブローカー – Generation 3
- ssh-broker-ctl(1): 接続ブローカー・コントロール・ユーティリティ
- ssh-troubleshoot(8): トラブルシューティングを目的としてシステム情報を収集するためのユーティリティ
- sshg3(1): Secure Shell ターミナル・クライアント – Generation 3
- scpg3(1): Secure Shell ファイル・コピー・クライアント – Generation 3
- sftpg3(1): Secure Shell ファイル転送クライアント – Generation 3
- ssh-translation-table(1): Secure Shell ファイル転送変換テーブル
- ssh-keygen-g3(1): 認証鍵ペア・ジェネレータ
- ssh-keyfetch(1): サーバ・ホスト鍵をダウンロードするためのユーティリティ
- ssh-cmpclient-g3(1): 証明書 CMP 登録クライアント
- ssh-scepclient-g3(1): 証明書 SCEP 登録クライアント
- ssh-certview-g3(1): 証明書ビューア

- ssh-ekview-g3(1): 外部鍵ビューア

Windows 上の Tectia SSHターミナル GUI のコマンドライン・オプション (`ssh-client-g3.exe`) については、B.2 を参照してください。

接続ブローカーの設定ファイルのオプションについては、ssh-broker-config(5) を参照してください。

ssh-broker-g3

ssh-broker-g3 — Tectia 接続ブローカー - Generation 3

書式

`ssh-broker-g3 [ -a, --broker-address= ADDR ] [ -f, --config-file= FILE ] [ -D, --debug= LEVEL ] [ -l, --debug-log-file= FILE ] [ --pid-file= FILE ] [ --exit ] [ --reconfig ] [ -h ] [ -V ]`

説明

ssh-broker-g3 (Windows では **ssh-broker-g3.exe**) は Tectia Client 及び Tectia ConnectSecure のコンポーネントです。このコマンドは Tectia Client 及びクライアント・プログラム **sshg3**、**scpg3**、**sftpg3**、及び **ssh-client-g3.exe** (Windows の場合のみ) のすべての暗号化操作と認証関連のタスクを処理します。

ssh-broker-g3 は Secure Shell バージョン 2 プロトコルを使用して Secure Shell サーバと通信します。

ssh-broker-g3 コマンドを使用することで、手動で接続ブローカーを起動できます。これによって **ssh-broker-g3** がバックグラウンドで起動し、以後 **sshg3**、**sftpg3**、または **scpg3** を使用する場合は、新しいブローカー・セッションを開始するのではなく、この接続ブローカーのインスタンスを介して接続します。

接続ブローカーがバックグラウンドで実行されていないときにコマンドライン・クライアント (**sshg3**、**sftpg3**、または **scpg3**) を起動した場合、クライアントはブローカーをランオンデマンド・モードで起動します。このモードでは、最後のクライアントが切断された後に **ssh-broker-g3** は終了します。

ランオンデマンド・モードで実行されている **ssh-broker-g3** プロセスがあり、コマンドラインから接続ブローカーを起動した場合、新しい **ssh-broker-g3** プロセスは古い **ssh-broker-g3** プロセスにメッセージを送信してランオンデマンド・モードからバックグラウンド・モードに変更し、クライアントが切断した後もブローカーが実行されたままにします。

実行中の接続ブローカーのステータスは **ssh-broker-ctl** 及び **ssh-broker-gui** ユーティリティを使用して確認できます。

認証

接続ブローカーは自動的に認証エージェントとして動作し、ユーザの公開鍵を保存し、Secure Shell 接続で認証を転送します。鍵ペアは `ssh-keygen-g3` で作成できます。

接続ブローカーは OpenSSH クライアントの認証エージェントとしても機能します。

ユーザ認証に使用する公開鍵ペアは、デフォルトで `$HOME/.ssh2` ディレクトリ (Windows では `%APPDATA%\SSH\UserKeys`) に保存されます。詳細については、「ファイル」を参照してください。

接続ブローカーは、Secure Shell サーバの認証に使用されるホスト公開鍵を含むデータベースを自動的に維持及び確認します。サーバ・ホストに初めてログインするとき、ホストの公開鍵はユーザの `$HOME/.ssh2/hostkeys` ディレクトリ (Windows では `%APPDATA%\SSH\HostKeys`) に保存されます。詳細については、「ファイル」を参照してください。

オプション

ssh-broker-g3 の最も重要なオプションは以下の通りです。

`-a, --broker-address= ADDR`

ローカル・アドレス `ADDR` の接続ブローカー接続をリッスンします。

`-D, --debug= LEVEL`

デバッグ・レベル文字列を `LEVEL` に設定します。

`-f, --config-file= FILE`

接続ブローカーの設定ファイルをデフォルトの場所ではなく、`FILE` から読み込みます。

`-l, --debug-log-file= FILE`

デバッグ・メッセージを `FILE` にダンプします。

`--pid-file= FILE`

接続ブローカーのプロセス ID を `FILE` に保存します。

`--exit`

現在実行中の接続ブローカーを終了させます。これにより、すべての接続が終了します。

`--reconfig`

設定ファイル (`ssh-broker-config.xml`) を再読み込みして使用します。

`-V, --version`

プログラムのバージョンを表示して終了します。

`-h, --help`

コマンドライン・オプションの要約を表示して終了します。

環境変数

以下のオプションの環境変数は、特定の状況で必要とされます。

SSH_SECSH_BROKER `=ADDRESS`

この変数は、接続先となる個別の Tectia 接続ブローカープロセスへのアドレスを定義します。

この変数は、接続ブローカーのプロセスをデフォルト以外の場所から実行する場合や、**ssh-broker-g3** プロセスの所有者以外の userID を使用する場合に、そのプロセスの場所を定義するために必要になります。

ファイル

ssh-broker-g3 は以下のファイルを使用します。

`$HOME/.ssh2/ssh-broker-config.xml`

これは、 **ssh-broker-g3** (及び **sshg3**、 **scpg3**、 **sftpg3**) で使用されるユーザ固有の設定ファイルです。このファイルのフォーマットについては、 ssh-broker-config(5) で説明しています。 このファイルには通常、機密情報は含まれませんが、推奨されるパーミッションは、ユーザには読み込み/書き込みを許可し、それ以外はアクセスできないようにすることです。

Windows では、ユーザ固有の設定ファイルは `%APPDATA%\SSH\ssh-broker-config.xml` にあります。

`$HOME/.ssh2/random_seed`

このファイルは、乱数発生器にシード値を与えるために使用されます。このファイルは機密データを含むので、パーミッションは、ユーザには読み込み/書き込みを許可し、それ以外はアクセスできないようにする必要があります。このファイルはプログラムの初回実行時に作成され、自動的に更新されます。ここのファイルを読み込んだり、修正したりする必要はありません。

Windows では、乱数シード・ファイルは `%APPDATA%\SSH\random_seed` にあります。

`$HOME/.ssh2/identification`

このファイルには、リモート・ホストに接続する際のユーザ認証に使用される公開鍵と証明書に関する情報が含まれています。

Tectia Client G3 では、すべてのユーザ鍵をデフォルト・ディレクトリに保存し、すべてのユーザ鍵を公開鍵認証及び/または証明書認証に使用することを許可すれば、`identification` ファイルを使用する必要はありません。`identification` ファイルが存在しない場合、接続ブローカーは `$HOME/.ssh2` ディレクトリで見つかった各鍵を使用しようとします。`identification` ファイルが存在する場合、そのファイルに記載されている鍵が最初に試されます。

identification ファイルには秘密鍵のファイル名の一覧が含まれ、それぞれのファイル名の前にキーワード `IdKey` (または `CertKey`) が付いています。以下にファイルの例を示します。

```
IdKey       mykey
```

これは、公開鍵認証を使用してログインしようとするときに、`$HOME/.ssh2/mykey` を使用するように接続ブローカーに指示します。

このファイルは、デフォルトでは `$HOME/.ssh2` ディレクトリにあるとみなされますが、鍵ファイルのパスを指定することもできます。パスは、絶対パスまたは `$HOME/.ssh2` ディレクトリの相対パスで指定します。 `IdKey` が複数ある場合は、identification ファイルに記載されている順番に試行されます。

Windows では、identification ファイルは `%APPDATA%\SSH\identification` にあります。ファイル内の鍵のパスは、絶対パスまたは `%APPDATA%\SSH` ディレクトリの相対パスで指定でき、例えば `C:\%username-without-domain%\private_keys\mykey` のように、サーバ側の認証ファイルでサポートされているものと同じパターン文字列を 含むことができます。 デフォルトのユーザ鍵ディレクトリは `%APPDATA%\SSH\UserKeys` で、デフォルトのユーザ証明書ディレクトリは `%APPDATA%\SSH\UserCertificates` です。

`$HOME/.ssh2/hostkeys`

これは、サーバ・ホストの公開鍵を保存するための、ユーザ固有のデフォルトのディレクトリです。`ssh-broker-config.xml` ファイルで `strict-host-key-checking` が `yes` に設定されていない限り、サーバに接続すると、新しい鍵を、または変更された鍵を自動的に受け入れるかどうかを求められます。新しい鍵または変更された鍵を受け入れる前に、鍵のフィンガープリントを検証する必要があります。

リモート・ホストとの最初の接続時にホスト鍵を受け取り (またはホスト鍵が変更されており)、その鍵を保存することを選択すると、そのファイル名はデフォルトでハッシュ化されたフォーマットで保存されます。ハッシュ化されたホスト鍵フォーマットは、ホスト上でのアドレス・ハーベスティングを困難にするためのセキュリティ機能です。

保存フォーマットは、設定ファイル `ssh-broker-config.xml` の `known-hosts` エレメントの `filename-format` 属性で制御できます。属性値は `plain` または `hash` (デフォルト) でなければなりません。

鍵を手動で追加する場合は、鍵の名前は `key_<port>_<host>.pub` のパターンにする必要があります。この場合の `<port>` は、Secure Shell サーバが動作しているポート、`<host>` はサーバへの接続時に使用するホスト名 (例: `key_22_alpha.example.com.pub`) です。

ハッシュ化されたフォーマットとプレーンテキスト・フォーマットの鍵が両方とも存在する場合、ハッシュ化されたフォーマットが優先されます。

クライアントが接続しているホストとポートによって、識別が異なることに注意してください。たとえば、短いホスト名 `alpha` は、完全修飾ドメイン名 `alpha.example.com` とは異なるとみなされます。また、IP を使った接続 (例: `10.1.54.1`)も、同じホストの異なるポートに接続するのと同様に、別のホストとみなされます。

Windows では、ユーザ固有のホスト鍵ファイルは `%APPDATA%\SSH\HostKeys` にあります。

ホスト鍵の詳細については、4.2 を参照してください。

`$HOME/.ssh2/hostkeys/salt`

これは、ハッシュ化されたホスト鍵名用の初期化ファイルです。

Windows では、salt ファイルは `%APPDATA%\SSH\HostKeys\salt` にあります。

`/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`

これは、 **ssh-broker-g3** (及び **sshg3**、 **scpg3**、 **sftpg3**) で使用される設定ファイルで、工場出荷時の設定が含まれています。このファイルは編集せず、デフォルト設定を確認するためにのみ使用して下さい。このファイルのフォーマットについては、ssh-broker-config(5) で説明しています。

Windows では、デフォルトの設定ファイルは `<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml` にあります。

`/etc/ssh2/ssh-broker-config.xml`

これは、 **ssh-broker-g3** (及び **sshg3**、 **scpg3**、 **sftpg3**) で使用されるグローバル (システム全体の) 設定ファイルです。このファイルのフォーマットについては、ssh-broker-config(5) で説明しています。

Windows では、グローバル設定ファイルは `<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml` にあります。

`/etc/ssh2/hostkeys`

ホスト鍵がユーザ固有の `$HOME/.ssh2/hostkeys` ディレクトリに見つからない場合、すべてのユーザでこのディレクトリが次に確認されます。ホスト鍵ファイルは自動的にここに置かれるわけではなく、システム管理者 (`root`) が手動で更新する必要があります。

管理者が各ホストに接続してホスト鍵を取得する場合、鍵はデフォルトでハッシュ化されたフォーマットになります。この場合、管理者の `$HOME/.ssh2/hostkeys/salt` ファイルも `/etc/ssh2/hostkeys` ディレクトリにコピーする必要があります。

Windows では、システム全体のホスト鍵ファイルはデフォルトで以下の場所にあります。

`"C:\ProgramData\SSH\HostKeys"`

`/etc/ssh2/hostkeys/salt`

これは、ハッシュ化されたホスト鍵名用の初期化ファイルです。このファイルは、ホスト鍵を取得したのと同じ管理者が手動でここにコピーする必要があります。

Windows では、すべてのユーザの salt ファイルはデフォルトで以下の場所にあります。

`"C:\ProgramData\SSH\HostKeys\salt"`

`/etc/ssh/ssh_known_hosts`

これは、OpenSSH クライアントが既知のサーバ・ホストの公開鍵データを保存するために使用する、デフォルトのシステム全体のファイルです。このファイルは Tectia Client でもサポートされています。

ホスト鍵がユーザ固有の `$HOME/.ssh/known_hosts` ファイルに見つからない場合、すべてのユーザでこのディレクトリが次に確認されます。

Tectia Client や ConnectSecure では、新しいホスト鍵は常に Tectia ユーザ固有のディレクトリ `$HOME/.ssh2/hostkeys` に保存されるので、 `ssh_known_hosts` ファイルが自動的に更新されることはありません。

`$HOME/.ssh/known_hosts`

これは、OpenSSH クライアントが既知のサーバ・ホストの公開鍵データを保存するために使用する、デフォルトのユーザ固有のファイルです。`known_hosts` ファイルは Tectia Client でもサポートされています。

`known_hosts` ファイルには、それぞれの既知のホスト鍵のハッシュ化されたフォーマットまたはプレーンテキスト・フォーマットのエントリと、サーバで使用されているポート (標準の 22 でない場合にのみ) が含まれています。 `known_hosts` ファイルのフォーマットの詳細については、OpenSSH sshd(8) man ページを参照してください。

Tectia Client や ConnectSecure では、新しいホスト鍵は常に Tectia のディレクトリ `$HOME/.ssh2/hostkeys` に保存されるので、 `known_hosts` ファイルが自動的に更新されることはありません。

`$HOME/.ssh2/authorized_keys` (サーバ・ホスト上)

このディレクトリは、ログインに使用することが許可されたユーザ公開鍵のために Tectia Server が使用するデフォルトの場所です。

Windows 上のTectia Server では、ユーザ公開鍵のデフォルトのディレクトリは `%USERPROFILE%\.ssh2\authorized_keys` です。

`$HOME/.ssh2/authorization` (サーバ・ホスト上)

これは、以前のバージョンの Tectia Server (**sshd2**) で使用されていたデフォルトのファイルで、ログインに使用することが許可されたユーザ公開鍵が記載されています。このファイルは Tectia Server G3 (**ssh-server-g3**) でもオプションとしてで使用できます。

Windows 上の Tectia Server では、authorization ファイルはデフォルトで `%USERPROFILE%\.ssh2\authorization` にあります。

このファイルのフォーマットの詳細については、ssh-server-g3(8) man ページを参照してください。

`$HOME/.ssh/authorized_keys` (サーバ・ホスト上)

これは、OpenSSH サーバ (**sshd**) が使用するデフォルトのファイルで、ログインに使用することが許可されたユーザの公開鍵が含まれています。

このファイルのフォーマットの詳細については、OpenSSHsshd(8) man ページを参照してください。

ssh-broker-ctl

ssh-broker-ctl — Tectia 接続ブローカーコントロール・ユーティリティ

書式

`ssh-broker-ctl command`
[`options` ...]

説明

ssh-broker-ctl (Windows では **ssh-broker-ctl.exe**) は接続ブローカー(**ssh-broker-g3**) のコントロール・ユーティリティです。たとえば、接続ブローカーのステータスの表示、接続ブローカーの再設定や停止、鍵と証明書の管理、接続の管理などに使用できます。

オプション

以下の一般オプションがあります。

`-a, --broker-address ADDRESS`

接続先となる個別の Tectia 接続ブローカー・プロセスへのアドレスを定義します。

環境変数 `SSH_SECSH_BROKER` で接続ブローカー・アドレスを定義しても同じ効果が得られます。

ヒント

ssh-broker-g3 プロセスの所有者以外の userID を使用して **ssh-broker-ctl** を実行している場合は、 `-a` オプションを指定して **ssh-broker-ctl** が接続先を認識できるようにする必要があります。この場合、 **ssh-broker-ctl** は特権ユーザ (root) で実行する必要もあります。

たとえば、ユーザ `SSHBRKR` が **ssh-broker-g3** プロセスを所有している場合、 **ssh-broker-ctl** を以下のコマンドで実行します。

```
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status -s
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status --pid
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker list-connections
```

`-D, --debug STR`

デバッグ・レベルを定義します。

`-e, --charset= CS`

出力に使用する文字コードを定義します。サポートされている文字コードは `utf8`、`iso-8895-1`、 `latin1`、 `iso-8859-15`、 `latin9`、 及び `ascii` です。

`-q, --quiet`

コマンドによって、出力をほとんど表示しないか、全く表示しないかを定義します。

`-s, --short`

より短く、機械で読み取りやすい出力フォーマットを使用することを定義します。

`--time-format= FMT`

出力に使用する時刻表示形式を定義します。デフォルトはシステム・ロケールの設定に依存します。

`-v, --verbose`

より多くの情報がある場合、それを出力することを定義します。

`-V, --version`

バージョン文字列を表示します。

`-w, --wide`

出力が長い行になっても、切り詰めないことを定義します。

`-h, --help`

コマンドライン・オプションに関するコンテキストに合わせたヘルプ・テキストを表示します。特定のコマンドに関するヘルプも用意されています。たとえば、`status` コマンドのヘルプを表示させるには、次のコマンドを実行します。

```
$ ssh-broker-ctl status --help
```

コマンド

注意

コマンド・オプションの詳細な説明については、コマンド固有の `--help` オプションを使用して確認してください。

ssh-broker-ctl は以下のコマンドを受け付けます。

`add-certificate [ options ] <certificate-file>`

指定された X.509 サブ CA 証明書を接続ブローカーの証明書キャッシュに追加します。この証明書は証明書の検証で使用できますが、恒久的には保存されません。接続ブローカーを再起動すると、証明書は削除されます。

`add-crl [ options ] <crl-file>`

指定された X.509 CRL を接続ブローカーの CRL キャッシュに追加します。この CRL は証明書の検証で使用できますが、恒久的には保存されません。接続ブローカーを再起動すると、CRL は削除されます。

`add-key filename`

指定されたファイル名から新しい秘密鍵を追加します。この秘密鍵は、設定に恒久的には保存されません。接続ブローカーを停止すると、鍵は削除されます。

`add-provider type parameter`

鍵プロバイダを接続ブローカーに登録します。 `type` オプションはサポートされているプロバイダ・タイプの 1 つであり、 `parameter` オプションはプロバイダ・タイプに固有のパラメータ文字列です。

サポートされている鍵プロバイダのタイプと対応するパラメータ形式のリストについては、コマンド固有の `--help` オプションを使用して確認してください。

`auth-handler [ options ]`

自身をデフォルトの認証フォーム・ハンドラとして登録します。認証プロンプトを処理できないクライアントに対する認証プロンプトはすべて (主に SOCKS プロキシやその他のトンネル)、このクライアントに向けられます。

サポートされている鍵プロバイダのタイプと対応するパラメータ形式のリストについては、コマンド固有の `--help` オプションを使用して確認してください。

`close-channel channel-id ...`

指定されたチャネルを閉じます。複数のチャネル ID を入力して、複数のチャネルを閉じることもできます。

`close-connection connection-id ...`

指定された接続を閉じます。複数の接続 ID を入力して、複数の接続を閉じることもできます。

`close-tunnel-listener tunnel-id ...`

開いているトンネル・リスナを閉じます。トンネル ID は、 **ssh-broker-ctl list-tunnel-listeners** コマンドで返される ID 番号、またはコロンで区切られたリスナ・アドレス及びポートのペアのいずれかです。リスナ・アドレスが省略された場合は、ローカル・リスナ (127.0.0.1) が選択されます。たとえば、以下のコマンドは、ID 7 のリスナ、及び 168.192.0.15 ポート番号 1234 及び 127.0.0.1 ポート番号 2112 でリッスンしているリスナを閉じます。

```
$ ssh-broker-ctl ctl 7 168.192.0.15:1234 :2112
```

`config-value [ options ] path`

指定されたパスに基づいて接続ブローカーから設定値を取得し、その結果を表示します。

`connection-status [ --show-channels ] [ --write-hostkey= FILE ] connection-id`

接続 ID (**list-connections** コマンドで表示される数値の識別子) の詳細な接続ステータスを表示します。

`connector [ options ] [ enable|disable ]`

接続ブローカーの Connector 機能を有効または無効にします。パラメータを指定しない場合は、現在の状態を表示します。

`disconnect-client client-id`

接続ブローカー・クライアント・プロセスを切断します。

`debug [ --append ] [ --clear ] [ --log-file= file ] [ --monitor ] [ --protocol-dump ] [ debug-level ]`

接続ブローカーのデバッグ・レベルを、指定されたレベルに設定します。ここで `debug-level` パラメータが指定されない場合、現在のデバッグ・レベルは変更されません。

`generate-key [ options ] key-name`

接続ブローカー内の鍵プロバイダを使用して、秘密鍵を生成します。デフォルトでは、秘密鍵はソフトウェア鍵として、ユーザのホーム・ディレクトリのファイルに保存されます。鍵プロバイダは、秘密鍵の保存に他の方法を提供できます。

`keylog [ --remove ] [ --all ] [ --update <key-id|key-hash> ] [ --init ] [ --uninit ] [ --close ] [ -v, --verbose ] [ key-id|key-hash|hostname ]`

キーログは、アップロードされた公開鍵を管理し、そのログを表示するために使用します。キーログは公開鍵を保存するのではなく、鍵と鍵がアップロードされたホストに関する情報を保存するだけです。この情報は、鍵のアップロード先のホストを追跡するなど、後々の鍵の管理に利用できます。デフォルトでは、キーログはオンになっていないため、先に有効にする必要があります。

オプションを指定しない場合は、アップロードされた鍵のリストを表示します。鍵またはホスト名を指定した場合は、選択された鍵のみが表示されます。

`key-passphrase [ --all ] [ --clear ] [ --passphrase-file= filename ] [ --passphrase-string= passphrase ] [ key-id|key-hash ]`

ユーザの秘密鍵のパスフレーズまたは PIN コードの入力を求めます。

`key-upload [ options ] [ --replace-key ] [ --scan-key ] [ --delete-key ] key [user@]server [ #port ]`

選択した鍵 (`key` は鍵 ID 番号、公開鍵ハッシュまたはファイル名) を、サーバ上の認可された鍵ディレクトリまたはファイルに、自動的に検出されたアップロード方法でアップロードします。操作後、この鍵はパスワードなしでサーバにログインするために公開鍵認証で使用できます。キーログが有効な場合、コマンドはキーログのパスフレーズの入力を求め(必要な場合)、公開鍵に関する情報が鍵のアップロード・ログに保存されます。

オプション `--replace-key` は、選択された鍵を通常の鍵ローテーション・ルールに従ってローテートします。オプション `--scan-key` は、選択されたホストの鍵をスキャンするために使用します。オプション `--delete-key` は、選択された認証鍵を削除するために使用します。

`list-connections [ -c, --show-channels ] [ -s, --short ] [ --client-pid= PID ] [ --disconnected ]`

現在開いている接続の一覧を接続パラメータ及びトラフィック統計データとともに表示します。他のコマンドで接続を識別するために使用できる接続 ID も表示します。

`list-channels [ -s, --short ]`

現在開いている接続チャネルの一覧を、チャネル・タイプとトラフィック統計データとともに表示します。他のコマンドで接続を識別するために使用できる接続 ID も表示します。

`list-clients [ -c, --show-channels ] [ -s, --short ] [ --all ]`

現在接続しているクライアント・プロセスの一覧を表示します。

`list-keys [ -s, --short ] [ --extra certificates ] [ --provider= ID ]`

ユーザの秘密鍵の一覧を、鍵のタイプやサイズ、ファイル名、鍵プロバイダ情報などの基本的な鍵の属性とともに表示します。鍵のフィンガープリントと識別子も出力します。この識別子は、接続ブローカーのその他のコマンドで秘密鍵を識別するために使用されます。

`list-profiles [ -s, --short ] [ -v, --verbose ] [ name ... ]`

接続ブローカーの接続プロファイルの一覧を表示します。プロファイル名と、ホスト名やユーザ名などの基本的な接続設定を表示します。プロファイル名が指定されている場合、それに該当するプロファイルのみが表示されます。

`list-providers [ provider ... ]`

接続ブローカーの鍵プロバイダの一覧を表示します。プロバイダ名または ID 番号が 1 つ以上指定されている場合、それに該当するプロバイダのみが表示されます。プロバイダ名には、完全なプロバイダ名またはプレフィックスを指定できます。

`list-tunnel-listeners [ options ]`

現在アクティブなトンネル・リスナ (ポート転送とも呼ばれます) の一覧を表示します。

`open-tunnel-listener [ options ] listen-port [user@]server [#port] [ dst-host ] [ dst-port ]`

sshg3 `-L` や `-R` オプションと同様に、トンネル・リスナを開きます。異なるのは、**ssh-broker-ctl** はトンネルを開いた後に終了する点です。トンネルのステータスは **ssh-broker-ctl list-tunnel-listeners** コマンドで表示でき、 **ssh-broker-ctl close-tunnel-listener** コマンドでトンネルを閉じることができます。

ローカル・モード (デフォルト) では、リスナは localhost のリスナ・ポートに開かれます。すべての接続はサーバを経由して、そこから最終的な接続先のアドレスとポートにトンネルされます。トンネルの種類の `socks` と `socks-proxy` では接続先の情報は SOCKS クライアントから取得されるので、接続先の情報は必要ありません。トンネルの種類の `tcp`、 `ftp`、 及び `local` は接続先の情報を必要とします。

`performance [ options ] [ show ] [ clear ] [ show-and-clear ] [ show-total ] [ interval <time|index|'all'> ]`

パフォーマンス・プロファイリング・データの表示と操作を行います。

`pkcs10-sign [ options ] key-id [ subject-name ]`

指定された鍵で PKCS#10 証明書要求に署名します。 `key-id` には鍵 ID または鍵ハッシュのいずれかを指定できます。 `template` オプションを使用しない限り、サブジェクト名のパラメータは必須です。サブジェクト名が有効な識別名でない場合、一般名コンポーネントに自動的に変換されます。たとえば、 `My Name` というサブジェクト名の文字列は `CN=My Name` に変換されます。

`probe-key [ options ] address#port`

Secure Shell サーバ・ホスト鍵を探します。指定されたアドレスとポート (デフォルトでは 22) に接続し、サーバの公開鍵または証明書を表示します。

`reload`

接続ブローカーの設定ファイルを再読み込みします。

`remove-key [ options ] key-id`

秘密鍵を恒久的に削除します。

`remove-provider [ --all ] provider-id`

鍵プロバイダを接続ブローカーから削除します。

`start`

接続ブローカーがまだ実行されていなければ、デーモン・モードで起動します。

`start-gui`

接続ブローカーの GUI プロセスがまだ実行されていなければ、起動します。

`status [ -s, --short ] [ -q, --quiet ] [ --pid ] [ --all ]`

パラメータを指定しない場合、現在実行されている接続ブローカーのプロセスの簡単な統計データと設定の要約を表示します。

`stop`

接続ブローカーを停止します。

`validate-certificate [ options ] <certificate-file>`

指定された X.509 証明書を検証します。ホスト名が指定されている場合、証明書がそのホストのホスト証明書として受け入れられるかどうかの確認も行います。

`view-key [ -s, --short ] [ -v, --verbose ] [ --clear ] [ --write-key= file ] key-id`

指定された鍵の情報を表示します。鍵に証明書がある場合、その要約も表示します。

ssh-troubleshoot

ssh-troubleshoot — システム情報を収集するためのツール

書式

`ssh-troubleshoot [ options ] [ command [ command-options ] ]`

説明

ssh-troubleshoot (Windows では **ssh-troubleshoot.cmd**) はオペレーティング・システムに関する情報 (そのバージョン、パッチ、構成設定、インストールされているソフトウェア・コンポーネント、現在の環境と状態) と Tectia のインストールに関する情報 (インストールされている製品コンポーネントとバージョン、その状態、グローバル及びユーザ固有の設定) を収集するツールです。

収集された情報は、`ssh_troubleshoot_<host>_<date>_<time>.tar` (Unix の場合) または `ssh_troubleshoot_*.log` (Windows の場合) という名前のファイルに保存されます。トラブルシューティングの際は、解析のために SSH のテクニカル・サポートにこのファイルを送信してください。

必要なすべての情報を得るために、ディレクトリへのルート・アクセスを必要とする場合があるので、管理者としてコマンドを実行してください。

オプション

各オプションは別々に入力してください。組み合わせることはできません。以下のオプションがあります。

`-d, --debug LEVEL`

デバッグ・レベル文字列を `LEVEL` に設定します。

`-k, --keep-going`

エラー発生後も可能な限りデータ収集を継続することを定義します。Windows ではサポートされていません。

`-o, --output FILENAME`

収集したデータを保存するためのデフォルト以外の出力ファイルを定義します。Windows ではサポートされていません。

`FILENAME` が '-' の場合、収集されたデータは標準出力に出力されます。デフォルトの出力ファイルは一時アーカイブ・ディレクトリに作成され、`ssh-troubleshoot-data-<hostname>-<timestamp>.tar` として保存されます。タイムスタンプのフォーマットは `yyyymmdd-hhmmUTC` です。

`-u, --user USERNAME`

info コマンド用に別のユーザを定義します。デフォルトはカレント・ユーザです。このオプションは、Tectia のユーザ固有の設定ファイルを取得するホーム・ディレクトリに影響します。Windows ではサポートされていません。

`-q, --quiet`

コマンドの進捗に関する詳細な報告を抑制し、エラーのみを報告します。

`-h, --help`

このヘルプ・テキストを表示します。

コマンド

ssh-troubleshoot は以下のコマンドを受け付けます。

`info`

システム設定に関する情報を収集します。収集されたデータは、 `tar` ファイル (Unix の場合) または `log` ファイル (Windows の場合) としてに保存されます。

オプション:

`--include-private-keys`

指定されたユーザの設定ディレクトリから、秘密鍵を含むすべてのデータを収集します。デフォルトでは、秘密鍵や認識できないファイルは結果データに含まれません。このオプションは Windows ではサポートされていません。

sshg3

sshg3 — Secure Shell ターミナル・クライアント - Generation 3

書式

`sshg3` [`options` **...**]
`profile` | [`user@`] `host` [`#port`]
[`command`]

説明

sshg3 (Windows では **sshg3.exe**) は、リモート・マシンにログインし、リモート・マシン上でコマンドを実行するためのプログラムです。 **sshg3** は、セキュアでないネットワーク上の 2 つのホスト間で、セキュアで暗号化された通信チャネルを提供します。このコマンドラインは、セキュアでない **rlogin**、 **rsh**、及び **telnet** プログラムを置き換えます。また、X11 接続や任意の TCP/IP ポートも **sshg3** によってセキュアなチャネルで転送できます。

sshg3 を使用してリモート・ホストに接続するには、`ssh-broker-config.xml` ファイルで定義されている接続プロファイルの名前 (`profile`) か、リモート・ホストの IP アドレスまたは DNS 名、及びオプションでリモート・ユーザ名と Secure Shell サーバのポート (`[user@]host[#port]`) を使用します。ユーザ名を指定しない場合は、ローカル・ユーザ名とみなされます。ポートを指定しない場合、デフォルトの Secure Shell ポート 22 が使用されます。リモート・ホストの場合は、Secure Shell バージョン 2 サーバが動作している必要があります。

sshg3 は接続ブローカーのクライアントとして動作し、実際の接続ブローカープロセスである **ssh-broker-g3** をトランスポートとして (ランオンデマンド・モードで) 起動するか、またはすでに動作している接続ブローカープロセスを使用します。認証のためにパスワードまたはパスフレーズが必要な場合、接続ブローカーはユーザに入力を求めます。接続ブローカーは、`ssh-broker-config.xml` ファイルで指定された設定を使用します。

ユーザのIDがサーバに承認されると、サーバは指定されたコマンドを実行するか、マシンにログインしてユーザに通常のシェルを提供します。リモート・コマンドやシェルとの通信はすべて、自動的に暗号化されます。

疑似 tty が割り当てられていない場合、セッションは透過的であり、バイナリ・データをセキュアに転送するために使用できます。

セッションは、リモート・マシン上のコマンドまたはシェルが終了、またはすべての X11 及び TCP/IP 接続が閉じられたときに終了します。リモート・プログラムの終了ステータスが **sshg3** の終了ステータスとして返されます。

エージェント転送 (Unix)

ssh-broker-g3 は認証エージェントとして動作し、`ssh-broker-config.xml` ファイルで、または **sshg3** コマンドラインで (`-a` オプションを使用して) 無効にしない限り、エージェントへの接続は自動的にリモート側へ転送されます。

X11 転送

ユーザが X11 を使用している (環境変数 `DISPLAY` が設定されている) 場合、シェル (またはコマンド) から起動される X11 プログラムは暗号化されたチャネルを経由し、実際の X サーバへの接続はローカル・マシンから行われるように、X11 ディスプレイへの接続を自動的にリモート側に転送できます。ユーザは手動で `DISPLAY` を設定してはなりません。X11 接続の転送は、`ssh-broker-config.xml` ファイルで、または **sshg3** コマンドラインで (`+x` オプションを使用して) 許可できます。デフォルトでは、X11 転送は無効になっています。

sshg3 によって設定された `DISPLAY` の値は、サーバ・マシンを指しますが、ディスプレイ番号は 0 よりも大きくなります。これは正常で、 **sshg3** がサーバ・マシン上に「プロキシ」X サーバを作成し、暗号化されたチャネルで接続を転送するために起こります。

sshg3 は、サーバ・マシン上の Xauthority データも自動的にセットアップします。この目的のために、このコマンドラインはランダムな認証クッキーを生成し、それをサーバ上の Xauthority データに保存し、転送されたすべての接続がこのクッキーを持つことを確認し、接続が開かれたときに本物のクッキーに置き換えられます。実際の認証クッキーがサーバ・マシンに送られることはありません (クッキーがプレーンで送られることもありません)。

TCP ポート転送

任意の TCP/IP 接続をセキュアなチャネルで転送することは、`ssh-broker-config.xml` ファイルで、または **sshg3** コマンドラインで (`-L` 及び `-R` オプションを使用して) 指定できます。

オプション

同じオプションが両方の場所で設定されている場合、コマンドライン・オプションは `ssh-broker-config.xml` ファイル内の設定を上書きします。以下のオプションがあります。

`-a, --no-agent-forwarding`

認証エージェント転送を無効にします。工場出荷時の設定では、エージェント転送は有効になっています。

`+a`

認証エージェント転送を有効にします。工場出荷時の設定では、エージェント転送は有効になっていますが、接続ブローカーの設定ファイルで禁止することができます。この場合、ユーザはコマンドラインから有効にすることができず、この `+a` は無視されます。

`-B, --batch-mode`

バッチ・モードを使用します。ターミナルでのユーザ操作を必要とする場合は、認証に失敗します。

バッチ・モードを使用するには、事前にサーバのホスト鍵をクライアントに保存し、ユーザ認証に非対話的な方法 (ホストベース認証やパスフレーズなしの公開鍵認証など) を設定しておく必要があります。

`-C`

現在の接続から圧縮を無効にします。

`+C`

この特定の接続に対して zlib 圧縮を有効にします。

`-c, --ciphers= LIST`

サーバに提示されることを許可された暗号を設定します。暗号名をカンマ区切りのリストにします。例:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている暗号名を表示するには、値として `help` を入力します。

`-D, --debug= LEVEL`

デバッグ・レベルを設定します。`LEVEL` は 0 から 99 までの数字で、99 はすべてのデバッグ情報を表示することを指定します。このオプションは、コマンドラインの最初の引数である必要があります。

注意

`-D` オプションは Unix でのみ適用されます。Windows では、接続ブローカーのデバッグ・オプションである `-D, -l` をこのコマンドライン・ツールの代わりに使用します。

注意

デバッグ・レベルは、 **sshg3** コマンドが接続ブローカーを起動するときにのみ設定できます。接続ブローカーがすでに実行されている場合、このオプションはコマンドに影響を与えません。

`-e, --escape-char= CHAR`

エスケープ文字を設定します (none: 無効、デフォルト: ~)。

`-f, --fork-into-background`

認証後、バックグラウンド・モードに移行します (Unix のみ)。このオプションはトンネル及びリモート・コマンドと併用します。 `-S` の意味もふくみます (コマンドが指定されていない限り)。トンネルが指定されている場合、このオプションによって **sshg3** はバックグラウンドで待機し、無期限に接続を待つようになります。 **sshg3** のリッスンを停止するには、kill コマンドを実行する必要があります。

`-g, --gateway`

ポートをゲートウェイします。つまり、ローカルで転送されるポートに他のホストも接続できることを意味します。このオプションは "`-L`" または "`-R`" オプションの前に指定する必要があります。このオプションの `+` と `-` のロジックに注意してください。

`+g`

ポートをゲートウェイしません。ローカルホストから送信されるトンネリング接続のみをリッスンします。これがデフォルト値です。このオプションの `+` と `-` のロジックに注意してください。

`-i FILE`

identification ファイルに定義された秘密鍵を公開鍵認証に使用することを定義します。

`-K, --identity-key-file= FILE`

指定された、秘密鍵または証明書の鍵ファイルをユーザ認証に使用することを定義します。鍵ファイルのパスはこのコマンドで指定します。

ファイルが秘密鍵の場合は、その秘密鍵が読み込まれ、接続ブローカーの鍵ストアにある既知の鍵と比較されます。鍵が既知ではない場合は、その鍵がデコードされ、鍵ストアに一時的に追加されます。ファイルが証明書であり、接続ブローカーが一致する秘密鍵を保持している場合は、その秘密鍵が使用されます。証明書と秘密鍵はどちらも、コマンドラインで複数の `-K` オプションを使用して指定できます。

`-L, --localfwd [ protocol/ ] [ listen-address: ] listen-port:dst-host:dst-port`

ローカル (クライアント) ホストのポートをリモートの接続先ホスト及びポートに転送します。

これは、ローカル・クライアント上のリスナ・ポート (`listen-port`) を割り当てます。このリスナへの接続が行われるたびに、接続は Secure Shell を介してリモート・サーバにトンネルされ、指定された接続先ホスト及びポート (`dst-host:dst-port`) にサーバからもう一つの接続が行われます。サーバからのその先の接続はセキュアではなく、通常の TCP 接続になります。

引数 `protocol` を指定すると、プロトコル固有の転送が有効になります。実装されているプロトコルは `tcp` (デフォルト、特別な処理は行わない)、 `ftp` (FTP データ・チャネル用に一時的な転送を作成し、FTP セッション全体を効果的に保護する)、及び `socks` です。

`socks` プロトコルの場合、引数の構文は "`-L socks/[listen-address:]listen-port`" です。これを設定すると、Tectia Client または ConnectSecure は他のアプリケーションの SOCKS サーバとして動作し、SOCKS トランザクションの要求に応じて転送を作成するようになります。これは SOCKS4 と SOCKS5 の両方をサポートしています。

`listen-address` を指定すると、クライアント側の全てのインターフェースをバインドするための `--gateway` が使用されない限りそのインターフェイスのみでリッスンされます。省略すると、すべてのインターフェイスでリッスンされます。

`-l, --user= USERNAME`

このユーザ名でログインします。

`-m, --macs= LIST`

サーバに提示されることを許可された MAC を設定します。MAC 名をカンマ区切りのリストにします。例:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている MAC 名を表示するには、値として `help` を入力します。

`-u, --kexs= kexs`

サーバに提示されることを許可された鍵交換 (KEX) 方式を設定します。鍵交換名をカンマ区切りのリストにします。例:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている鍵交換方法を表示するには、値として `help` を入力します。

`-j, --hostkey-algs= algs`

サーバに提示されることを許可されたホスト鍵アルゴリズムを設定します。ホスト鍵アルゴリズムをカンマ区切りのリストにします。例:

```
--hostkey-algs ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされているホスト鍵アルゴリズムを表示するには、値として `help` を入力します。

`-n, --dev-null` (Unix) , `-n, --null` (Windows)

`/dev/null` (Unix) 及び `NUL` (Windows) からの入力をリダイレクトします。

`-o option`

Tectia Client 4.x 形式の設定ファイルから読み込んだかのようにオプションを処理します。 `ForwardX11`、 `ForwardAgent`、 `AllowedAuthentications`、及び `PidFile` のオプションがサポートされています。たとえば、 `-o "ForwardX11=yes"` とします。 `-o "PidFile=/tmp/sshg3.pid"` とすると、sshg3 がバックグラウンドに移行した場合、そのプロセス ID が "/tmp/sshg3.pid" ファイルに保存されます。

`-P, --password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM`

パスワード認証の応答としてクライアントが送信するユーザ・パスワードを設定します。`PASSWORD` は、このオプションの引数として直接指定できます (推奨ではありません)。より良い代替手段は、パスワードを含むファイルへのパスを入力するか (`--password=file://PASSWORDFILE`)、パスワードを出力するプログラムまたはスクリプトへのパスを入力する (`--password=extprog://PROGRAM`) 方法です。

`extprog://` オプションを使用してシェル・スクリプトを参照する場合は、そのスクリプトがユーザのシェルを定義し、実際のパスワードを出力することも確認してください。そのようになっていない場合、シェル・スクリプトに使用するシェルを特定できないた

め、実行されたプログラムは失敗します。たとえば、パスワードの文字列が `my_password.txt` という名前のファイルに定義されていて、bash シェルを使用する場合は、以下の行をスクリプトに含めます。

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

警告

コマンドラインでパスワードを供給することは、安全な選択肢ではありません。たとえば、マルチユーザ環境では、コマンドラインで直接指定されたパスワードはプロセス・テーブルから容易に復元できます。よりセキュアな認証方法を設定する必要があります。非対話的なバッチ・ジョブの場合、パスフレーズなしの公開鍵認証やホストベース認証を使用する方がよりセキュアです。最低でも、ファイルやプログラムを使ってパスワードを供給してください。

`-p, --port= PORT`

リモート・ホストのこのポートに接続します。Secure Shell サーバが同じポートでリッスンしている必要があります。

`--publickey-algorithms= PUBLICKEY_ALGORITHMS`

選択された署名アルゴリズムのみが公開鍵認証に使用されることを許可します。例:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている署名アルゴリズムを表示するには、値として `help` を入力します。

`-q`

サイレント・モード。致命的なエラーのみを報告します。このオプションは、接続ブローカーの設定ファイルで行われた `quiet-mode` の設定を上書きします。

`-R, --remotefwd [ protocol/ ] [ listen-address: ] listen-port:dst-host:dst-port`

リモート (サーバ) ホストのポートをローカル側の宛先ホスト及びポートに転送します。

これは、リモート・サーバ上のリスナ・ポート (`listen-port`) を割り当てます。このリスナへの接続が行われるたびに、接続は Secure Shell を介してローカル・クライアントにトンネルされ、指定された接続先ホスト及びポート (`dst-host:dst-port`) にクライアントからもう一つの接続が行われます。クライアントからのその先の接続はセキュアではなく、通常の TCP 接続になります。

引数 `protocol` を指定すると、プロトコル固有の転送が有効になります。実装されているプロトコルは `tcp` (デフォルト、特別な処理は行わない) と `ftp` (FTP データ・チャネル用に一時的な転送を作成し、FTP セッション全体を効果的に保護する) です。

`listen-address` を指定すると、サーバ上のそのインターフェイスのみでリッスンされます。省略すると、すべてのインターフェイスでリッスンされます。

`-S, --no-session-channel`

セッション・チャネルを要求しません。これは、ポート転送要求でセッション・チャネル (及び tty) が不要な場合や、サーバがそれを与えない場合に使用できます。

`+S`

セッション・チャネルを要求します。これがデフォルト値です。

`-s, --subsystem subsystem remote_server`

リモート・サーバ上で起動するサブシステムまたはサービスを設定します。サブシステムはリモート・コマンドとして指定します。例: `sshg3 -s sftp <server>`

`-t, --tty`

コマンドが与えられても tty を割り当てます。

`-v, --verbose`

詳細モードを使用します。接続に失敗した場合は、より詳細な情報やエラー診断が出力されます。

`-x, -X, --no-x11-forwarding`

X11 接続転送を無効にします。これがデフォルト値です。

`+x, +X`

X11 接続転送を有効にします。

`-z, --broker-log-file= FILE`

接続ブローカーのログ・ファイルを `FILE` に設定します。このオプションは、 **ssh-broker-g3** がこのプロセスによって起動される場合にのみ機能します。

`--aa, --allowed-authentications= METHODS`

ユーザ認証で使用を許可する方法を定義します。方法をカンマ区切りのリストにします。例:

```
--allowed-authentications keyboard-interactive,password
```

現在サポートされている認証方法を表示するには、値として `help` を入力します。

`--abort-on-failing-tunnel`

トンネル・リスナの作成に失敗した場合（ポートがすでに予約されている場合など）に中断します。

`--any-alg`

サポートされている全ての暗号、 MAC、 鍵交換、 ホスト鍵及び公開鍵アルゴリズムが使用されることを許可します。

`--compressions= METHODS`

サーバに提示されることを許可された圧縮方法を設定します。方法をカンマ区切りのリストにします。

現在サポートされている圧縮方法を表示するには、値として `help` を入力します。

`--disconnect-message=MESSAGE`

接続を切断したときに表示されるメッセージです。切断メッセージの値は文字列です。このメッセージには以下の変数を複数指定することができます。

- `time`: 切断の時刻
- `random`: 16桁のランダムな16進数
- `random4`: 4桁のランダムな16進数
- `random8`: 8桁のランダムな16進数
- `random16`: `random` と同じです
- `pid`: sshg3 のプロセスID
- `broker_pid`: ブローカのプロセスID
- `conn_id`: 接続ID
- `session_id`: セッションID
- `target_host`: ターゲット・サーバ名
- `target_port`: ターゲット・サーバ・ポート

ランダムな値のいずれかが切断メッセージに使用されると、その値は認証成功メッセージの前にユーザに表示されます。接続の前後で値が異なる場合は、接続が何者かにスプーフィングされている可能性があります。

切断メッセージはデフォルトでは無効です。

`--exclusive`

接続を試みるたびに新しい接続を開くことを定義します。これを定義しない場合、接続ブローカーは最近閉じた接続を再使用できます。

`--hostkey-policy= POLICY`

サーバ・ホスト鍵のチェックと未知のサーバ・ホスト鍵の処理に関するポリシーを定義します。指定できる値は以下の通りです。

- `ask` (デフォルト): ホスト鍵ストレージに鍵が見つからない場合や、鍵が変更されている場合、ユーザはサーバ・ホスト鍵を検証して受け入れるよう求められます。

- `strict`: サーバへの接続は、ホスト鍵がユーザの既知のホスト鍵のストレージで見つかった場合にのみ許可されます。
- `tofu`: Trust On First Use (TOFU)。新しいホスト鍵は、それを受け入れることをユーザに求めることなく保存されます。
- `advisory` (推奨しません): 新しいホスト鍵は、それを受け入れることをユーザに求めることなく保存され、変更されたホスト鍵を提供するサーバへの接続も許可されます。

警告

ポリシーを `advisory` に設定する前に、慎重に検討してください。ホスト鍵のチェックを無効にすると、接続が攻撃に対して脆弱になります。

`ssh-broker-config.xml` 設定ファイルの `default-settings` にある `<auth-server-publickey>` エレメントで、またはプロファイル毎にホスト鍵ポリシーを設定することもできます。 **auth-server-publickey** を参照してください。

このオプションがコマンドライン・クライアントと、`ssh-broker-config.xml` で設定されている場合、コマンドラインの値が使用されます。

`--identity= ID`

秘密鍵の ID をユーザ認証に使用することを定義します。ID には、接続ブローカー内部の鍵の番号、鍵のハッシュ、または鍵ファイル名のいずれかを指定できます。

`--identity-key-hash ID`

ユーザ認証に使用する秘密鍵を、それに対応する公開鍵のハッシュで定義します。

`--identity-key-id ID`

接続ブローカー内部の鍵番号をユーザ認証に使用することを定義します。

`--keep-alive= VALUE`

キープアライブ・メッセージを Secure Shell サーバに送信する頻度を定義します。値を秒単位で入力します。デフォルト値は `0` で、キープアライブ・メッセージが無効であることを意味します。

`--kip`

ユーザ認証に許可する方法として、キーボード・インタラクティブとパスワードを定義します。以下と同じです。

```
--allowed-authentications keyboard-interactive,password
```

`--remote-environment name= VALUE`

このオプションを使用すると、定義された環境変数がクライアント側からサーバに渡されます。コマンド、シェル、またはサブシステムを要求する際にサーバ上で環境変数が適用されます。

サーバは環境変数の設定を制限できます。

`ssh-broker-config.xml` 設定ファイルの `default-settings` にある `<remote-environment>` エレメントで、またはプロファイル毎にサーバに渡す環境変数を設定することもできます。**remote-environment** を参照してください。

同じ変数がコマンドライン・クライアントに入力され、`ssh-broker-config.xml` で設定されている場合、コマンドライン・バージョンが使用されます。

`--remote-environment-format name= VALUE`

定義された環境変数がクライアント側からサーバに渡されます。接続ブローカーはその値を処理してからサーバに送信します。

`value` に `%U` を使用すると、ユーザ名を示すことができます。接続ブローカーは `%U` を実際のユーザ名に置き換えてからサーバに送信します。

詳細については、上記の `--remote-environment` オプションを参照してください。

`--tcp-connect-timeout= VALUE`

Secure Shell サーバへの TCP 接続を確立するためのタイムアウト時間 (秒単位) を定義します。値を正の数で入力します。

`--template-profile profile`

接続時に指定されたプロファイルを使用します。

これは、例えば最新のアルゴリズムをサポートしないレガシー・システムに接続するとき等に便利です。このような場合は `ssh-broker-config.xml` 設定ファイルに以下のように要求されるアルゴリズムを含むプロファイルを定義できます。

```
<profile name="legacy"
         host=""
         id="legacy">
  <kexs>
    <kex name="diffie-hellman-group1-sha1" />
  </kexs>
  <hostkey-algorithms>
    <hostkey-algorithm name="ssh-dss" />
  </hostkey-algorithms>
  <ciphers>
    <cipher name="aes128-cbc" />
  </ciphers>
</profile>
```

その後、以下のように接続します。

```
$ sshg3 --template-profile legacy user@host
```

注: セキュアではないアルゴリズムの使用はデータ漏洩につながります。レガシー・サーバを最新のアルゴリズムをサポートス路用アップグレードすることを推奨します。

`-V, --version`

プログラムのバージョンを表示して終了します。

`-h, --help, -?`

コマンドライン・オプションの要約を表示して終了します。

コマンド

sshg3 では、以下のいずれかのコマンドを実行できます。

`remote_command [arguments] ...`

リモート・ホストでコマンドを実行します。

`-s service`

リモート・サーバのサービスを有効にします。

エスケープ・シーケンス

sshg3 は、実行中のセッションを管理するためのエスケープ・シーケンスをサポートしています。エスケープ・シーケンスを有効にするには、改行文字の後に直接入力する必要があります (最初に **Enter** キーを押します)。入力中の画面にエスケープ・シーケンスは表示されません。

以下のエスケープ・シーケンスがサポートされています。

~.

接続を終了します。

~`Ctrl`**-Z**

セッションを一時停止します。

~~

エスケープ文字をその文字として送信します。

~#

転送された接続を一覧表示します。

~-

エスケープ文字を不可逆的に無効にします。

~?

エスケープ・シーケンスの要約を表示します。

~r

手動で鍵更新を開始します。

~s

サーバとクライアントのバージョン、受信パケット、送信パケット、圧縮、鍵交換アルゴリズム、公開鍵アルゴリズム、対称暗号など、接続の統計データを提供します。

~u

選択した公開鍵を自動的にサーバにアップロードします。ユーザが鍵を 1 つしか持っていない場合は、その鍵がアップロードされます。それ以外の場合は、 `id_rsa_<size>_a` に一致する名前を持つ最大の鍵が選択されます。

~U

公開鍵をサーバにアップロードします。利用できる鍵の一覧が表示され、ユーザはアップロードする鍵を選択するよう求められます。

~c

個々のチャネルの統計データ（データ・ウィンドウ・サイズなど）を表示します。これはデバッグのためです。

~V

クライアントのバージョン番号を stderr にダンプします (トラブルシューティングに活用できます)。

環境変数

Secure Shell サーバは接続時に、 **sshg3** が使用できる複数の環境変数を自動的に設定します。設定される正確な変数は、Secure Shell サーバによって異なります。 **sshg3** では以下の変数を使用できます。

`DISPLAY`

`DISPLAY` 変数は X11 サーバの場所を示します。これは、 `hostname:n` の値を指すように、サーバによって自動的に設定されます。ここで `hostname` は、サーバとシェルが動作しているホストを示し、 `n` は 1 以上の整数です。 **sshg3** この特殊値を使用して、X11 の接続をセキュアなチャネルに転送します。

通常、 `DISPLAY` を明示的に設定してはなりません。X11 接続が保護されなくなります (必要な認証クッキーを手動でコピーしなければならなくなります)。

`HOME`

ユーザのホーム・ディレクトリ

LOGNAME

USER の同義語です。この変数を使用しているシステムとの互換性のために設定されます。

MAIL

ユーザのメールボックス

PATH

デフォルトの PATH に設定されます。オペレーティング・システムによって、または一部のシステムでは /etc/environment または /etc/default/login になります。

SSH_SOCKS_SERVER

sshg3 で使用される SOCKS サーバのアドレス。

SSH2_AUTH_SOCK

これが存在する場合、認証エージェント (またはそのローカルの代替) との通信に使用する Unix ドメインのソケットのパスを示すために使用されます。

SSH2_CLIENT

接続のクライアント側を識別します。この変数には、スペースで区切られた 3 つの値 (クライアント IP アドレス、クライアント・ポート番号、及びサーバ・ポート番号) が含まれています。

SSH2_ORIGINAL_COMMAND

強制コマンドを実行した場合、これは **sshg3** に渡された元のコマンドになります。たとえば、相手側から引数を取得するために使用できます。これは実際のコマンドである必要はなく、ファイル名やデバイス名、パラメータ名など、自由に設定できます。

SSH2_TTY

これは、現在のシェルまたはコマンドに関連する tty の名前 (デバイスへのパス) に設定されます。現在のセッションに tty がない場合、この変数は設定されません。

TZ

time-zone 変数は、サーバの起動時に設定されていれば、現在のタイム・ゾーンを示すように設定されます (サーバはこの値を新しい接続に渡します)。

USER

ユーザの名前

Tectia Server で設定される変数のリストについては、 ssh-server-g3(8) man ページを参照してください。

終了値

sshg3 は、操作の結果に基づいて以下の値を返します。

```
0     Operation was successful.
1     sshg3 has encountered an error,
      the reason is usually given in an error message.
```

リモート・コマンドを実行すると、 **sshg3** はコマンド実行のステータスを、以下の終了コードで表示して終了します。

```
0     The remote command was run successfully.
127   The requested remote command was not found.
```

例

ローカル・ユーザ名でホスト `remotehost`、ポート番号 2222 に接続し、シェルを開きます。

```
$ sshg3 remotehost#2222
```

`ssh-broker-config.xml` ファイルの接続プロファイル `profile1` で指定されたホストに接続し、 `who` コマンドを実行します (コマンドの実行後に終了します)。

```
$ sshg3 profile1 who
```

ホスト `remotehost` に `user` として接続し、クライアントのポート番号 143 から `imapserver` のポート番号 143 へとローカル・ポート転送を開きます。シェルは開きません。他のホストもこのローカル・ポートに接続できます。`remotehost` から `imapserver` への接続はセキュアではなくなります。

```
$ sshg3 -L 143:imapserver:143 -g -S user@remotehost
```

scpg3

scpg3 — Secure Shell ファイル・コピー・クライアント - Generation 3

書式

`scpg3 [ options...]`
`[ src_profile: | [user@] src_host [#port]: ] src_file...`
`[ dst_profile: | [user@] dst_host [#port]: ] dst_file_or_dir`

説明

scpg3 (Windows では **scpg3.exe**) は、ネットワーク上でファイルをセキュアにコピーするために使用されます。 **scpg3** は、Secure Shell バージョン 2 プロトコルを使用したセキュアな転送を提供するために **ssh-broker-g3** を起動します。 **ssh-broker-g3** は、認証のためにパスワードやパスフレーズが必要な場合にそれらを要求します。 **scpg3** は、`ssh-broker-config.xml` ファイルで指定された設定を使用します。

2 つのリモート・ホスト間のコピーもできます。リモート・ホストは、 **sftp-server** (または **sft-server-g3**) サブシステムが有効になっている Secure Shell バージョン 2 サーバを実行している必要があります。Tectia Server ではデフォルトで、 **sft-server-g3** が有効になっています。

ファイル名には、そのファイルがアップロードまたはダウンロードされることを示すために、ホスト、ユーザ、及びポートが指定できます ([`user@`] `host` [`#port`])。ユーザ名を指定しない場合は、ローカル・ユーザ名が使用されます。ポートを指定しない場合、デフォルトの Secure Shell ポート 22 が使用されます。その代わりに、`ssh-broker-config.xml` ファイルに定義されている接続プロファイル (`profile`) を指定することもできます。

注意

scpg3 コマンドで接続プロファイルを入力する場合、Tectia Client は引数の形式によって異なる意味に解釈されることに注意してください。指定された属性値に `@` 記号がある場合、Tectia Client は常に `<username@hostname>`、つまりプロファイルではないと解釈します。

また、プロファイル名にドットがある場合 (例: `host.x.example.com`)、コマンドライン上でそのドットをエスケープする必要があります。Unix では、 Windows では、代わりに `host~.x~.example~.com` と入力します。このようにしないと、プロファイル名がホスト名として扱われ、現在の Windows ユーザ名がログインに使用されます。

`host` パラメータはオプションで角括弧 ([]) で囲み、セミコロンを使用できるようにすることができます。`file` 引数には単純なワイルドカードを指定できます。アスタリスク (*) は任意の数の任意の文字、クエスチョン・マーク (?) は任意の 1 文字を意味します。

ファイル名の特殊文字については、「ファイル名のサポート」 を参照してください。

オプション

以下のコマンドライン・パラメータを使用すると、 **scpg3** のオプションを詳細に指定できます。

`-4`

すべての接続に関連する DNS の解決が IPv4 アドレスとして解決されることを定義します。

`-6`

すべての接続に関連する DNS の解決が IPv6 アドレスとして解決されることを定義します。

`-a [ arg ]`

アスキー・モードを使用してファイルを転送します。つまり、改行はオンザフライで変換されます。z/OS の Tectia と他のホスト間の転送では、アスキーと EBCDIC の自動変換も有効になります。 **sftpg3ascii** コマンドを参照してください (「コマンド」)。

サーバが改行規則を通知せず、そのホスト・タイプを指定するホスト・プロファイルを使用していない場合、 `-a` の後に引数を与えることで、 **scpg3** にヒントを与えることができます。デフォルトでは、転送先の改行規則を設定しますが、引数の前に `src:` または `dest:` を付けることで、それぞれ転送元と転送先の改行規則を指定できます。使用できる規則は `dos`、 `unix`、及び `mac` で、それぞれ `\r\n`、 `\n`、及び `\r` を改行として使用します。 `-a` と引数の間にはスペースを入れません。以下に例を示します。

```
$ scpg3 -asrc:unix -adest:dos src_host:src_file dest_host:dest_file
```

改行規則を強制するには、 `force-dos`、 `force-unix`、及び `force-mac` の値を使用します。これらの設定は、リモート SSH サーバが SCP クライアントに示唆する内容に関係なく、改行モードを強制します。

`-B, --batch-mode`

バッチ・モードを使用します。ターミナルでのユーザ操作を必要とする場合は、認証に失敗します。

バッチ・モードを使用するには、事前にサーバのホスト鍵をクライアントに保存し、ユーザ認証に非対話的な方法　(ホストベース認証やパスフレーズなしの公開鍵認証など) をセットアップしておく必要があります。

`-b  buffer_size_bytes`

SFTP プロトコルの 1 回の読み取りまたは書き込み要求のための最大バッファ・サイズを定義します (デフォルト: `32768` バイト)。

1 つのファイルの転送中に並行して送信される SFTP プロトコルの読み取りまたは書き込み要求の最大数は、`-N` オプションで指定できます。

ストリーミング (--streaming を参照) が使用されている場合 (デフォルトでは buffer_size_bytes より大きなファイルを Tectia Server との間で転送するため)、このオプションはバッファ・サイズの定義に使用されないことに注意してください。

-C

現在の接続で圧縮を無効にします。

+C

この特定の接続で zlib 圧縮を有効にします。

-c, --ciphers= LIST

許可された暗号を設定し、サーバに提供します。暗号名をカンマ区切りのリストにします。例:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

値 any は全てのサポートされているアルゴリズムを許可します。現在サポートされている暗号名を表示するには、値として help を入力します。

-D, --debug= LEVEL

デバッグ・レベルを設定します。LEVEL は 0 から 99 までの数字で、99 はすべてのデバッグ情報を表示することを指定します。このオプションは、コマンドラインの最初の引数である必要があります。

注意

-D オプションは Unix でのみ適用されます。Windows では、接続ブローカーのデバッグ・オプションである -D, -l をこのコマンドライン・ツールの代わりに使用します。

注意

デバッグ・レベルは、 **scpg3** コマンドが接続ブローカーを起動するときにのみ設定できます。接続ブローカーがすでに実行されている場合、このオプションはコマンドに影響を与えません。

-d

ターゲットがディレクトリであることを強制します。

-I, --interactive

既存の転送先ファイルを上書きするかどうかの確認を求めます (-B では動作しません)。

-i FILE

identification ファイルに定義された秘密鍵を公開鍵認証に使用することを定義します。

`-K, --identity-key-file= FILE`

指定された秘密鍵または証明書の鍵ファイルをユーザ認証に使用することを定義します。鍵ファイルのパスはコマンドで指定します。

ファイルが秘密鍵の場合は、その秘密鍵が読み込まれ、接続ブローカーの鍵ストアにすでに保持されている鍵と比較されます。鍵が既知ではない場合は、その鍵がデコードされ、鍵ストアに一時的に追加されます。ファイルが証明書であり、接続ブローカーが一致する秘密鍵を保持している場合は、その秘密鍵が使用されます。証明書と秘密鍵はどちらも、コマンドラインで複数の `-K` オプションを使用して指定できます。

`-m fileperm [:dirperm]`

このオプションは Windows でのみ使用できます。Unix サーバにファイルをアップロードする際のデフォルトのファイル及びディレクトリのパーミッション・ビットを設定します。

`-N max_requests`

並行して送信される、SFTP プロトコルの読み取りまたは書き込み要求の最大数を定義します (デフォルト: 10)。

各読み取りまたは書き込み要求で使用されるバッファのサイズは、`-b` オプションで指定できます。

この値は 1 つのファイルの転送に適用されます。並行して送信されるファイルの数を定義するために使用することはできません。

ストリーミング (`--streaming` を参照) が使用されている場合 (デフォルトでは `-b` オプションで指定した `buffer_size_bytes` より大きなファイルを Tectia Server との間で転送する場合ため)、このオプションは使用されません。

`-O, --offset= r<offset> | w<offset> | l<length> | t<length>`

オフセットを設定します。オフセット `r<offset>` は転送元ファイル内の開始オフセットを指定します。オフセット `w<offset>` は転送先ファイル内の開始オフセットを指定します。長さ `l<length>` は、コピーするデータ量を指定します。切り詰めの長さ `t<length>` を指定した場合、ファイル・データのコピー後に、転送先ファイルの切り詰めまたは拡張を行う長さを指定します。

`-p`

送信元と送信先が両方とも Unix ファイル・システム上 (z/OS USS を含む) にある場合、ファイルのパーミッションとタイムスタンプを保持します。転送元または転送先のいずれかが Windows 上にある場合、タイムスタンプは保持しますが、ファイルのパーミッションは保持しません。転送先が z/OS MVS 上にある場合、何も保持されません。

`-P port`

リモート・マシンのこの Secure Shell ポートに接続します (デフォルト: `22`)。

`-Q`

進捗インジケータを表示しません。このオプションの効果は `--progress-display=no` を使用した場合と同じです。

このオプションはパラメータ `--statistics` と併用しないでください。

`-q`

サイレント・モードを使用します (致命的なエラーのみ表示されます)。このオプションは、接続ブローカーの設定ファイルで行われた `quiet-mode` の設定を上書きします。

`-r`

サブディレクトリを再帰的に処理します。

`-u, --unlink-source`

コピー後、転送元ファイルを削除します。 (ファイル移動)

`-v, --verbose`

詳細モードを使用します (`-D 2` と同じ)。

`-W, --whole-file`

インクリメンタル・チェックを試みません。デフォルトでは (このオプションが指定されていない場合)、インクリメンタル・チェックが試みられます。このオプションは `--checksum` オプションと一緒にしか使用できません。

`--aa, --allowed-authentications= METHODS`

許可されたユーザ認証の方法をを設定し、サーバに提供します。方法のカンマ区切りリストを指定します。例:

```
--allowed-authentications keyboard-interactive,password
```

現在サポートされている認証方法を表示するには、値として `help` を入力します。

`--any-alg`

サポートされている全ての暗号、 MAC、 鍵交換、 ホスト鍵及び公開鍵アルゴリズムが使用されることを許可します。

`--append`

転送先ファイルの末尾にデータを追加します。

`--binary`

バイナリ転送モードを使用します。サーバが Tectia Server for IBM z/OS の場合、アスキーから EBCDIC への変換を行わないようにサーバに要求し、ファイルは Stream 形式で転送されます。 `--src-site` 及び `--dst-site` オプションを使用して値を変更できます。

`--checkpoint=b <bytes>`

チェックポイント更新間隔をバイトで指定します (デフォルト: 10 MB)。このオプションは、 `--checksum=checkpoint` のときにのみ使用できます。

`--checkpoint=s <seconds>`

チェックポイントの更新間隔を時間で指定します (デフォルト: `10` 秒)。このオプションは、 `--checksum=checkpoint` のときにのみ使用できます。

`--checksum [ =yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]`

ファイル転送を再開できるファイル内のポイントを特定するために、MD5 または SHA-1 チェックサム、あるいは独立したチェックポイントデータベースを使用します。 `buffer_size_bytes` より小さいファイルはチェックされません。小さなファイルには `md5-force` または `sha1-force` を使用します (デフォルト: `yes`。この場合、FIPS モードでは SHA1 チェックサムを使用し、それ以外では MD5 チェックサムを使用します)。大きなファイルを 1 つずつ転送する場合は、チェックポイントを使用します。

`--compressions= METHODS`

許可された圧縮方法を設定し、サーバに提供します。方法のカンマ区切りリストを指定します。

現在サポートされている圧縮方法を表示するには、値として `help` を入力します。

`--dst-site= PARAM`

指定されたサイト・パラメータを転送先ファイルで使用します。 **sftpg3 site** コマンドを参照してください (「コマンド」)。

`--exclusive`

接続を試みるたびに新しい接続を開くことを定義します。これを定義しない場合、接続ブローカーは最近閉じた接続を再使用できます。

`--fips`

FIPS 暗号化ライブラリを使用してチェックサムを実行します。

`--force-lower-case`

転送先ファイル名が小文字に変換されます。

`--hostkey-algorithms= HOSTKEYALGORITHMS`

許可されたホスト鍵アルゴリズムを設定し、サーバに提供します。ホスト鍵アルゴリズムのカンマ区切りリストを指定します。例:

```
--hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされているホスト鍵アルゴリズムを表示するには、値として `help` を入力します。

`--overwrite [ =yes | no ]`

既存の転送先ファイルを上書きするかどうかを選択します (デフォルト: `yes`)。

`--identity= ID`

秘密鍵の ID をユーザ認証に使用することを定義します。ID には、接続ブローカー内部の鍵番号、鍵のハッシュ、または鍵ファイル名のいずれかを指定できます。

`--identity-key-hash= ID`

ユーザ認証に使用する秘密鍵と、それに対応する公開鍵のハッシュを定義します。

`--identity-key-id= ID`

接続ブローカー内部の鍵番号をユーザ認証に使用することを定義します。

`--keep-alive= VALUE`

キープアライブ・メッセージ (非操作パッケージ) を Secure Shell サーバに送信する頻度を定義します。値を秒単位で入力します。デフォルト値は 0 で、キープアライブ・メッセージが無効であることを意味します。

`--kexs= kexs`

許可された鍵交換 (KEX) 方式を設定し、サーバに提供します。鍵交換名のカンマ区切りリストを指定します。例:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている鍵交換方法を表示するには、値として `help` を入力します。

`--kip`

キーボード・インタラクティブとパスワードを、ユーザ認証に許可する方法として定義します。以下と同じです。

```
--allowed-authentications keyboard-interactive,password
```

`--macs= LIST`

許可された MAC を設定し、サーバに提供します。MAC 名のカンマ区切りリストを指定します。例:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている MAC 名を表示するには、値として `help` を入力します。

`--password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM`

パスワード認証の応答としてクライアントが送信するユーザ・パスワードを設定します。`PASSWORD` は、このオプションの引数として直接指定できます (推奨されません)。よ

り良い代替手段は、パスワードを含むファイルへのパスを入力するか (`--password=file://PASSWORDFILE`)、パスワードを出力するプログラムまたはスクリプトへのパスを入力する (`--password=extprog://PROGRAM`) 方法です。

`extprog://` オプションを使用してシェル・スクリプトを参照する場合は、そのスクリプトがユーザのシェルを定義し、実際のパスワードを出力することも確認してください。そのようになっていない場合、シェル・スクリプトに使用するシェルを特定できないため、実行されたプログラムは失敗します。たとえば、パスワードの文字列が `my_password.txt` という名前のファイルに定義されていて、bash シェルを使用する場合は、以下の行をスクリプトに含めます。

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

警告

コマンドラインでパスワードを指定することは、安全な選択肢ではありません。たとえば、マルチユーザ環境では、コマンドラインで直接指定されたパスワードはプロセス・テーブルから容易に復元できます。よりセキュアな認証方法をセットアップする必要があります。非対話的なバッチ・ジョブの場合、パスフレーズなしの公開鍵認証やホストベース認証を使用する方がよりセキュアです。最低でも、ファイルやプログラムを使ってパスワードを供給してください。

`--plugin-path=PATH`

プラグイン・パスを `PATH` に設定します。これは FIPS モードでのみ使用します。

`--prefix=PREFIX`

ファイルが転送されている間にのみ、ファイル名にプレフィックスを付加します。ファイル転送に成功すると、付加されたプレフィックスは削除されます。

z/OS では、MVS データ・セット名にプレフィックスを適用する場合、ハイレベル修飾子 (HLQ) の後にプレフィックスが挿入されます。プレフィックスを独立した修飾子にする場合は、プレフィックスの末尾にドットを含めます。

```
--prefix=PREFIX.
```

`--publickey-algorithms=PUBLICKEY_ALGORITHMS`

選択された署名アルゴリズムのみが公開鍵認証に使用されることを許可します。例:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている署名アルゴリズムを表示するには、値として `help` を入力します。

`--src-site=PARAM`

指定されたサイト・パラメータを転送元ファイルで使用します。 **site** コマンドを参照してください (「コマンド」)。

`--statistics [ =no | yes | simple ]`

注意

リリース 6.1.5 では `--statistics` オプションの動作が変更され、 `--statistics-format` オプションが削除されました。これらの代わりに、新たに `--summary-display` オプションと `--summary-format` オプションを使用します。

`--statistics` オプションは、ファイル転送操作の後に表示される統計データのスタイルを選択します。 `--statistics` と `--summary-display` は併用しないでください。

`--statistics` オプションは以下の値を取ります。

`no` - 統計データは作成されません。

`yes` - ファイル転送中に進捗バーを表示します。これがデフォルトです。出力の例を以下に示します。

```
scpg3 --statistics="yes" sourcefile destinationfile
sourcefile                 |  127MB |  42.9MiB/s | TOC: 00:00:03 | 100%
```

`simple` - ファイル転送が終了すると、1 行の簡単な統計が表示されます。例:

```
scpg3 --statistics=simple sourcefile destinationfile
sourcefile   | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`--summary-display [ =no | yes | simple | bytes ]`

ファイル転送操作の後に表示される、ファイル転送サマリ・データのスタイルを選択します。サマリの表示とともに、進捗バーのデータもデフォルトで表示されます。

`--summary-display` と `--statistics` は併用しないでください。

`--summary-display` オプションは以下の値を取ります。

`no` - サマリ・データは作成されません。これがデフォルトです。

`yes` - 詳細なサマリ・データが作成されます。 `summary-format` オプションで内容を設定できます。デフォルトでは、以下の内容がサマリに表示されます。

```
Default settings:                   Render for example this:
"Source: %c:%g\n"                   user@host1#22:/path/to/source/file
"Source parameters: %e\n"           X=TEXT, C=ISO8859-1,D=IBM.1047
"Destination: %C:%G\n"              user@host2#22:/path/to/destination/file
"Destination parameters: %E\n"      NONE
"File size: %s bytes\n"             123456 bytes
"Transferred: %t bytes\n"           123456 bytes
"Rate: %RB/s\n"                     345kiB/s
"Start: %xy-%xt-%xd %xh:%xm:%xs\n"  2010-01-26 13:10:56
"Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n"   2010-01-26 13:23:30
"Time: %y\n"                        00:12:34
```

`simple` - 1 行の簡単なサマリが表示されます。例:

```
scpg3 --summary-display=simple sourcefile destinationfile
```

```
sourcefile | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`bytes` - 転送されたバイト数を報告する基本的な統計データが表示されます。例:

```
scpg3 --summary-display=bytes sourcefile destinationfile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

`--summary-format= FORMAT_STRING`

サマリの形式と内容を選択します。このオプションは、 `--summary-display=yes` のときに使用できます。このオプションは `--statistics` と併用しないでください。

以下の定義を使用して、サマリの内容を選択します。

```
%c  - source connection: user@host#port or profile
%C  - destination connection: user@host#port or profile
%D* - current date
%e  - source parameters (file transfer and data set parameters)
%E  - destination parameters (file transfer and data set parameters)
%f  - source file name
%F  - destination file name
%g  - /path/to/source/file
%G  - /path/to/destination/file
%k  - compression done ("zlib" or "none")
%p  - transfer percentage
%q  - transfer rate in bit/s
%Q  - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r  - transfer rate in bytes/s
%R  - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%s  - file size in bytes
%S  - file size as "XXyB" (B, kiB, MiB or GiB)
%t  - transfer size in bytes
%T  - transfer size as "XXyB" (B, kiB, MiB or GiB)
%x* - start date
%X* - end date
%y  - elapsed time
%Y  - time remaining
%z  - ETA or TOC, if transfer has finished
%Z  - string "ETA" or "TOC", if transfer has finished

Where * is one of the following:

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

Other special characters in format strings are:

\n - line feed
\r - carriage return
\t - horizontal tab
```

```
\\ - backslash
```

`--progress-display [ =no | bar | line ]`

ファイル転送中の進捗状況を表示するモードを選択します。デフォルトは `bar` で、進捗バーが表示されます。`line` オプションは、`--progress-line-format` オプションで行われた設定に従って進捗情報を表示します。

このオプションは `--statistics` と併用しないでください。

`--progress-line-format= FORMAT_STRING`

進捗ラインに表示する情報を選択します。このオプションは、`--progress-display=line` のときに使用できます。

このオプションは `--statistics` と併用しないでください。

次のコマンドで記述した定義を使用して、進捗ラインの内容を選択します: `--summary-format`

`--progress-line-interval= seconds`

ライン・モードでの進捗情報の更新頻度を定義します。間隔は秒単位で指定します。デフォルトは 60 秒です。

このオプションは `--statistics` と併用しないでください。

`--streaming [ =yes | no | force | ext ]`

サーバがサポートしている場合、ファイル転送にストリーミングを使用します。`buffer_size_bytes` より小さいファイルはストリーミングを使用して転送されません。小さなファイルには `force` を使用します。デフォルト: `yes`

MVS データ・セットへの直接アクセスを可能にするには、z/OS ホストで `ext` を使用します。その他の環境では小さなファイルの転送が遅くなることがあるため、主にメインフレームのデータ・セット転送に使用する場合にのみ、このオプションを使用してください。

チェックサムが計算される場合、ファイル転送はステージングを使用し、ストリーミングは除外されるため、`--streaming=ext` オプションには `--checksum=no` オプションも必要です。

`--sunique`

ファイルを一意の名前で保存します。転送されたファイルの中に同じ名前のファイルが複数ある場合、この機能は、繰り返されるファイル名の末尾に連番を付加します (例: `file.name`、`file.name1`、及び`file.name2`)。

`--tcp-connect-timeout= VALUE`

Secure Shell サーバへの TCP 接続を確立するときのタイムアウト時間 (秒単位) を定義します。タイムアウトの値を正の数で入力します。値を 0 (ゼロ) にすると、この機能が無効になり、代わりにデフォルトのシステム TCP タイムアウトが使用されます。

`--template-profile profile`

指定したプロファイルを接続に使用します。

`--user= USERNAME`

アドレス文字列でユーザ名が提供されていない場合、このユーザ名を使用してログインします。

`-V, --version`

プログラムのバージョンを表示して終了します。

`-h, --help, -?`

コマンドライン・オプションの要約を表示して終了します。

ファイル名のサポート

ファイル名に使用できる文字コードは、オペレーティング・システムによって異なります。Unix ではファイル名に使用できる特殊文字がありますが、Windows では以下の文字は使用できません。

```
\/ : * ? " < > |
```

scpg3 コマンドを使用して、特殊文字を含むファイル (例: `unixfilename*?".txt`) を Unix サーバから Windows にコピーする場合、Windows で通用する新しい名前をファイルに付ける必要があります。以下のフォーマットでコマンドを入力します。

```
$ scpg3 user@unixserver:"unixfilename~*~?\".txt" windowsfilename.txt
```

特殊文字を含むファイル名を **scpg3** コマンドの引数として入力する場合は、プラットフォーム固有の構文に従うのが一般的なルールです。

Tectia は、アスキー文字のみを含むファイル名を完全にサポートしています。その他の文字コードを使った文字が含まれるファイル名については、動作を保証しません。

ワイルドカードの使用

scpg3 コマンドはワイルドカードとして `*` と `?` をサポートしています。

これらのワイルドカードは、コマンドの中でリモート側とローカル側の両方で使用できます。以下のコマンドの例では、ディレクトリ `dir2` のサブディレクトリのうち、名前がプレフィックス `data-` で始まるすべてのディレクトリから、すべてのテキスト・ファイル (`*.txt`) を現在のローカル・ディレクトリ (`.`) にコピーします。

```
$ scpg3 -r user@server:"dir2/data-*/*.txt" .
```

Unix では、ファイル名に * と ? の文字も使用できます。そのため、ファイル名に属する文字とワイルドカードを区別するために、エスケープ文字を使用する必要があります。詳細については、「特殊文字のエスケープ」tを参照してください。

特殊文字のエスケープ

ある OS でファイル名に使用されている特殊文字は、Tectia のコマンドで特別な意味を持つ場合があります。また、ファイル転送システムのさまざまな部分で意味が異なる可能性があります。

scpg3 コマンドでは、以下の文字は特別な意味を持ち、ファイル名を引数として取るコマンドではエスケープする必要があります。

***** : は任意の数の任意の文字に対応するワイルドカードです。

?: クエスチョン・マークは任意の 1 文字に対するワイルドカードです。

" クォーテーション・マークは、表示通りに扱う文字列の前後に配置します。

\: バックスラッシュは Unix のエスケープ文字です。

~ チルダは Windows のエスケープ文字です。

エスケープ文字は **scpg3** コマンドに対して、次に来る文字を表示通りに扱い、特別な意味を持たせないように指示します。エスケープ文字はローカル・マシンのオペレーティング・システムに応じて選択されます。

\ 及び ~は特殊文字そのものであり、ファイル名に使用する場合は、その前にもエスケープ文字を配置する必要があります。したがって、Unix では \、Windows では ~ を含むファイル名を **scpg3** コマンドに入力する必要がある場合は、該当する以下のエスケープ文字を付加します。

\\ : Unix の場合

~~: Windows の場合

Tectia の **scpg3** コマンドでエスケープ文字を使用する方法、及び異なるオペレーティング・システムで特殊文字を含むファイル名を入力する方法については、以下の例を参照してください。

scpg3 コマンドのファイル名の例:

以下のファイル名は Unix で有効ですが、コマンド内でエスケープ文字が必要です。

```
file|name.txt
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

Unix で **scpg3** コマンドを使用する場合、エスケープ文字が互いにエスケープし合うため、複数のエスケープ文字が必要になることがあります。上記のファイル名は以下のフォーマットで入力します。

```
file\|name.txt     or  "file|name.txt"
```

```
file-\"name\".txt
file\\\?name.txt   or  "file\?name.txt"
file\\\*name.txt   or  "file\*name.txt"
file\\\\name.txt   or  "file\\\name.txt"
file\ -\ name.txt  or  "file - name.txt"
file~name.txt
```

Unix のコマンドの例:

```
$ scpg3 user@server:file\\\*name.txt .
```

```
$ scpg3 user@server:file\ -\ name.txt .
```

Windows で **scpg3** コマンドを使用する場合、上記の Unix のファイル名を以下のフォーマットで入力します。

```
"file|name.txt"
file-\"name\".txt     (Note that Windows requires \ to escape the " character)
"file~?name.txt"
"file~*name.txt"
file~\name.txt
"file - name.txt"
file~~name.txt
```

このようにすると、 **scpg3** コマンドがクォーテーション・マークを除いた文字列をパラメータとして受け取るように、オペレーティング・システムはクォーテーション・マーク ("") を解釈します。

Windows のコマンドの例:

```
> scpg3 user@server:"file~*name.txt" filename.txt
```

```
> scpg3 user@server:"file - name.txt" .
```

環境変数

scpg3 は以下の環境変数を使用します。

SSH_SFTP_CHECKSUM_MODE `=yes|no|md5|sha1|md5-force|sha1-force|checkpoint`

チェックサムを比較するための設定を定義します。　　使用できる値の詳細については、**checksum** を参照してください。

SSH_SFTP_SHOW_BYTE_COUNT `=yes|no`

この変数が `yes` に設定されている場合、ファイル転送に成功した後、転送されたバイト数が表示されます。また、転送元ファイルと転送先ファイルの名前も表示されます。デフォルトは `no` です。

SSH_SFTP_STATISTICS `=yes|no|simple`

この変数が `yes` に設定されている場合 (デフォルト)、ファイルの転送中に通常の進捗バーが表示されます。 `no` に設定されている場合、進捗バーは表示されません。 `simple` に設定されている場合は、ファイル転送後に統計データが表示されます。

終了値

scpg3 は、操作の結果に基づいて以下の値を返します。

```
0     Operation was successful.
1     Internal error.
2     Connection aborted by the user.
3     Destination is not a directory, but a directory was specified by the user.
4     Connecting to the host failed.
5     Connection lost.
6     File does not exist.
7     No permission to access file.
8     Undetermined error from sshfilexfer.
11    Some non-fatal errors occured during a directory operation.
101   Wrong command-line arguments specified by the user.
```

例

ローカル・システムからリモートの Unix システムにファイルをコピーする場合:

```
$ scpg3 localfile user@remotehost:/dst/dir/
```

ローカル・システムからリモートの Windows システムにファイルをコピーする場合:

```
$ scpg3 localfile user@remotehost:/C:/dst/dir/
```

リモート・システムからローカル・ディスクにファイルをコピーする場合:

```
$ scpg3 user@remotehost:/src/dir/srcfile /dst/dir/dstfile
```

`ssh-broker-config.xml` **ファイルで定義されている接続プロファイルを使用して、あるリモート・システムから別のリモート・システムにファイルをコピーする場合:**

```
$ scpg3 profile1:/src/dir/srcfile profile2:/dst/dir/dstfile
```

sftpg3

sftpg3 — Secure Shell ファイル転送クライアント - Generation 3

書式

`sftpg3` [`options` **...**]
[`profile` | [`user@`] `host` [`#port`] | [`:path`] | **sftp://** [[`user`] [`:password`] @] `host` [`:port`] [`/path`]]

説明

sftpg3 (Windows では **sftpg3.exe**) は、ネットワーク上でのセキュア・ファイル転送に使用できる FTP ライクなクライアントです。 **sftpg3** は、Secure Shell バージョン 2 プロトコルを使用したセキュアな転送を提供するために **ssh-broker-g3** を起動します。 **ssh-broker-g3** は、認証のためにパスワードやパスフレーズが必要な場合にそれらを要求します。 **sftpg3** は、`ssh-broker-config.xml` ファイルで指定された設定を使用します。

対話的に起動すると、 **sftpg3** は SFTP コマンドを入力するためのプロンプトを表示します。また、実行するコマンドを記述したバッチ・ファイルを使って、 **sftpg3** を非対話的に起動することもできます。使用できるコマンドについては、「コマンド」を参照してください。

コマンドラインを使用して各種パラメータのデフォルト値を設定する代わりに、 **sftpg3** が起動するたびに実行される起動バッチ・ファイルでコマンドを定義できます。 デフォルトでは、 **sftpg3** は `ssh_sftp_batch_file` という名前のファイルを、Unix では `$HOME/.ssh2/`、Windows では `%APPDATA%\SSH\` というユーザ固有のディレクトリで探します。

sftpg3 にはローカルとリモートの 2 つの接続エンド・ポイントがあり、どちらも SFTP クライアント・ホスト以外のホストに接続できます。引数なしで起動した場合、ローカル・エンド・ポイントは SFTP クライアント・ホストのファイル・システムに接続され、リモート・エンド・ポイントは接続されていない状態になります。 SFTP クライアント・ホスト以外の接続ホストは、 **sftp-server** (または **sft-server-g3**) サブシステムが有効になっている Secure Shell バージョン 2 サーバを実行している必要があります。Tectia Server ではデフォルトで、 **sft-server-g3** が有効になっています。

リモート接続のエンド・ポイントは、 **sftpg3** コマンドの引数として直接指定するか、**sftpg3** が起動した後に **open** SFTP コマンドで指定できます。ローカル接続のエンド・ポイントは **lopen** SFTP コマンドで指定できます。

接続時には、 `ssh-broker-config.xml` ファイルで定義されている接続プロファイルの名前 (`profile`) か、リモート・ホストの IP アドレスまたは DNS 名、及びオプションでリモート・ユーザ名と Secure Shell サーバのポート ([`user@`] `host` [`#port`] [`:path`]) を指定できます。ユーザ名を指定しない場合は、ローカル・ユーザ名が使用されます。ポートを指定しない場合、デフォルトの Secure Shell ポート 22 が使用されます。

注意

sftpg3 で接続プロファイルを入力する場合、 Tectia Client は引数の形式によって異なる意味に解釈することに注意してください。指定された属性値に @ 記号がある場

合、 Tectia Client は常に `<username@hostname>`、つまりプロファイルではないと解釈します。

また、プロファイル名にドットがある場合 (例: `host.x.example.com`)、コマンドライン上でそのドットをエスケープする必要があります。 Unix では、 Windows では、代わりに `host~.x~.example~.com` と入力します。このようにしないと、プロファイル名がホスト名として扱われ、現在の Windows ユーザ名がログインに使用されます。

ファイル名の特殊文字については、「ファイル名のサポート」を参照してください。

オプション

以下のオプションがあります。

`-4`

すべての接続に関連する DNS の解決が IPv4 アドレスとして解決されることを定義します。

`-6`

すべての接続に関連する DNS の解決が IPv4 アドレスとして解決されることを定義します。

`-b buffer_size_bytes`

SFTP プロトコルの 1 回の読み取りまたは書き込み要求のための最大バッファ・サイズを定義します (デフォルト: `32768` バイト)。

1 つのファイルの転送中に並行して送信される SFTP プロトコルの読み取りまたは書き込み要求の最大数は、`-N` オプションで指定できます。

ストリーミング (`--streaming` を参照) が使用されている場合 (デフォルトでは `buffer_size_bytes` より大きなファイルを Tectia Server との間で転送する場合)、このオプションはバッファ・サイズの定義に使用されないことに注意してください。

`-B [ - | batch_file ]`

`-B -` オプションを使用すると、標準入力からの読み込みを可能とします。このオプションは **sftpg3** でプロセスを起動し標準入力パイプでリダイレクトしたい場合などに便利です。

`batch_file` の名前を属性として定義することで、与えられたファイルから SFTP コマンドをバッチ・モードで実行できます。このファイルには、許可された任意の SFTP コマンドを含めることができます。コマンドについては、「コマンド」を参照してください。

バッチ・モードを使用するには、事前にサーバのホスト鍵をクライアントに保存し、ユーザ認証に非対話的な方法 (ホストベース認証やパスフレーズなしの公開鍵認証など) をセットアップしておく必要があります。

`-C`

現在の接続で圧縮を無効にします。

`+C`

この特定の接続で zlib 圧縮を有効にします。

`-c, --ciphers= LIST`

許可された暗号を設定し、サーバに提供します。例:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている暗号名を表示するには、値として `help` を入力します。

`-D, --debug= LEVEL`

デバッグ・レベルを設定します。`LEVEL` は 0 から 99 までの数字で、99 はすべてのデバッグ情報を表示することを指定します。このオプションは、コマンドラインの最初の引数である必要があります。

注意

`-D` オプションは Unix でのみ適用されます。Windows では、接続ブローカーのデバッグ・オプションである `-D, -l` をこのコマンドライン・ツールの代わりに使用します。

注意

デバッグ・レベルは、 **scpg3** コマンドが接続ブローカーを起動するときにのみ設定できます。接続ブローカーがすでに実行されている場合、このオプションはコマンドに影響を与えません。

`-i FILE`

identification ファイルに定義された秘密鍵を公開鍵認証に使用することを定義します。

`-K, --identity-key-file= FILE`

指定された秘密鍵または証明書の鍵ファイルをユーザ認証に使用することを定義します。鍵ファイルのパスはコマンドで指定します。

ファイルが秘密鍵の場合は、その秘密鍵が読み込まれ、接続ブローカーの鍵ストアにすでに保持されている鍵と比較されます。鍵が既知ではない場合は、その鍵がデコードされ、鍵ストアに一時的に追加されます。ファイルが証明書であり、接続ブローカーが一致する秘密鍵を保持している場合は、その秘密鍵が使用されます。証明書と秘密鍵はどちらも、コマンドラインで複数の `-K` オプションを使用して指定できます。

`-N max_requests`

並行して送信される、SFTP プロトコルの読み取りまたは書き込み要求の最大数を定義します (デフォルト: 10)。

各読み取りまたは書き込み要求で使用されるバッファのサイズは、 `-b` オプションで指定できます。

この値は 1 つのファイルの転送に適用されます。並行して送信されるファイルの数を定義するために使用することはできません。

ストリーミング (`--streaming` を参照) が使用されている場合 (デフォルトでは `-b` オプションで指定した `buffer_size_bytes` より大きなファイルを Tectia Server との間で転送する場合)、このオプションは使用されません。

`-P port`

リモート・マシンのこの Secure Shell ポートに接続します (デフォルト: `22`)。

`-q, --quiet`

エラー、警告、及び情報メッセージの表示を抑止します。このオプションは、接続ブローカーの設定ファイルで行われた `quiet-mode` の設定を上書きします。

`-v, --verbose`

詳細モードを使用します (`-D 2` と同じ)。

`--aa, --allowed-authentications= METHODS`

許可されたユーザ認証の方法をを設定し、サーバに提供します。方法のカンマ区切りリストを指定します。例:

```
--allowed-authentications keyboard-interactive,password
```

現在サポートされている認証方法を表示するには、値として `help` を入力します。

`--any-alg`

サポートされている全ての暗号、 MAC、 鍵交換、 ホスト鍵及び公開鍵アルゴリズムが使用されることを許可します。

`--compressions= METHODS`

許可された圧縮方法を設定し、サーバに提供します。方法のカンマ区切りリストを指定します。

現在サポートされている圧縮方法を表示するには、値として `help` を入力します。

`--exclusive`

接続を試みるたびに新しい接続を開くことを定義します。これを定義しない場合、接続ブローカーは最近閉じた接続を再使用できます。

`--fips`

FIPS 暗号化ライブラリを使用してチェックサムを実行します。

`--hostkey-algorithms=` `algorithms`

許可されたホスト鍵アルゴリズムを設定し、サーバに提供します。ホスト鍵アルゴリズムのカンマ区切りリストを指定します。例:

```
--hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされているホスト鍵アルゴリズムを表示するには、値として `help` を入力します。

`--identity=` `ID`

秘密鍵の ID をユーザ認証に使用することを定義します。ID には、接続ブローカー内部の鍵番号、鍵のハッシュ、または鍵ファイル名のいずれかを指定できます。

`--identity-key-hash=` `ID`

ユーザ認証に使用する秘密鍵と、それに対応する公開鍵のハッシュを定義します。

`--identity-key-id=` `ID`

接続ブローカー内部の鍵番号をユーザ認証に使用することを定義します。

`--keep-alive=` `VALUE`

キープアライブ・メッセージを Secure Shell サーバに送信する頻度を定義します。値を秒単位で入力します。デフォルト値は 0 で、キープアライブ・メッセージが無効であることを意味します。

`--kexs=` `kexs`

許可された鍵交換（KEX）方式を設定し、サーバに提供します。鍵交換名のカンマ区切りリストを指定します。例:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

値 `any` は全てのサポートされているアルゴリズムを許可します。現在サポートされている鍵交換方法を表示するには、値として `help` を入力します。

`--kip`

キーボード・インタラクティブとパスワードを、ユーザ認証に許可する方法として定義します。以下と同じです。

```
--allowed-authentications keyboard-interactive,password
```

`--macs=` `LIST`

許可された MAC を設定し、サーバに提供します。MAC 名のカンマ区切りリストを指定します。例:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

値 any は全てのサポートされているアルゴリズムを許可します。現在サポートされている MAC 名を表示するには、値として help を入力します。

--password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM

パスワードまたはパスフレーズ (以下、パスワード) を要求する認証方法に対する応答として、クライアントが送信するユーザ・パスワードまたはパスフレーズを設定します。これは、パスワードで保護された証明書や公開鍵にも使用できます。

PASSWORD は、このオプションの引数として直接指定できます (推奨されません)。より良い代替手段は、パスワードを含むファイルへのパスを入力するか (--password=file://PASSWORDFILE)、パスワードを出力するプログラムまたはスクリプトへのパスを入力する (--password=extprog://PROGRAM) 方法です。

extprog:// オプションを使用してシェル・スクリプトを参照する場合は、そのスクリプトがユーザのシェルを定義し、実際のパスワードを出力することも確認してください。そのようになっていない場合、シェル・スクリプトに使用するシェルを特定できないため、実行されたプログラムは失敗します。たとえば、パスワードの文字列が my_password.txt という名前のファイルに定義されていて、bash シェルを使用する場合は、以下の行をスクリプトに含めます。

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

警告

コマンドラインでパスワードを指定することは、安全な選択肢ではありません。たとえば、マルチユーザ環境では、コマンドラインで直接指定されたパスワードはプロセス・テーブルから容易に復元できます。よりセキュアな認証方法をセットアップする必要があります。非対話的なバッチ・ジョブの場合、パスフレーズなしの公開鍵認証やホストベース認証を使用する方がよりセキュアです。最低でも、ファイルやプログラムを使ってパスワードを供給してください。

--plugin-path= PATH

プラグイン・パスを PATH に設定します。これは FIPS モードでのみ使用します。

--tcp-connect-timeout= VALUE

Secure Shell サーバへの TCP 接続を確立するときのタイムアウト時間 (秒単位) を定義します。タイムアウトの値を正の数で入力します。値を 0 (ゼロ) にすると、この機能が無効になり、代わりにデフォルトのシステム TCP タイムアウトが使用されます。

--template-profile profile

指定したプロファイルを接続に使用します。

--publickey-algorithms= PUBLICKEY_ALGORITHMS

選択された署名アルゴリズムのみが公開鍵認証に使用されることを許可します。例:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

値 any は全てのサポートされているアルゴリズムを許可します。現在サポートされている署名アルゴリズムを表示するには、値として help を入力します。

-l, --user= USERNAME

USERNAME は、アドレス文字列でユーザ名が指定されていない場合、ログオン時に使用されます。

-V, --version

プログラムのバージョンを表示して終了します。

-h, --help, -?

コマンドライン・オプションの要約を表示して終了します。

コマンド

sftpg3 がコマンドを受け入れる準備ができると、 sftp> プロンプトが表示されます。その後ユーザは、以下のコマンドを入力できます。

!	append	ascii	auto	binary
break	cd	chmod	close	continue
debug	delete	digest	echo	exit
get	getext	help	helpall	lappend
lcd	lchmod	lclose	ldelete	ldigest
lls	llsroots	lmkdir	localopen	locsite
lopen	lpwd	lreadlink	lrename	lrm
lrmdir	ls	lsite	lsroots	lsymlink
mget	mkdir	mput	open	pause
put	pwd	quit	readlink	rename
rm	rmdir	set	setext	setperm
sget	site	sput	sunique	symlink
type	verbose			

! [command] [arguments]

ローカル・マシン上で対話型シェルを起動します。command が指定されている場合は、実行されるコマンドとしてそれが使用されます。コマンドによっては、オプションの arguments を指定できます。

append [-u, --unlink-source] [--streaming] [--force-lower-case] [--statistics] [--summary-display] [--summary-format] [--progress-display] [--progress-line-format] [--progress-line-interval] srcfile [dstfile]

指定されたローカル・ファイルをリモート・ファイルに追加します。グロブは使用できません。

オプション:

`-u, --unlink-source`

ファイル転送後、転送元ファイルを削除します。

`--streaming [=yes | no | force | ext ]`

ファイル転送にストリーミングを使用します (サーバがサポートしている場合)。`buffer_size_bytes` より小さいファイルはストリーミングで転送されません。小さなファイルには `force` を使用します。デフォルト: `yes`

MVS データ・セットへの直接アクセスを可能にするには、z/OS ホストで `ext` を使用します。その他の環境では小さなファイルの転送が遅くなることがあるため、主にメインフレームのデータ・セット転送に使用する場合にのみ、このオプションを使用してください。

チェックサムが計算される場合、ファイル転送はステージングを使用し、ストリーミングは除外されるため、`--streaming=ext` オプションには `--checksum=no` オプションも必要です。

`--force-lower-case`

転送先ファイル名が小文字に変換されます。

`--statistics`、`--summary-display`、`--summary-format`、`--progress-display`、`--progress-line-format`、及び `--progress-line-interval` のオプションのセマンティクスは **get** と同じです。

`ascii [ -s ] [ remote_nl_conv ] [ local_nl_conv ]`

ascii コマンドは転送モードをアスキーに設定します。

z/OS の Tectia と他のホスト間の転送では、アスキーと EBCDIC の自動変換も可能になります。デフォルトでは、ISO8859-1 と IBM-1047 のコード・セット間で変換されます。ファイルの転送には `LINE` 形式が使用されます。**site** 及び **lsite** コマンドを使用すると、値を変更できます。

ascii コマンドをオプションとともに入力した場合、転送モードはアスキーに設定されませんが、転送されるファイルで使用される改行規則には影響します。また、ホスト・プロファイルでホスト・タイプを指定することで、サーバの改行規則を設定することもできます。詳細については、「`profiles` エレメント」の `host-type` 属性を参照してください。

オプション:

`-s`

現在の改行規則を表示します。システムにより、以下の改行記号を使用します。

```
dos:    CRLF (\r\n, 0x0d 0x0a)
```

```
mac:    CR (\r, 0x0d)
mvs:    NEL (\n, 0x15)
unix:   LF (\n, 0x0a)
```

`remote_nl_conv local_nl_conv`

この構文は、リモートとローカルの改行規則を定義するために使用できます。`local_nl_conv` オプションはローカル・エンドで動作しますが、通常は正しいローカルの改行規則がすでにコンパイルされていることに注意してください。

下位の転送レイヤのために改行規則に関するヒントを設定できます。その場合はデフォルトで、サーバから与えられた実際の改行規則を使用しようとします。または、改行モードを強制することもできます。

改行規則に関するヒントを設定するには、`remote_nl_conv` 及び `local_nl_conv` オプションで`dos`、 `unix`、及び `mac` の値を使用します。これらの設定は、リモートの SSH サーバが SFTP クライアントに改行情報を自動的に提供しない場合に使用されます。例:

```
sftp> ascii
 File transfer mode is now ascii.
sftp> ascii unix dos
 Newline conventions updated.
```

改行規則を強制するには、 `force-dos`、 `force-unix`、及び `force-mac` の値を使用します。これらの設定は、リモート SSH サーバが SFTP クライアントに提供する内容に関係なく、改行モードを強制します。

```
sftp> ascii
 File transfer mode is now ascii.
sftp> ascii force-unix force-dos
 Newline conventions updated.
```

どちらかのオプションを `ask` に設定することもできます。その場合、 **sftpg3** は必要に応じて改行規則の入力を求めます。

`auto`

ファイル転送モードはファイルの拡張子から自動的に選択されます。

`binary`

ファイルはバイナリ・モードで転送されます。

`break`

バッチ・ファイルの実行を中断します。バッチ・ファイルの実行は **continue** コマンドで継続できます。

`bye`

アプリケーションを終了します。

`cd directory`

現在のリモート作業ディレクトリを変更します。

`chmod [ -R ] [ -f ] [ -v ] OCTAL-MODE [ file ...]`

,

`chmod [ -R ] [ -f ] [ -v ] [ ugoa ] [ +-= ] [ rwxs ] [ file ...]`

Unix のファイルでは、指定されたファイルのファイル・パーミッションをビット・パターン `OCTAL-MODE` に設定するか、またはシンボリック・モード `[ugoa] [+-=] [rwxs]` に従ってファイル・パーミッションを変更します。

オプション:

`-R`

ファイルやディレクトリを再帰的に変更します。

`-f`

サイレント・モードを使用します (エラー・メッセージが抑制されます)。

`-v`

詳細モードを使用します (処理されたすべてのファイルを一覧表示します)。

`close`

リモート接続を終了します。

`continue`

中断されたバッチ・ファイルの実行を継続します。

`debug [ disable | no | debuglevel ]`

デバッグを無効または有効にします。 `disable` または `no` を指定すると、デバッグは無効になります。それ以外の場合は、コマンドライン・オプション `-D` と同様に、`debuglevel` にデバッグ・レベル文字列を設定します。

`delete [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

`file` で指定されたファイルまたはディレクトリの削除を試みます。オプションは **rm** と同じです。

`digest [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

ファイル・データに対して MD5、 SHA-1 または SHA-2 ダイジェストを計算します。このダイジェストはディスク上のデータに対して計算されます。コードまたは行の区切り文字の変換が有効な場合、それらはダイジェストを計算する際に無視されます。

オプション:

`-H, --hash=[alg]`

ハッシュ・アルゴリズムを指定します。サーバによりサポートが異なります。 `md5`、`sha1`、 `sha256` および `sha512` が一般的にサポートされます (デフォルト: `sha1`)。

`-o, --offset=OFFSET`

ファイル・オフセット `OFFSET` から読み取りを開始します。

`-l, --length=LENGTH`

ファイル・データを `LENGTH` バイトを読み取ります。

`echo Text to be echoed.`

テキストをエコーします。このコマンドは、バッチ・モードでテキストをバッチ・ログに表示するために使用できます。

`exit`

アプリケーションを終了します。

`get [-p, --preserve-attributes] [-u, --unlink-source] [-I, --interactive] [--overwrite] [--checksum] [-W, --whole-file] [--checkpoint] [--streaming] [--force-lower-case] [--prefix] [--statistics] [--summary-display] [--summary-format] [--progress-display] [--progress-line-format] [--progress-line-interval] [--max-depth=] file ...`

指定されたファイルをリモート・エンドからローカル・エンドに転送します。デフォルトでは、ディレクトリはその内容とともに再帰的にコピーされますが、この設定は、接続ブローカーの設定にある SFTP 互換モードの設定で変更できます (`ssh-broker-config.xml` の `sftpg3-mode`)。現在設定されている SFTP 互換モードを表示するには、以下のコマンドを実行します。

```
sftp> help get
```

現在設定されている互換モードは、コマンド **get** のヘルプの先頭に表示されます。

SFTP 互換モードには以下のオプションがあります。

`tectia`

sftpg3 クライアントは、カレント・ディレクトリとそのすべてのサブディレクトリから再帰的にファイルを転送します。

`ftp`

get コマンドは **sget** として実行されます。つまり、1 つのファイルが転送され、サブディレクトリはコピーされません。

`openssh`

指定されたディレクトリから通常のファイルとシンボリック・リンクのみがコピーされます。サブディレクトリはコピーされません。それ以外の場合、 **get** コマンドのセマンティクスは変更されません。

オプション:

`-p, --preserve-attributes`

送信元と送信先が両方とも Unix ファイル・システム上 (z/OS USS を含む) にある場合、ファイルのパーミッションとタイムスタンプを保持します。転送元または転送先のいずれかが Windows 上にある場合、タイムスタンプは保持しますが、ファイルのパーミッションは保持しません。転送先が z/OS MVS 上にある場合、何も保持されません。

`-u, --unlink-source`

ファイル転送後、転送元ファイルを削除します。空になったディレクトリも削除されます (移動モード)。

`-I, --interactive`

既存の転送先ファイルを上書きするかどうかの確認を求めます (バッチ・モードでは動作しません)。

`--overwrite [ =yes | no ]`

既存の転送先ファイルを上書きするかどうかを決定します (デフォルト: `yes`)。

`--checksum [ =yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]`

ファイル転送を再開できるファイル内のポイントを特定するために、MD5 または SHA-1 チェックサム、あるいは独立したチェックポイントデータベースを使用します。`buffer_size_bytes` より小さいファイルはチェックされません。小さなファイルには `md5-force` または `sha1-force` を使用します (デフォルト: `yes`。つまり、MD5 チェックサムを使用します)。大きなファイルを 1 つずつ転送する場合は、チェックポイントを使用します。

`-W, --whole-file`

インクリメンタル・チェックを試みません。デフォルトでは (このオプションが指定されていない場合)、インクリメンタル・チェックが試みられます。このオプションは `--checksum` オプションと一緒にしか使用できません。

`--checkpoint=s <seconds>`

チェックポイントの更新間隔を時間で指定します (デフォルト: `10` 秒)。このオプションは、 `--checksum=checkpoint` のときにのみ使用できます。

`--checkpoint=b <bytes>`

チェックポイント更新間隔をバイトで指定します (デフォルト: 10 MB)。このオプションは、 `--checksum=checkpoint` のときにのみ使用できます。

`--streaming [ =yes | no | force | ext ]`

サーバがサポートしている場合、ファイル転送にストリーミングを使用します。`buffer_size_bytes` より小さいファイルはストリーミングで転送されません。小さなファイルには `force` を使用します。デフォルト: `yes`

MVS データ・セットへの直接アクセスを可能にするには、z/OS ホストで `ext` を使用します。その他の環境では小さなファイルの転送が遅くなることがあるため、主にメインフレームのデータ・セット転送に使用する場合にのみ、このオプションを使用してください。

チェックサムが計算される場合、ファイル転送はステージングを使用し、ストリーミングは除外されるため、 `--streaming=ext` オプションには `--checksum=no` オプションも必要です。

拡張ストリーミングを有効にする別の方法として、 `SSH_SFTP_STREAMING_MODE=ext` 及び `SSH_SFTP_CHECKSUM_MODE=no` を環境変数として定義できます。

`--force-lower-case`

転送先ファイル名が小文字に変換されます。

`--max-depth= VALUE`

ディレクトリを再帰的にコピーするかどうかを定義します。指定できる値は以下の通りです。

`0` - 無限再帰。ディレクトリはその内容とともに再帰的にコピーされます。

`1` - 指定されたディレクトリのファイルのみをコピーし、サブディレクトリからはコピーしません。

`2-n` - 指定されたディレクトリ・レベル数から再帰的にファイルをコピーします。ここで `n` は、システム固有の最大値を意味します。

このコマンドライン・オプションは、接続ブローカーの設定のエレメント `sftpg3-mode` で設定されている再帰の深さ、及び/または環境変数 `SSH_SFTP_CMD_GETPUT_MODE` を使用して設定された再帰の深さを上書きします。

`--prefix= PREFIX`

ファイルが転送されている間にのみ、ファイル名にプレフィックスを付加します。ファイル転送に成功すると、付加されたプレフィックスは削除されます。

z/OS では、MVS データ・セット名にプレフィックスを適用する場合、ハイレベル修飾子 (HLQ) の後にプレフィックスがデフォルトで挿入されます。プレフィックスを独立した修飾子にする場合は、プレフィックスの末尾にドットを含めます。

```
--prefix=PREFIX.
```

`--statistics [ =no | yes | simple ]`

注意

リリース 6.1.5 では `--statistics` オプションの動作が変更され、 `--statistics-format` オプションが削除されました。これらの代わりに、新たに `--summary-display` オプションと `--summary-format` オプションを使用します。

`--statistics` オプションは、ファイル転送操作の後に表示される統計データのスタイルを選択します。 `--statistics` と `--summary-display` は併用しないでください。

`--statistics` オプションは以下の値を取ります。

`no` - 統計データは作成されません。

`yes` - ファイル転送中に進捗バーを表示します。これがデフォルトです。出力の例を以下に示します。

```
sftp> get --statistics="yes" sourcefile
sourcefile               |  127MB |  42.9MiB/s | TOC: 00:00:03 | 100%
```

`simple` - ファイル転送が終了すると、1 行の簡単な統計が表示されます。例:

```
sftp> get --statistics=simple testfile
sourcefile   | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`--summary-display [ =no | yes | simple | bytes ]`

ファイル転送操作の後に表示される、ファイル転送サマリ・データのスタイルを選択します。サマリの表示とともに、進捗バーのデータもデフォルトで表示されます。

`--summary-display` と `--statistics` は併用しないでください。

`--summary-display` オプションは以下の値を取ります。

`no` - サマリ・データは作成されません。これがデフォルトです。

`yes` - 詳細なサマリ・データが作成されます。 `summary-format` オプションで内容を設定できます。デフォルトでは、以下の内容がサマリに表示されます。

```
Default settings:                   Render for example this:
"Source: %c:%g\n"                   user@host1#22:/path/to/source/file
"Source parameters: %e\n"           X=TEXT, C=ISO8859-1,D=IBM.1047
"Destination: %C:%G\n"              user@host2#22:/path/to/destination/file
"Destination parameters: %E\n"      NONE
"File size: %s bytes\n"             123456 bytes
"Transferred: %t bytes\n"           123456 bytes
"Rate: %RB/s\n"                     345kiB/s
"Start: %xy-%xt-%xd %xh:%xm:%xs\n"  2010-01-26 13:10:56
"Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n"   2010-01-26 13:23:30
"Time: %y\n"                        00:12:34
```

`simple` - 1 行の簡単なサマリが表示されます。例:

```
sftp> get --summary-display=simple sourcefile
sourcefile  | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`bytes` - 転送されたバイト数を報告する基本的な統計データが表示されます。例:

```
sftp> get --summary-display=bytes sourcefile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

`--summary-format= FORMAT_STRING`

サマリの形式と内容を選択します。このオプションは、 `--summary-display=yes` のときに使用できます。このオプションは `--statistics` と併用しないでください。

以下の定義を使用して、サマリの内容を選択します。

```
%c  - source connection: user@host#port or profile
%C  - destination connection: user@host#port or profile
%D* - current date
%e  - source parameters (file transfer and data set parameters)
%E  - destination parameters (file transfer and data set parameters)
%f  - source file name
%F  - destination file name
%g  - /path/to/source/file
%G  - /path/to/destination/file
%k  - compression done ("zlib" or "none")
%p  - transfer percentage
%q  - transfer rate in bit/s
%Q  - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r  - transfer rate in bytes/s
%R  - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%s  - file size in bytes
%S  - file size as "XXyB" (B, kiB, MiB or GiB)
%t  - transfer size in bytes
%T  - transfer size as "XXyB" (B, kiB, MiB or GiB)
%x* - start date
%X* - end date
%y  - elapsed time
%Y  - time remaining
%z  - ETA or TOC, if transfer has finished
%Z  - string "ETA" or "TOC", if transfer has finished

Where * is one of the following:

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

Other special characters in format strings are:
```

```
\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash
```

`--progress-display [ =no | bar | line ]`

ファイル転送中の進捗状況を表示するモードを選択します。デフォルトは `bar` で、進捗バーが表示されます。 `line` オプションは、 `--progress-line-format` オプションで行われた設定に従って進捗情報を表示します。

このオプションは `--statistics` と併用しないでください。

`--progress-line-format= FORMAT_STRING`

進捗ラインに表示する情報を選択します。このオプションは、 `--progress-display=line` のときに使用できます。

このオプションは `--statistics` と併用しないでください。

上記の **get** コマンドのオプション `--summary-format` で記述した定義を使用して、進捗ラインの内容を選択します。

`--progress-line-interval= seconds`

ライン・モードでの進捗情報の更新頻度を定義します。間隔は秒単位で指定します。デフォルトは 60 秒です。

このオプションは `--statistics` と併用しないでください。

`getext`

自動転送モードでアスキーとなる拡張子を表示します。

`lappend [ options ...] srcfile [ dstfile ]`

append と同じですが、 指定されたリモート・ファイルをローカル・ファイルに追加します。

`lcd directory`

現在のローカル作業ディレクトリを変更します。

`lchmod [ -R ] [ -f ] [ -v ] OCTAL-MODE [ file ...]`

,

`lchmod [ -R ] [ -f ] [ -v ] [ ugoa ] [ +-= ] [ rwxs ] [ file ...]`

chmod と同じですが、 ローカル・ファイル上で動作します。

`lclose`

ローカル接続を終了します。

`ldelete [ options ... ] file ...`

delete と同じですが、 ローカル・ファイル上で動作します。

`ldigest [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

digest と同じですが、 ローカル・ファイル上で動作します。

`lls [ -R ] [ -l ] [ -S ] [ -r ] [ -p ] [ -z|+z ] [ file ...]`

ls と同じですが、 ローカル・ファイル上で動作します。

`llsroots`

lsroots と同じですが、 ローカル・ファイル上で動作します (ローカル・エンドが VShell サーバに開かれている場合)。

`lmkdir directory`

mkdir と同じですが、 ローカル・ファイル上で動作します。

`localopen [ user@ ] hostname [ #port ] [ -l ] [ --user= USERNAME ]`

lopen と同じです。

`lopen [ user@ ] hostname [ #port ] [ -l ] [ --user= USERNAME ]`

ローカル・エンドをホスト `hostname` へ接続するよう試みます。これが成功すると、 **lls** および関連するコマンドはそのホストのファイル・システム上で動作するようになります。

オプション:

`-l`

ローカル・エンドを SFTP クライアント・ホストのファイル・システムに接続します (サーバを必要としません)。これは、 **lopen** コマンドが与えられていない場合のデフォルトの状態でもあります。

`--user`

接続に使用するユーザを `USERNAME` と定義します。

`locsite [ none | name1=value1 name2=value2 ... ]`

site と同じですが、 ローカル・ファイルとデータ・セット上で動作します。

`lpwd`

現在のローカル作業ディレクトリの名前を表示します。

`lreadlink path`

readlink と同じですが、 ローカル・ファイル上で動作します。

`lrename oldfile newfile`

rename と同じですが、 ローカル・ファイル上で動作します。

`lrm [ options ...] file ...`

rm と同じですが、 ローカル・ファイル上で動作します。

`lrmdir directory`

rmdir と同じですが、 ローカル・ファイル上で動作します。

`ls [ -R ] [ -l ] [ -S ] [ -r ] [ -p ] [ -z|+z ] [ file ...]`

リモート・サーバ上のファイル名を一覧表示します。ディレクトリの場合は、内容が一覧表示されます。引数が与えられない場合は、現在の作業ディレクトリの内容が一覧表示されます。

オプション:

`-R`

ディレクトリ・ツリーが再帰的に一覧表示されます。デフォルトでは、引数で指定されたディレクトリのサブディレクトリにはアクセスしません。

`-l`

パーミッション、所有者、サイズ、及び変更時刻も表示されます (ロング・フォーマット)。

`-S`

ファイル・サイズに基づいて並べ替えが行われます (デフォルト: アルファベット順に並べ替え)。

`-r`

並び順が逆になります。

`-p`

一度に 1 ページ分の一覧のみ表示されます。

`-z`

クライアントがロング・フォーマットの出力を生成します。

`+z`

サーバから提供されたロング・フォーマットの出力が利用可能であれば、それが使用されます (オプション `-l` のエイリアス)。

`lsite [ none | name1=value1 name2=value2 ... ]`

site と同じですが、 ローカル・ファイルとデータ・セット上で動作します。

`lsroots`

サーバの仮想ルートをダンプします。(これは VShell の拡張機能です。これがないと、VShell サーバのファイル・システム構造を把握できません)

`lsymlink targetpath linkpath`

symlink と同じですが、 ローカル・ファイル上で動作します。

`mget [ options ...] file ...`

get と同義です。

`mkdir directory`

`directory` で指定されたディレクトリの作成を試みます。

`mput [ options ...] file ...`

指定されたファイルをローカル・エンドからリモート・エンドに転送します。オプションとセマンティクスは **get** と同じです。**put** と同義です。

`open [ user@ ] hostname [ #port ] [ -l ] [ --user= USERNAME ]`

リモート・エンドをホスト `hostname` へ接続するよう試みます。

オプション:

`-l`

リモート・エンドを SFTP クライアント・ホストのファイル・システムに接続します (サーバを必要としません)。

`--user`

接続に使用するユーザを `USERNAME` と定義します。

`pause [ seconds ]`

バッチ・ファイルの実行を `seconds` 秒間、または `seconds` が指定されていない場合は **ENTER** キーが押されるまで一時停止します。

`put [ options ...] file ...`

指定されたファイルをローカル・エンドからリモート・エンドに転送します。オプションとセマンティクスは **get** と同じです。

`pwd`

現在のリモート作業ディレクトリの名前を表示します。

`quit`

アプリケーションを終了します。

`readlink path`

`path` がシンボリック・リンクの場合、リンクの指し示す先を表示します。

`rename oldfile newfile`

`oldfile` の名前を `newfile` に変更しようと試みます。`newfile` がすでに存在する場合、ファイルは変更されません。

`rm [ -I, --interactive ] [ -r, --recursive ] file ...`

`file` で指定されたファイルまたはディレクトリの削除を試みます。

オプション:

`-I, --interactive`

ファイルまたはディレクトリを削除するかどうかの確認を求めます (バッチ・モードでは動作しません)。

`-r, --recursive`

ディレクトリが再帰的に削除されます。

`rmdir directory`

`directory` で指定されたディレクトリの削除を試みます。このコマンドは、ディレクトリが空で、サブディレクトリも存在しない場合にのみディレクトリを削除します。

`set [ defaults | [ --commands=name1,name2,... exit-value=VALUE ] | option1=value1 option2=value2 ... ]`

各種パラメータのデフォルト値を設定します。 `set` コマンドには以下のオプションを指定できます。

`defaults`

システムのデフォルトとなるパラメータを設定します。

`checksum [ =yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]`

ファイル転送を再開できるファイル内のポイントを特定するために、MD5 または SHA-1 チェックサム、あるいは独立したチェックポイントデータベースを使用します。`buffer_size_bytes` より小さいファイルはチェックされません。小さなファイルには `md5-force` または `sha1-force` を使用します。デフォルトは `md5` です (z/OS のデフォルトは `no` です)。大きなファイルを 1 つずつ転送する場合は、チェックポイントを使用します。

`compatibility-mode [ =tectia | ftp | openssh ]`

ファイル転送で使用する再帰モードを定義します。

`tectia`

sftpg3 クライアントは、カレント・ディレクトリとそのすべてのサブディレクトリから再帰的にファイルを転送します。これがデフォルトのモードです。

`ftp`

1 つのファイルが転送され、サブディレクトリはコピーされません。

`openssh`

指定されたディレクトリから通常のファイルとシンボリック・リンクのみがコピーされます。サブディレクトリはコピーされません。

`compressions [ =none | zlib ]`

ファイル転送で圧縮を使用するかどうかを定義します。

`none`

圧縮は使用されません。これがデフォルトです。

`zlib`

ファイル転送時に zlib 圧縮を有効にします。

`exit-value= VALUE`

バッチ・モードでエラーが発生した場合の **sftpg3** の終了値を定義します。値は 0 ～ 255 の間でなければなりません。 `exit-value` に 0 以外を設定し、 `--commands` パラメータを使用しない場合、最初のエラーが発生した時点でバッチ実行が終了します。

例 1: このバッチ・ジョブの rename コマンドが失敗すると、 **sftpg3** は停止して終了値 "6" を返します。

```
open user@host
set exit-value=6
rename file file2
<next command in batch job>
quit
```

例 2: 特定のコマンドが失敗する可能性を無視し、実際の操作の結果とは無関係に終了値 "0" を返すようにする場合は、コマンドの後に `set exit-value=0` を使用します。このバッチ・ジョブの例は、ファイル名の変更に失敗する可能性を無視します。

```
open user@host
rename file file2
set exit-value=0
<next command in batch job>
```

```
quit
```

`--commands= name1,name2,... exit-value= VALUE`

このオプションは、指定されたコマンドのいずれかに失敗した場合に **sftpg3** のバッチ・ジョブを中止させます。このオプションで指定されていないコマンドが失敗した場合、バッチ・ジョブの実行は継続され、 `exit-value` で定義されている値がバッチ・ジョブの終了値に設定されます。 `exit-value=0` の場合、失敗したコマンドの終了値が返されることに注意してください。

例 3: sftpg3 がバッチ・モードで実行されている場合、 **put**、 **get**、または **ls** コマンドが失敗すると実行が中断されます。その他のコマンド (後述の例外を除く) が失敗しても、バッチ・ファイルの終わりまで実行は継続されます。いずれの場合も "3" が返されます。

```
set --commands=put,get,ls exit-value=3
```

例 4: sftpg3 がバッチ・モードで実行されている場合、 **put** または **get** コマンドが失敗したときに実行が中断されます。その他のコマンド (後述の例外を除く) が失敗しても、実行は継続されます。どのような場合でも、最後に失敗したコマンドの元の終了値が返されます。

```
set --commands=put,get exit-value=0
```

例外: `--commands` オプションで特定のコマンドに `exit-value` を設定した場合、以下のような場合にもバッチ・ジョブの実行が中断されます。

- **cd** コマンドのエラー
- 無効なコマンド
- `Authentication failed` エラー
- `Unable to connect to server` エラー
- `Connection aborted` エラー

デフォルト (`set defaults`) では、エラーが発生した場合でも **sftpg3** は停止せずに実行を継続し、最後のエラー・メッセージを返します。

`--commands` で追加された無効なコマンドは無視されます。

`overwrite [ =yes | no ]`

既存の転送先ファイルを上書きするかどうかを決定します (デフォルト: `yes`)。

`progress-display [ =bar | line | no ]`

ファイル転送中の進捗状況を表示するモードを選択します。デフォルトは `bar` で、進捗バーが表示されます。 `line` オプションは、 `progress-line-format` オプションで行われた設定に従って進捗情報を表示します。 `no` オプションは進捗状況の表示を無効にします。

progress-line-format= FORMAT_STRING

進捗ラインに表示する情報を選択します。このオプションは --progress-display=line のときに使用します。次のコマンドのコンテンツ・オプションの定義を参照してください: get --progress-line-format。

progress-line-interval= seconds

ライン・モードでの進捗情報の更新頻度を定義します。間隔は秒単位で指定します。デフォルトは 60 秒です。

summary-display [=no | yes | simple | bytes]

ファイル転送操作の後に表示される、ファイル転送サマリ・データのスタイルを選択します。サマリの表示とともに、進捗バーのデータもデフォルトで表示されます。このオプションは --statistics と併用しないでください。

次のコマンドで説明されているオプションを参照してください: get --summary-display

summary-format= FORMAT_STRING

サマリの形式と内容を選択します。このオプションは、 --summary-display=yes のときに使用できます。このオプションは --statistics と併用しないでください。

次のコマンドのコンテンツ・オプションの定義を参照してください: get --summary-format

streaming [=yes | no | force | ext]

サーバがサポートしている場合、ファイル転送にストリーミングを使用します。buffer_size_bytes より小さいファイルはストリーミングで転送されません。小さなファイルには force を使用します。デフォルト: yes

MVS データ・セットへの直接アクセスを可能にするには、z/OS ホストで ext を使用します。その他の環境では小さなファイルの転送が遅くなることがあるため、主にメインフレームのデータ・セット転送に使用する場合にのみ、このオプションを使用してください。

チェックサムが計算される場合、ファイル転送はステージングを使用し、ストリーミングは除外されるため、 streaming=ext オプションには checksum=no オプションも必要です。

setext [extension ...]

自動転送モードでアスキーとなるファイル拡張子を設定します。ファイル拡張子には通常の zsh-fileglob 正規表現を使用できます。

setperm fileperm [:dirperm]

アップロードする際のデフォルトのファイル及びディレクトリのパーミッション・ビットを設定します。(既存のファイルまたはディレクトリのパーミッションを保持するには、 fileperm の前に p を付けます)

`sget [ options ... ] srcfile [ dstfile ]`

指定された 1 つのファイルをリモート・エンドからローカル・エンドに、`dstfile` で定義されているファイル名で転送します。ディレクトリはコピーされません。ワイルドカードは使用できません。オプションは **get** と同じです。

`site [ none | name1=value1 name2=value2 ... ]`

リモート・ホストのファイルおよびデータ・セットのパラメータを設定します。パラメータは 1 つずつ入力することも、複数のパラメータをスペースやカンマで区切って入力することもできます。長いパラメータと略語の両方が使用できます。引数なしで実行すると、 **site** コマンドは入力されたパラメータのリストを出力します。 `none` を設定すると、すべてのパラメータがリセットされます。

使用できるパラメータは以下の通りです。

- `AUTOMOUNT=YES|NO|IMMED`
- `[NO]AUTOMOUNT|[NO]AUTOM`
- `AUTORECALL=YES|NO`
- `[NO]AUTORECALL|[NO]AUTOR`
- `BLKSIZE|B|BLOCKSI= size`
- `BLOCKS|BL`
- `CONDDISP|CO=CATLG|UNCATLG|KEEP|DELETE`
- `CYLINDERS|CY`
- `DATACLAS|DA= class`
- `DATASET_SEQUENCE_NUMBER|SEQNUM= number`
- `DEFER|DE=YES|NO`
- `[NO]DEFER|DE`
- `DIRECTORY_SIZE|M|DI|DIRSZ= size`
- `EXPIRY_DATE|EXPDT= yyddd|yyyyddd`
- `FILE_STATUS|STATUS=NEW|MOD|SHR|OLD`
- `FILETYPE|FILET=SEQ|JES`
- `FIXRECFM|FI= length`
- `JOB_ID|JESID= ID`
- `JOB_OWNER|JESO= name`

- JOBNAME|JESJOB= name
- KEYLEN|KEYL= length
- KEYOFF|KEYO= offset
- LABEL_TYPE|LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL
- LIKE= like
- LRECL|R|LR= length
- MGMTCLAS|MG= class
- NORMDISP|NOR=CATLG|UNCATLG|KEEP|DELETE
- PRIMARY_SPACE|PRI= space
- PROFILE|P|PROF= profile
- RECFM|O|REC= recfm
- RECORD_TRUNCATE|U|TRUN=YES|NO
- [NO]TRUNCATE|[NO]TRU|[NO]TRUN
- RETENTION_PERIOD|RET= days
- SECONDARY_SPACE|SE|SEC= space
- SIZE|L= size
- SPACE_RELEASE|RLSE=YES|NO
- SPACE_UNIT|SU=BLKS|TRKS|CYLS|AVGRECLEN
- SPACE_UNIT_LENGTH|SUL= length
- STAGING|S|STAGE=YES|NO
- STORCLAS|ST= class
- SVC99_TEXT_UNITS|SVC99= string
- TRACKS|TR
- TRAILING_BLANKS|TRAIL=YES|NO
- [NO]TRAILINGBLANKS|[NO]TRAI|[NO]TRAIL
- TRANSFER_CODESET|C|CODESET= codeset
- TRANSFER_FILE_CODESET|D|FCODESET= codeset
- TRANSFER_FILE_LINE_DELIMITER|J|FLDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL

- TRANSFER_FORMAT|F|FORMAT=LINE|STREAM|RECORD
- TRANSFER_LINE_DELIMITER|I|LDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL
- TRANSFER_MODE|X|MODE=BIN|TEXT
- TRANSFER_TRANSLATE_DSN_TEMPLATES|A|XDSNT= templates
- TRANSFER_TRANSLATE_TABLE|E|XTBL= table
- TYPE|T=PS|PO|PDS|POE|PDSE|GDG|HFS|VSAM|ESDS|KSDS|RRN
- UNIT|UN= unit
- UNIT_COUNT|UC|UNC= number
- UNIT_PARALLEL|UNP=YES|NO
- VOLUME_COUNT|VC|VOLCNT= number
- VOLUMES|VO|VOL= vol1+vol2+...

sput [options ...] srcfile [dstfile]

指定された 1 つのファイルをローカル・エンドからリモート・エンドに、dstfile で定義されているファイル名で転送します。ディレクトリはコピーされません。ワイルドカードは使用できません。オプションは **get** と同じです。

sunique [on] [off]

一意の名前を持つファイルを保存します。オプションを指定しない場合、コマンドは 'sunique' の状態を切り替えます。

転送されたファイルの中に同じ名前のファイルが複数ある場合、この機能は、繰り返されるファイル名の末尾に連番を付加します (例: file.name、file.name1、及び file.name2)。

symlink targetpath linkpath

targetpath を指すシンボリック・リンク linkpath を作成します。

type [ascii | auto | binary | default]

ファイル転送の種類を設定します。種類が指定されない場合は、現在のファイル転送の種類が表示されます。

ascii

アスキー・モードでファイルを転送します。詳細については、**ascii** を参照してください。

auto

自動モードでファイルを転送します。詳細については、**auto** を参照してください。

`binary`

バイナリ・モードでファイルを転送します。詳細については、**binary** を参照してください。

`default`

バイナリ・モードでファイルを転送します。サーバが Tectia Server for IBM z/OS の場合、サーバに追加のパラメータを指定しないことを除いて、このモードは `binary` と同じです。詳細については、**binary** を参照してください。

`verbose`

詳細モードを有効にします (**debug 2** コマンドと同じ)。詳細モードは、後から **debug disable** で無効にできます。

`help [ topic ]`

`topic` が指定されていない場合、使用できるトピックを一覧表示します。`topic` が指定されている場合、そのトピックに関して使用できるオンライン・ヘルプを表示します。

`helpall`

すべてのトピックについてオンライン・ヘルプを表示します。

コマンドの解釈

sftpg3 はコマンドライン上でバックスラッシュ (\) とクォーテーション・マーク ("") の両方を認識します。バックスラッシュはコマンドラインの解釈において、任意の文字の特別な意味を無視するために使用できます。バックスラッシュの後に続く文字が特別な意味を持たない場合でも、バックスラッシュは削除されます。

スペースを含むファイル名を指定する場合は、ファイル名をクォーテーション・マークで囲みます。

注意

get . 及び **put .** コマンドは、カレント・ディレクトリにあるすべてのファイルをダウンロードまたはアップグレードし、場合によっては、カレント・ディレクトリにあるファイルを上書きします。

sftpg3 は **chmod**、 **lchmod**、 **ls**、 **lls**、 **rm**、 **lrm**、 **get**、及び **put** コマンドでワイルドカード文字 (グロブ・パターンとも呼ばれる) の指定をサポートしています。

コマンドライン編集 (Unix)

Unix では、以下のキー・シーケンスでコマンドライン編集を行えます。

Ctrl-Space

マークを設定します。

Ctrl-A

行の先頭へ移動します。

Ctrl-B

カーソルを 1 文字左に移動させます。

Ctrl-D

カーソルの右側の文字を消去します。コマンドラインが空の場合はプログラムを終了します。

Ctrl-E

行の末尾へ移動します。

Ctrl-F

カーソルを 1 文字右に移動させます。

Ctrl-H

バックスペース

Ctrl-I

タブ

Ctrl-J

Enter

Ctrl-K

行の残り部分を削除します。

Ctrl-L

行を再描画します。

Ctrl-M

Enter

Ctrl-N

次の行に移動します。

Ctrl-P

前の行に移動します。

Ctrl-T

2 つの文字を切り替えます。

Ctrl-U

行を削除します。

Ctrl-W

領域を削除します (領域のもう一方の端には Ctrl-Space でマークされます)。

Ctrl-X

拡張コマンドを開始します。

Ctrl-Y

削除した行を貼り付けます。

Ctrl-_

最後の動作を取り消します。

Ctrl-X Ctrl-L

領域を小文字にします。

Ctrl-X Ctrl-U

領域を大文字にします。

Ctrl-X Ctrl-X

カーソルとマークを交換します。

Ctrl-X H

バッファ全体をマークします。

Ctrl-X U

最後の動作を取り消します。

Esc Ctrl-H

後方の単語を削除します。

Esc Delete

後方の単語を削除します。

Esc Space

余分なスペースを削除します (スペースを 1 つだけ残します)。

Esc <

行の先頭へ移動します。

Esc >

行の末尾へ移動します。

Esc @

現在の単語をマークします。

Esc A

1 文戻ります。

Esc B

1 語戻ります。

Esc C

現在の単語を大文字で初めます。

Esc D

現在の単語を削除します。

Esc E

1 文進みます。

Esc F

1 語進みます。

Esc K

現在の文を削除します。

Esc L

現在の単語を小文字に変更します。

Esc T

単語を入れ替えます。

Esc U

現在の単語を大文字に変更します。

Delete

バックスペース

ファイル名のサポート

ファイル名で使用できる文字セットは、オペレーティング・システムによって異なります。Unix ではファイル名に使用できる特殊文字がありますが、Windows では以下の文字は使用できません。

```
\/ : * ? " < > |
```

sftpg3 コマンドライン・ツール (対話型およびバッチ・ファイルの両方) は、エスケープ文字が ~ (チルダ) であることを除き、Windows プラットフォーム上でも Unix シェルのコマンドラインの構文とセマンティクスに従います。

ファイル名に特殊文字を含むファイル (`unixfilename*?".txt` など) を Unix サーバから Windows に転送する場合、Windows でも使用できる新しい名前をファイルに付ける必要があります。

sftpg3 コマンドライン・クライアントには 2 つのバージョンの **get** コマンドがあります:

> **get** コマンドは、複数のファイルを同時に転送するために使用できますが、ターゲットのファイル名は指定できません。ファイル名に特殊文字が含まれている場合は、その名前が Windows でも使用できるように、Unix でファイル名を変更しておく必要があります。
>
> **sget** コマンドは、一度に 1 つのファイルを転送するために使用され、転送先ファイルの新しい名前を定義できます。このコマンドは、Windows で名前が使用できるように使用します。コマンド・シーケンスは以下の通りです。

```
$ sftpg3
```

```
sftp> open user@server
```

```
sftp> sget "file*name.txt" windowsfilename.txt
```

特殊文字のエスケープ

sftpg3 コマンドでは、以下の文字は特別な意味を持ち、ファイル名を引数として取るコマンドではエスケープする必要があります。

***** : は任意の数の任意の文字に対応するワイルドカードです。

?: クエスチョン・マークは任意の 1 文字に対するワイルドカードです。

"" クォーテーション・マークは、表示通りに扱う文字列の前後に配置します。

****: バックスラッシュは Unix のエスケープ文字です。

~ チルダは Windows のエスケープ文字です。

エスケープ文字は、 **sftpg3** 次に来る文字を表示通りに扱い、特別な意味を持たせないように指示します。エスケープ文字はローカル・マシンのオペレーティング・システムに応じて選択されます。

**** 及び **~**は特殊文字そのものであり、ファイル名に使用する場合は、その前にもエスケープ文字を配置する必要があります。したがって、Unix では ****、Windows では **~** を含むファイル名

をいずれかの **sftpg3** コマンドに入力する必要がある場合は、該当する以下のエスケープ文字を付加します。

\\ : Unix の場合

~~: Windows の場合

ファイル名またはファイル名の一部がクォーテーション・マーク "" で囲まれている場合、**sftpg3** コマンドはクォーテーション・マーク内の部分を表示通りに解釈するので、クォーテーション・マーク内の文字がワイルドカードやその他の特殊文字として解釈されることはありません。

ただし、Unix ではクォーテーション・マーク " もファイル名の一部とすることができます。ファイル名に " 文字を入力する必要がある場合、Unix でも Windows でもその前にエスケープ文字を加える必要があります。

たとえば、Windows で `file-"name".txt` という名前のファイルをコマンドに入力するには、以下のコマンドを入力します。

```
sftp> sget "file-~"name~".txt" filename.txt
```

Tectia の **sftpg3** コマンドでエスケープ文字を使用する方法、及び異なるオペレーティング・システムで特殊文字を含むファイル名を入力する方法については、以下の例を参照してください。

sftpg3 コマンドのファイル名の例:

以下のファイル名は Unix で有効ですが、コマンドの中でエスケープ文字が必要です。

```
file|name.txt
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

Unix で **sftpg3** コマンドライン・ツールを使用する場合、上記のファイル名を以下のフォーマットで入力します。

```
file\|name.txt     or  "file|name.txt"
file-\"name\".txt  or  "file-\"name\".txt"
file\?name.txt     or  "file?name.txt"
file\*name.txt     or  "file*name.txt"
file\\name.txt     or  "file\\name.txt"
file\ -\ name.txt  or  "file - name.txt"
file~name.txt      or  "file~name.txt"
```

Unix のコマンドの例:

```
sftp> get "file*name.txt"
```

```
sftp> sget "file*name.txt" newfilename.txt
```

Windows で **sftpg3** コマンドを使用する場合、上記の Unix のファイル名を以下のフォーマットで入力します。

```
file~|name.txt     or  "file|name.txt"
file-~"name~".txt  or  "file-~"name~".txt"
file~?name.txt     or  "file?name.txt"
file~*name.txt     or  "file*name.txt"
file~\name.txt     or  "file\name.txt"
file~ -~ name.txt  or  "file - name.txt"
file~~name.txt     or  "file~~name.txt"
```

Windows のコマンド・シーケンスの例:

```
> sftpg3 open user@server
```

```
sftp> get "file name.txt"
```

```
sftp> sget "file*name.txt" filename.txt
```

環境変数

sftpg3 は以下の環境変数を使用します。

SSH_SFTP_BATCH_FILE=`startup_batch_file`

sftpg3 起動バッチ・ファイルへのパスを定義します。このファイルは **sftpg3** の起動時に実行され、 **sftpg3** が起動されるたびに、ファイルに定義されているコマンドが実行されます。

この変数が定義されていない場合は、 **sftpg3** は `ssh_sftp_batch_file` という名前の起動バッチ・ファイルを、Unix では `$HOME/.ssh2/`、Windows では `%APPDATA%\SSH\` というユーザ固有のディレクトリで探します。

この変数が定義されていて、ファイルが存在しないかアクセスできない場合、 **sftpg3** は起動に失敗します。

SSH_SFTP_CHECKSUM_MODE `=yes|no|md5|sha1|md5-force|sha1-force|checkpoint`

チェックサムを比較するための設定を定義します。 使用できる値の詳細については、 **checksum** を参照してください。

SSH_SFTP_SHOW_BYTE_COUNT `=yes|no`

この変数が `yes` に設定されている場合、ファイル転送に成功した後、転送されたバイト数が表示されます。また、転送元ファイルと転送先ファイルの名前も表示されます。デフォルトは `no` です。

SSH_SFTP_STATISTICS `=yes|no|simple`

この変数が `yes` に設定されている場合 (デフォルト)、ファイルの転送中に通常の進捗バーが表示されます。 `no` に設定されている場合、進捗バーは表示されません。 `simple` に設定されている場合は、ファイル転送後に統計データが表示されます。

終了値

sftpg3 は、操作の結果に基づいて以下の値を返します。

```
0     Operation was successful.
1     Internal error.
2     Connection aborted by the user.
3     Destination is not a directory, but a directory was specified by the user.
4     Connecting to the host failed.
5     Connection lost.
6     File does not exist.
7     No permission to access file.
8     Undetermined error from sshfilexfer.
11    Some non-fatal errors occured during a directory operation.
101   Wrong command-line arguments specified by the user.
```

バッチ・モードでは、実行中にエラーが発生しなかった場合にのみ **sftpg3** は値 0 を返します。バッチ処理中に、現在の作業ディレクトリの変更に失敗したり、接続の確立に失敗したり、接続が失われたりすると、 **sftpg3** は中断されます。その他のエラーは stderr に報告され、最後のエラー値が **sftpg3** プロセスの終了値として返されます。

例

sftpg3 セッションを開き、リモート・エンドを `ssh-broker-config.xml` ファイルの接続プロファイル `profile1` で定義されているサーバに接続します (ローカル・エンドは、SFTP クライアント・ホストのファイル・システムに接続されています)。

```
$ sftpg3 profile1
```

sftpg3 をバッチ・モードで実行します。

```
$ sftpg3 -B batch.txt
```

バッチ・ファイル `batch.txt` の内容の例を以下に示します。非対話型認証方法を使用し、サーバのホスト鍵はあらかじめ保存されています。

```
lopen user@unixserver.example.com
open user@winserver.example.com
binary
lcd backup
cd c:/temp
get --force-lower-case Testfile-X.bin
lchmod 700 testfile-x.bin
quit
```

このバッチ・ファイルの例では、接続のローカル・エンドを Unix サーバに、接続のリモート・エンドを Windows サーバに開き、転送モードをバイナリに設定しています。ローカル・ディレクトリを `backup` に、リモート・ディレクトリを `C:\Temp` に変更し、リモート・ディレクトリからローカル・ディレクトリにファイルをコピーします。ファイル名は小文字 (`testfile-x.bin`) に変更されます。転送後、所有者には完全な権限が、その他のユーザは一切の権限を持たないようにファイル・パーミッションが変更されます。

ssh-translation-table

ssh-translation-table — Secure Shell 変換テーブル

書式

`ssh-translation-table` [`options` **...**]
[`filename`]

説明

ssh-translation-table (Windows では **ssh-translation-table.exe**) は符号化文字集合 (CCS) 変換のための変換テーブルを生成するユーティリティ・プログラムです。 **ssh-translation-table** は変換テーブルを `filename` に保存します。 `filename` が指定されていない場合、 **ssh-translation-table** は変換テーブルを標準出力に書き込みます。

オプション

以下のオプションがあります。

`-b, --binary`

z/OS 固有のバイナリ・ファイル・フォーマットを使用します。

`-f, --from=` `CODESET`

インバウンド変換のソース・コード・セットを指定します。これはアウトバウンド変換のターゲット・コード・セットでもあります。デフォルト値は `ISO8859-1` です。例:

```
--from ISO8859-15
```

`-t, --to=` `CODESET`

インバウンド変換のターゲット・コード・セットを指定します。これはアウトバウンド変換のソース・コード・セットでもあります。デフォルト値は、基礎となる実装が ICU の場合は `IBM-1047,swaplfnl`、それ以外の場合は `IBM-1047` です。例:

```
--to IBM-037
```

`-l, --list-charsets`

利用できる文字セットを一覧表示します。すべての文字セットが 1 バイト文字セットではないことに注意してください。1 バイトの文字セットのみ使用できます。

`-D, --debug=` `LEVEL`

デバッグ・レベルを設定します。`LEVEL` は 0 から 99 までの数字で、99 はすべてのデバッグ情報を表示することを指定します。このオプションは、コマンドラインの最初の引数である必要があります。

`-h, --help`

コマンドライン・オプションの要約を表示して終了します。

変換テーブル

変換テーブルは、文字変換を記述した 2 つのテーブル (インバウンド・テーブルとアウトバウンド・テーブル) を含むファイルです。各テーブルは 256 個の変換値で構成されています。

Tectia のファイル転送では、行からデータ・セットにデータを変換するときにインバウンド・テーブルが使用されます。アウトバウンド・テーブルは、ファイルからデータを変換して、そのデータを行で送信するときに使用されます。

z/OS 固有のバイナリ・フォーマットは、256 バイトのフィールド 3 つで構成されています。1 つ目は EBCDIC のコメントで、変換ソフトウェアでは無視されます。2 つ目はインバウンド・テーブル、3 つ目はアウトバウンド・テーブルです。

テキスト・フォーマットでは各所にコメントを挿入できます。変換値は 16 進数で表されます。

テーブルとは、256 個の値を 16 進数の 2 文字 (00 から FF まで) で表したリストです。値の位置は、変換のためのインデックスです。最初の位置、すなわち 00 の位置は、バイト値 0 に対する変換値を表します。

テーブル中の 16 進数の値の大文字と小文字は区別されません。つまり、値 0a と 0A は同じです。また、ファイルにコメントを追加することもできます。コメントは文字 '#' で始まります。その文字以降、行の最後まではコメントとして扱われ、無視されます。また、ホワイト・スペースはすべて無視されます。

注意

変換テーブルでは 1 バイトの変換のみがサポートされています。

ssh-translation-table コマンドで生成された変換テーブルの例を以下に示します。

```
## SSH TRANSLATION TABLE FILE FORMAT VERSION 1.0
#####################################################################
#
# This file is an example translation table that can be used to
# translate data from 'ISO8859-1' to 'IBM-1047,swaplfnl' while reading
# from a file or from 'IBM-1047,swaplfnl' to 'ISO8859-1' while writing
# to a file.
#
# The format of translation table file is following:
#
# - White spaces are ignored.
# - Everything after '#' character until end of line is a comment
#   that is ignored.
# - The first table is used when writing data to a file.
# - The second table is used when reading data from a file.
# - Both tables must exist.
# - Table is a simple hexadecimal representation of the
```

```
#   translation. Each value is represented as two hexadecimal
#   characters. The first line gives the values in table
#   positions 0-15 (00-0F), the second line 16-31 (10-1F)
#   and so on.
#
# Note: Only single byte translations are supported.
#
#####################################################################

# Inbound (network to file) translation table:
# IBM-1047,swaplfnl -> ISO8859-1

#0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F
000102039C09867F978D8E0B0C0D0E0F #0
101112139D0A08871819928F1C1D1E1F #1
808182838485171B88898A8B8C050607 #2
909116939495960498999A9B14159E1A #3
20A0E2E4E0E1E3E5E7F1A22E3C282B7C #4
26E9EAEBE8EDEEEFECDF21242A293B5E #5
2D2FC2C4C0C1C3C5C7D1A62C255F3E3F #6
F8C9CACBC8CDCECFCC603A2340273D22 #7
D861626364656667686 9ABBBF0FDFEB1 #8
B06A6B6C6D6E6F707172AABAE6B8C6A4 #9
B57E737475767778797AA1BFD05BDEAE #A
ACA3A5B7A9A7B6BCBDBEDDA8AF5DB4D7 #B
7B41424344454647484 9ADF4F6F2F3F5 #C
7D4A4B4C4D4E4F505152B9FBFCF9FAFF #D
5CF7535455565758595AB2D4D6D2D3D5 #E
303132333435363738 39B3DBDCD9DA9F #F

# Outbound (file to network) translation table:
# ISO8859-1 -> IBM-1047,swaplfnl
#
#0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F
00010203372D2E2F1605150B0C0D0E0F #0
101112133C3D322618193F271C1D1E1F #1
405A7F7B5B6C507D4D5D5C4E6B604B61 #2
F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F #3
7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6 #4
D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D #5
798182838485868788889919293949596 #6
979899A2A3A4A5A6A7A8A9C04FD0A107 #7
202122232425061728292A2B2C090A1B #8
30311A333435360838393A3B04143EFF #9
41AA4AB19FB26AB5BBB49A8AB0CAAFBC #A
908FEAFABEA0B6B39DDA9B8BB7B8B9AB #B
6465626663679E687471727378757677 #C
AC69EDEEEBEFECBF80FDFEFBFCBAAE59 #D
4445424643479C485451525358555657 #E
8C49CDCECBCFCCE170DDDEDBDC8D8EDF #F

# EOF
```

注意

ICU ライブラリがアスキーから EBCDICへの変換テーブルの生成に使用される場合、アスキーのライン・フィード文字 (0A) が EBCDIC のニューライン文字 (15) に正しく変換されるように、EBCDIC コードページ名に `,swaplfnl` を追加する必要があります。

カスタム変換テーブルを作成するには、まず **ssh-translation-table** で変換テーブルを作成してから、任意のテキスト・エディタで編集します。

環境変数

`SSH_CHARSET_CONV`

Tectia の変換 DLL のフル・パス名。 **ssh-translation-table** または変換 DLL がインストール・ディレクトリにない場合にのみ必要です。以下にパス名の例を示します。

```
SSH_CHARSET_CONV=/opt/tectia/lib/shlib/i18n_iconv.so
```

ssh-keygen-g3

ssh-keygen-g3 — 認証鍵ペア生成器

書式

`ssh-keygen-g3` [`options` **...**]
[`key1 key2` **...**]

説明

ssh-keygen-g3 (Windows では **ssh-keygen-g3.exe**) は Secure Shell 用の認証鍵を生成及び管理するツールです。Secure Shell クライアントを公開鍵認証で使用する場合は、このツールを実行して認証鍵を作成できます。また、システム管理者はこのツールを使用して、Secure Shell サーバ用のホスト鍵を生成することもできます。このツールは、openSSH の公開鍵や秘密鍵を Tectia 鍵フォーマットに変換したり、Tectia の鍵フォーマットから openSSH フォーマットに変換したりすることもできます。Tectia の公開鍵は Secure Shell (SSH) 公開鍵ファイル・フォーマット (RFC 4716) を使用します。

デフォルトでは、鍵ファイルのパスを指定しない場合、鍵ペアはユーザのホーム・ディレクトリの下 (Unix では `$HOME/.ssh2`、Windows では "`%APPDATA%\SSH\UserKeys`") に生成されます。ファイル名を指定しない場合も同様に、ユーザのホーム・ディレクトリに `id_type_bits_a` や `id_type_bits_a.pub` などのファイル名で鍵ペアが保存されます。

スペースを含むファイル・パスなどの文字列を指定する場合は、引用符 ("") で囲んでください。

オプション

以下のオプションがあります。

`-1 file`

鍵ファイルを SSH1 フォーマットから SSH2 フォーマットに変換します。注意: "`1`" は数字の 1 です (英字の L ではありません)。

`-7 file`

PKCS #7 ファイルから証明書を抽出します。

`-b bits`

生成される鍵の長さをビット数で指定します。各鍵タイプに許容される長さとデフォルトの長さは以下の通りです。

- DSA/RSA: 許容される長さは 512 〜 65536 ビット、デフォルトの長さは 3072 ビット

- ECDSA: 許容される長さは 256、384、521 ビット、デフォルトの長さは 384 ビット
- Ed25519: 許容される長さ、デフォルトの長さともに 256 ビット

`-B num`

鍵情報を表示するための基数を指定します (デフォルト: `10`)。

`-c comment`

生成された鍵のコメント文字列を指定します。

`-D file`

秘密鍵 `file` から公開鍵を生成します。

`-e file`

指定された鍵を編集します。 **ssh-keygen-g3** を対話型にします。鍵のパスフレーズやコメントを変更できます。

`-F, --fingerprint file`

指定された公開鍵のフィンガープリントとタイプ (RSA、DSA、ECDSA または Ed25519) をダンプします。デフォルトでは、フィンガープリントは SSH Babble 形式で指定されるので、フィンガープリントは「実在する」単語の文字列のように見えます (発音しやすくします)。出力フォーマットは `--fingerprint-type` オプションで変更できます。

このオプションの動作を変更するには、 `--fingerprint-type`、 `--hash`、 `--hostkeys-directory`、 `--known-hosts`、 `--rfc4716` のオプションを使用します。

`-F, --fingerprint host_ID`

指定された `host_ID` で識別される、ローカルに保存されたホスト鍵の場所、フィンガープリント、タイプ (RSA、DSA、ECDSA、または Ed25519) をダンプします。`host_ID` には、ホスト名または "`host#port`" の文字列を指定します。

このオプションの動作を変更するには、 `--fingerprint-type`、 `--hash`、 `--hostkeys-directory`、 `--known-hosts`、 `--rfc4716` のオプションを使用します。

`-H, --hostkey`

生成された鍵ペアをデフォルトのホスト鍵ディレクトリ (Unix では `/etc/ssh2`、Windows では "`<INSTALLDIR>\SSH Tectia Server`") に保存します。 デフォルトでは秘密鍵はパスフレーズ無しでまたは FIPS モードでは `<privatekey>.pass` にランダムなパスフレーズを生成して保存します。 `--hostkey` モードでパスフレーズを要求するためには `--prompt-pass` を使用します。

`-i file`

鍵 `file` の情報を読み込み、表示します。

`--pass-file file`

パスフレーズで保護された秘密鍵ファイルの情報を -i で表示するときに、パスフレーズを `file` から読み込みます。 <privatekey>.pass が存在する場合はデフォルトでそれを使用します。

PKCS #12 ファイルを SSH2 フォーマットの証明書と秘密鍵に変換します。

`-m, --generate-moduli-file`

Diffie-Hellman グループ交換のための moduli ファイル `secsh_dh_gex_moduli` を生成します。

`-p passphrase`

生成された鍵のパスフレーズを指定します。

`-P`

生成された鍵が空のパスフレーズで保存されることを指定します。

注意

FIPS モードでは、暗号化されていない秘密鍵を FIPS モジュールからエクスポートすることが FIPS 規定において禁じられているため、パスフレーズなしでユーザ鍵を生成することはできません。

`--random-pass`

生成された秘密鍵のためにランダムなパスフレーズを生成し <privatekey>.pass に保存します。デフォルトではパスフレーズは Base64 でエンコードされます。

`-k file`

PKCS #12 ファイルを SSH2 フォーマットの証明書と秘密鍵に変換します。

`-m, --generate-moduli-file`

Diffie-Hellman グループ交換のための moduli ファイル `secsh_dh_gex_moduli` を生成します。

`-q, --quiet`

鍵生成中の進捗インジケータを非表示にします。

`-r file`

`file` からランダム・プールにエントロピを追加します。`file` に「相対的にランダムな」データ (潜在的な攻撃者が予測できないデータ) が含まれている場合、プールの乱数性は高まります。生成された鍵のセキュリティを確保するためには、高い乱数性が不可欠です。

`-t dsa | rsa | ecdsa | ed25519`

鍵タイプを選択します。有効な値は `rsa` (デフォルト)、 `dsa`、 `ecdsa`、及び `ed25519` です。

`-x file`

秘密鍵を X.509 フォーマットから SSH2 フォーマットに変換します。

`--append [ =yes | no ]`

鍵を付加します。オプションの値は `yes` 及び `no` です。デフォルトは `yes` で、付加します。

`--copy-host-id host_ID destination`

指定された転送先ディレクトリにホスト ID をコピーします。

このオプションの動作を変更するには、 `--append`、 `--hostkeys-directory`、 `--known-hosts`、 `--overwrite` のオプションを使用します。

`--hostkey-file` を指定すると、そのファイルは接続ブローカーによって使用される通常のホスト識別ファイルとして扱われ、その内容が転送先ディレクトリにコピーされます。

`--delete-host-id host_ID`

指定されたホスト ID のホスト鍵を削除します。`host_ID` には、ホスト名または "`host#port`" の文字列を指定します。

このオプションの動作を変更するには、 `--host-key-file`、 `--hostkeys-directory`、 `--known-hosts` のオプションを使用します。

`--hash sha256 | sha1 | md5`

フィンガープリント生成のためのダイジェスト・アルゴリズムを指定します。有効なオプションは `sha256`、 `sha1` 及び `md5` です。. デフォルトは sha1 です。

`--fingerprint-type base64 | babble | babble-upper | pgp-2 | pgp-5 | hex | hex-upper`

フィンガープリントの出力フォーマットを指定します。このオプションを指定する場合は、 `-F` オプションと鍵ファイル名を先行させる必要があります。デフォルトのフォーマットは `babble` です。

このオプションの使用例については、「例」 を参照してください。

`--fips-mode`

暗号化ライブラリの FIPS モードを使用して鍵を生成します。

鍵には空でないパスフレーズを指定する必要があります。

デフォルトでは (このオプションを指定しない場合または Tectia の FIPSMODE スイッチファイルが存在しない場合)、暗号化ライブラリの標準モードを使用して鍵が生成されます。

`--fips-crypto-dll-path PATH`

FIPS 暗号化 DLL の場所を指定します。

`--hostkey-file file`

コピーするときは、場所を自動検出するのではなく、指定されたファイルをコピー元のホスト鍵として使用します。削除するときは、指定された場所からのみ削除します。指定されたファイルが、指定されたホストの識別子を含んでいない場合、何も行いません。

`--hostkeys-directory directory`

デフォルトの場所の代わりに、使用する既知のホスト鍵のディレクトリを指定します。

`--import-public-key infile [outfile]`

`infile` から公開鍵をインポートし、--key-format パラメータで指定されたフォーマットで `outfile` への保存を試みます。`outfile` が指定されていない場合は、指定するように求められます。デフォルトの出力フォーマットは SSH2 ネイティブのフォーマットです。

`--import-private-key infile [outfile]`

`infile` から秘密鍵をインポートし、--key-format パラメータで指定されたフォーマットで `outfile` への保存を試みます。`outfile` が指定されていない場合は、指定するように求められます。デフォルトの出力フォーマットは SSH2 ネイティブの秘密鍵フォーマットです。

`--import-ssh1-authorized-keys infile outfile`

SSH1 形式の authorized_keys ファイル `infile` をインポートし、SSH2 形式の認証ファイル `outfile` を生成し、`infile` の鍵を `outfile` と同じディレクトリに生成されたファイルに保存します。

`--key-format format`

出力される鍵のフォーマットは secsh2、 pkcs1、 pkcs8、 pkcs12、 openssh2、または openssh2-aes です。

`--key-hash hash`

このオプションは Tectia の鍵フォーマット以外でも使用できます。パスフレーズに基づく秘密鍵の生成に使用するハッシュ・アルゴリズムを指定します。デフォルト値は `sha1` です。その他に `sha224`、 `sha256`、 `sha384`、及び `sha512` のアルゴリズムがサポートされています。すべての鍵フォーマットが、すべてのハッシュ・アルゴリズムをサポートしているわけではないことに注意してください。

`--known-hosts file`

指定された既知の hosts ファイルを使用します。OpenSSH 形式の known-hosts ファイルに定義されているホストのフィンガープリントの取得を有効にします。

このオプションを使用すると、known_hosts ファイルのデフォルトの場所 (/etc/ssh/ssh_known_hosts 及び $HOME/.ssh/known_hosts) が上書きされます。空文字列を指定すると、known-hosts の使用は完全に無効になります。

--moduli-file-name file

Diffie-Hellman グループ交換のために生成された moduli を file に書き込みます。(オプション -m のデフォルトのファイル名は secsh_dh_gex_moduli です。)

--overwrite [=yes | no]

同じファイル名のファイルを上書きします。デフォルトは上書きです。

--rfc4716

RFC4716 で規定されたフォーマットでフィンガープリントを表示します。ダイジェスト・アルゴリズム (ハッシュ) は md5、出力フォーマットはコロン (:) で区切られた小文字の HEX で 16 バイト出力です。

--set-hostkey-owner-and-dacl file

Windows では、ホスト鍵 file の正しい所有者と DACL (任意アクセス制御リスト) を設定します。このオプションは、Tectia Server のインストール中にホスト鍵が生成される際に内部的に使用されます。

--sign-cert file

生成された公開鍵で証明書を作成し、file に書き込みます。その他の証明書オプションの完全なリストについては、 **--sign-cert help** でオプションのヘルプを確認ください。

-V

バージョン文字列を表示し、終了します。

-h, --help, -?

コマンドライン・オプションの要約を表示して終了します。

例

FIPS モードの暗号化ライブラリを使用して 3072 ビットの RSA 鍵ペアを作成し、デフォルトのユーザ鍵ディレクトリに newkey 及び newkey.pub のファイル名で保存します。

```
$ ssh-keygen-g3 --fips-mode -b 3072 newkey
```

サーバ・ホストの公開鍵のフィンガープリントを SSH Babble (デフォルト) 形式で表示します。

```
$ ssh-keygen-g3 -F hostkey.pub
Fingerprint for key:
xeneh-fyvam-sotaf-gutuv-rahih-kipod-poten-byfam-hufeh-tydym-syxex
```

ホスト公開鍵の Base64 でエンコードされた SHA256 フィンガープリントを表示します。

```
$ ssh-keygen-g3 --hash sha256 --fingerprint-type base64 -F hostkey.pub
Fingerprint for key `hostkey.pub':
9UmbXHpUodKPXS0pFIACGLjKoiHQBShPVZj6ShUNWgM (RSA)
```

秘密鍵を openSSH2-AES フォーマットに変換します。

```
$ ssh-keygen-g3 -p <password> --key-format openssh2-aes \
  --import-private-key <source_key_file> <destination_key_file>
```

注意: 変換される秘密鍵ファイルがパスフレーズで暗号化されている場合、パスフレーズは '-p' オプションで指定する必要があります。

Tectiaの公開鍵 `tectiakey.pub` を OpenSSH 公開鍵 `opensshkey.pub` に変換します。

```
$ ssh-keygen-g3 --key-format openssh2 --import-public-key \
  tectiakey.pub opensshkey.pub
```

Diffie-Hellman グループ交換のための moduli ファイル `dhgex-moduli` を生成します。

```
$ ssh-keygen-g3 -m --moduli-file-name dhgex-moduli
```

ssh-keyfetch

ssh-keyfetch — Secure Shell クライアント用のホスト鍵ツール

書式

`ssh-keyfetch` [`options` ...]
[`host`]

説明

ssh-keyfetch (Windows では **ssh-keyfetch.exe**) は、サーバのホスト鍵をダウンロードし、オプションで Secure Shell クライアントの既知のホスト鍵として設定するためのツールです。このツールは通常、システム管理者が初期セットアップの段階で使用します。

デフォルトでは、ホスト鍵はサーバから取得され、カレント・ディレクトリの `key_host_port.suffix` ファイルに保存されます。

オプション

以下のオプションがあります。

`-a, --set-trusted`

公開鍵をファイルに書き込むのではなく、既知のホスト鍵として公開鍵をユーザ固有のディレクトリ`$HOME/.ssh2/hostkeys`(Windows の場合は `%APPDATA%\SSH\HostKeys`) に追加します。このオプションを `-C` または `-K` と組み合わせることはできません。

警告

ssh-keyfetch を `-a` オプションとともに実行した場合、受信したホスト鍵は、ユーザに確認することなく自動的に受け入れられます。鍵の受信後に鍵のフィンガープリントを検証し有効性を確認する必要があります。そうしないと、中間者攻撃の対象になるリスクがあります。

ホスト鍵を検証するには、信頼できるソースからホスト鍵のフィンガープリントを入手し(サーバ管理者に電話するなど)、以下のコマンドの出力と照合します。

```
ssh-keygen-g3 --fingerprint <hostname>
```

`-A, --fetch-any`

サーバの公開鍵または証明書のいずれかを調べて取得します。

`-C, --fetch-certificate`

サーバ証明書のみを調べて取得します。

`-d, --debug debug-level`

デバッグを有効にします。

`-D, --debug-default`

デフォルト・レベルでデバッグを有効にします。

`-f, --filename-format nameformat`

既知のホスト鍵のファイル名フォーマット。受け入れられる値は `plain` 及び `hashed` です。デフォルトは `plain` です。

`-F, --fingerprint-type [ =babble | babble-upper | pgp-2 | pgp-5 | hex | hex-upper ]`

メッセージとログに表示されるフィンガープリントのための公開鍵のフィンガープリントのタイプ。最も一般的なタイプは `babble` (SSH Babble 形式) と `hex` です。デフォルトは `babble` です。 `--rfc4716` のオプションも参照してください。

`-H, --hash [ =md5 | sha1 ]`

フィンガープリント生成のためのダイジェスト・アルゴリズムを指定します。有効なオプションは `md5` と `sha1` です。

`-K, --kex-key-formats typelist`

プロトコルの鍵交換で受け入れられるホスト鍵のタイプを明示的に指定します。専門家のためのオプションです。詳細については、RFC 4253 を参照してください。

`-l, --log`

正常に受信した鍵をログ・フォーマットで報告します。ログ・フォーマットは鍵 1 つにつき 1 行、1 行につき 6 フィールドで構成されています。以下のフィールドがあります。

- accept|save
- replace|append
- hostname
- ip-port
- user-id
- key-file-path
- fingerprint

`-o, --output-file output-file`

`output-file` に結果を書き込みます。マイナス記号 ("-") は標準出力を表します。

`-O, --output-directory output-dir`

`output-dir` に結果を書き込みます。デフォルトはカレント・ディレクトリです。

`-p, --port port`

サーバ・ポート (デフォルト: `22`)。

`-P, --fetch-public-key`

サーバ公開鍵のみをプローブしてフェッチします。これがデフォルトの動作です。

`-q, --quiet`

サイレント・モード。エラーのみを報告します。

`-R, --rfc4716`

RFC4716 で規定されているフォーマットで公開鍵のフィンガープリントを表示します。ダイジェスト・アルゴリズム (ハッシュ) は md5、出力フォーマットはコロン (:) で区切られた小文字の HEX で 16 バイト出力です。

`-S, --proxy-url socks-url`

使用する SOCKS サーバを指定します。

`-t, --timeout timeout`

秒単位の接続タイムアウト (デフォルト: `10` 秒)。

`--append [ =yes | no ]`

新しいホスト鍵を付加する代わりに、このホストの既存の既知のホスト鍵を上書きします。オプションの値は `yes` 及び `no` です。デフォルトは付加です。

`-V, --version`

バージョン文字列を表示し、終了します。

環境変数

SSH_SOCKS_SERVER

ssh-keyfetch で使用される SOCKS サーバのアドレス。

例

SOCKS プロキシを経由してサーバに接続します。

```
$ ssh-keyfetch -S socks://fw.example.com:1080/10.0.0.0/8 server.outside.example
Public key from server.outside.example:22 saved.
 File: server.outside.example.pub
```

```
 Fingerprint: xucar-bened-liryt-lumup-minad-tozuc-pesyp-vafah-mugyd-susic-guxix
```

サーバ鍵を Tectia Client の既知の鍵として受け入れ、より厳格なログ・フォーマットで報告します。

```
$ ssh-keyfetch -a -l  newhost
Accepted newhost 22 testuser /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-syxyx
```

サーバ鍵を Tectia Client の既知の鍵として受け入れ、その鍵をグローバル設定の `hostkeys` ディレクトリに保存します。

```
$ ssh-keyfetch -a --output-directory /etc/ssh2/hostkeys
Accepted newhost 22 testuser /etc/ssh2/hostkeys/key_22_anotherhost.pub
bydop-mulym-zegar-nybuv-muled-syxyx-xigad-hozuf-kykek-vogid-dumid
```

サーバ鍵を Tectia Client の既知の鍵として受け入れ、保存された既知の鍵のファイル名として、情報を与えないためのハッシュを使用します。

```
$ ssh-keyfetch -f hashed -a  newhost
Public key from newhost:22 accepted as trusted hostkey.
 File:
 /home/testuser/.ssh2/hostkeys/keys_420b23ca959ab165e52e117a90baa89d92ffc535
 Fingerprint:
 xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-syxyx
```

ポート番号 222 で動作しているサーバの X.509 証明書を取得し、**ssh-certview** で内容を表示します。

```
$ ssh-keyfetch -C -p 222 -o - newhost | ssh-certview -
Certificate =
  SubjectName = <C=FI, O=SSH, OU=DEV, CN=newhost.ssh.com>
  IssuerName = <C=FI, O=SSH, CN=Sickle CA>
  SerialNumber= 24593438
  Validity =
    NotBefore = 2007 Sep 13th, 15:10:00 GMT
    NotAfter  = 2008 Sep 12th, 15:10:00 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm = RSA
      Modulus n  (1024 bits) :
...
  Fingerprints =
    MD5 = 3c:71:17:9b:c2:12:26:cf:96:27:fb:d7:a8:19:37:89
    SHA-1 =
    14:72:f3:0f:20:5e:75:ed:d2:c3:86:4b:69:45:00:47:ae:fe:31:64
```

以下のように鍵交換タイプのリストを明示的に指定するのは、`-A` オプションを指定するのと同等になります。

```
$ ssh-keyfetch -K ssh-rsa,ssh-dss,x509v3-sign-rsa,x509v3-sign-dss newhost
Public key from newhost:22 saved.
 File: key_newhost_22.pub
 Fingerprint:
 xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-syxyx
```

ssh-cmpclient-g3

ssh-cmpclient-g3 — CMP 登録クライアント

書式

```
ssh-cmpclient-g3 command [options] access [name]

Where command is one of the following:

     INITIALIZE psk|racerts keypair template
     ENROLL certs|racerts keypair template
     UPDATE certs [keypair]
     POLL psk|certs|racerts id

     RECOVER psk|certs|racerts template
     REVOKE psk|certs|racerts template

     TUNNEL racerts template

Most commands can accept the following options:
     -B            Perform key backup for subject keys.
     -o prefix     Save result into files with given prefix.
     -O filename   Save the result into the specified file.
                   If there is more than one result file,
                   the remaining results are rejected.
     -C file       CA certificate from this file.
     -S url        Use this SOCKS server to access the CA.
     -H url        Use this HTTP proxy to access the CA.
     -E            PoP by encryption (CA certificate needed).
     -v num        Protocol version 1|2 of the CA platform. Default is 2.
     -y            Non-interactive mode. All questions answered with 'y'.
     -N file       Specifies a file to stir to the random pool.
     -d level      Set debug level.
     -Z provspec   Specifies external key provider for the private key.
                   The format of provspec is "providername:initstring".


The following identifiers are used to specify options:
     psk      -p refnum:key (reference number and pre-shared key)
              -p file (containing refnum:key)
              -i number (iteration count, default 1024)
     certs    -c file (certificate file) -k url (private-key URL)
     racerts  -R file (RA certificate file) -k url (RA private-key URL)
     keypair  -P url (private-key URL)
     id       -I number (polling ID)
     template -T file (certificate template)
              -s subject-ldap[;type=value]
              -u key-usage-name[;key-usage-name]
              -U extended-key-usage-name[;extended-key-usage-name]
     access   URL where the CA listens for requests.
     name     LDAP name for the issuing CA (if -C is not given).
```

```
Key URLs are either valid external key paths or in the format:
     "generate://savetype:passphrase@keytype:size/save-file-prefix"
     "file://passphrase/relative-key-file-path"
     "file:relative-key-file-path"
     "any-key-file-path"

The key generation "savetype" can be:
 - ssh2, secsh2, secsh (Secure Shell 2 key type)
 - ssh1, secsh1 (legacy Secure Shell 1 key type)
 - pkcs1 (PKCS #1 format)
 - pkcs8s (passphrase-protected PKCS #8, "shrouded PKCS #8")
 - pkcs8 (plain-text PKCS #8)
 - x509 (Tectia-proprietary X.509 library key type)

     -h Prints usage message.
     -F Prints key usage extension and keytype instructions.
     -e Prints command-line examples.
```

説明

ssh-cmpclient-g3 コマンドライン・ツール (Windows では **ssh-cmpclient-g3.exe**) は CMP プロトコルを使用する証明書登録クライアントです。このコマンドライン・ツールでは、RSA または DSA の公開鍵ペアを生成し、その公開コンポーネントの証明書を取得できます。CMP は、証明書のライフサイクル管理のために IETF PKIX ワーキング・グループによって規定されたプロトコルで、RSA Keon などの一部の CA プラットフォームでサポートされています。

コマンド

ssh-cmpclient-g3 コマンドラインのコマンド・キーワードを以下に示します。3 文字より長ければ、短縮してもコマンドを識別できます。コマンドの大文字と小文字は区別されません。ユーザはコマンドごとに CA アドレス URL を指定する必要があります。ここでいう「ユーザ」とは、ユーザ、プログラム、またはハードウェア・デバイスを指します。

`INITIALIZE`

ユーザの初期証明書を要求します。この要求は、何らかの他の方法で CA または RA から受け取った参照番号と、それに対応する鍵 (PSK) を使用して認証されます。

ユーザは、PSK、非対称暗号鍵ペア、及びサブジェクト名を指定する必要があります。

`ENROLL`

ユーザがすでに鍵に対して有効な証明書を持っている場合に、新しい証明書を要求します。この要求は、公開鍵を使って認証されることを除けば、`initialize`と同じです。

`POLL`

要求がすぐには受け入れられなかった場合に、証明書をポーリングします。

`UPDATE`

既存の証明書の更新 (置き換え) を要求します。発行される証明書は、既存の証明書と (名前、フラグ、その他の拡張が) 同様のものになります。ユーザは鍵を変更でき、有効期限は CA によって更新されます。この要求は有効な既存の鍵ペアと証明書によって認証されます。

`RECOVER`

バックアップされた鍵の回復を要求します。この要求は PSK ベースまたは証明書ベースで認証されます。テンプレートには、すでにバックアップされており回復しようとしている秘密鍵の証明書が記述されています。ユーザは、自身でバックアップした鍵しか回復できません。

`REVOKE`

テンプレートで指定された鍵の失効を要求します。要求の認証は、PSK または失効対象者と同一ユーザに属する証明書を使用して行われます。

`TUNNEL`

RA トンネル・モードで動作します。要求を読み込み、オプションでコマンドラインに基づいてサブジェクト名、代替名、拡張子を変更します。要求を承認して CA に送信します。

オプション

ssh-cmpclient-g3 コマンドライン・オプションを以下に示します。ファイル名を指定した場合、既存の同名のファイルは上書きされることに注意してください。スペースを含むサブジェクト名などの文字列を指定する場合は、引用符 ("") で囲んでください。

`-B`

initialize、enroll、及び update コマンドで秘密鍵のバックアップを要求します。

`-o prefix`

結果として得られた証明書と CRL を、指定された `prefix` を持つファイルに保存します。プレフィックスにはまず数字が付加され、その後にオブジェクトの種類に応じてファイル拡張子 `.crt` または `.crl` が付加されます。

`-O filename`

指定された絶対ファイル名に結果を保存します。複数の結果ファイルがある場合、残りの結果は拒否されます。

`-C file`

CA 証明書を含むファイル・パスを指定します。鍵がバックアップされている場合、ファイル名を指定する必要がありますが、ほとんどの場合は代わりに CA の LDAP 名を指定できます。

`-S url`

CA が SOCKS 対応のファイアウォールの背後にある場合、SOCKS URL を指定します。URL のフォーマットは次のようになります: `socks://[username@]server[:port][/network/bits[,network/bits]]`

`-H url`

CA へのアクセスに、指定された HTTP プロキシ・サーバを使用します。URL のフォーマットは次のようになります: `http://server[:port]/`

`-E`

CA がサポートしている場合、暗号化の所有権証明 (PoP) を実行します。この方式の PoP では、要求は署名されず、代わりに CA から受け取った証明書を復号する能力に基づいて PoP が確立されます。CA は証明書を、ユーザの公開鍵で暗号化してからユーザに送信します。

`-v num`

CMP プロトコルのバージョンを選択します。値は、RFC 2510 ベースのプロトコルの場合は 1、CMPv2 の場合は2 (デフォルト) です。

`-N file`

鍵生成時にエントロピ源として使用するファイルを指定します。

`-d level`

デバッグ・レベル文字列を `level` に設定します。

`-Z provspec`

秘密鍵の外部鍵プロバイダを指定します。`provspec` は `"providername:initstring"` のフォーマットで指定します。

usage 行は以下のメタ・コマンドを使用します。

`psk`

CA または RA から与えられる参照番号と、それに対応する鍵の値。

`-p refnum:key|file`

`refnum` と `key` は、CA とユーザの間で共有される文字列です。`refnum` は、メッセージの認証に使用される秘密の `key` を特定します。`refnum` 文字列にコロンを含めてはなりません。

代わりに、参照番号と鍵を含むファイル名を引数として指定できます。

`-i number`

`number` は鍵ハッシュの反復回数を示します。

`certs`

認証に使用するユーザの既存の鍵と証明書。

`-k url`

秘密鍵の場所を指定する URL。これは、「書式」 で指定されたフォーマットの外部鍵 URL です。

`-c file`

`-k` オプションの引数で指定された公開鍵に対して発行された証明書を含むファイルのパス。

`racerts`

RA モードでは、認証のための RA 鍵と証明書。

`-k url`

秘密鍵の場所を指定する URL。これは、「書式」 で指定されたフォーマットの外部鍵 URL です。

`-R file`

`-k` オプションの引数で指定された公開鍵に対して発行された RA 証明書を含むファイルのパス。

`keypair`

証明されるサブジェクトの鍵ペア。

`-P url`

秘密鍵の場所を指定する URL。これは、「書式」 で指定されたフォーマットの外部鍵 URL です。

`id`

PKI のアクションが保留のままである場合に使用されるポーリング ID。

`-I number`

アクションが保留のままである場合に、RA または CA から与えられるポーリング・トランザクション ID の `number`。

`template`

証明されるサブジェクト名とフラグ。

`-T file`

操作のテンプレートとして使用される証明書を含むファイル。サブジェクトを識別するための値はここから読み込まれますが、ユーザは鍵、鍵用途フラグ、またはサブジェクト名を上書きできます。

`-s subject-ldap[;type=value]*`

リバース LDAP フォーマットのサブジェクト名。つまり、最も一般的なコンポーネントから始まり、代替のサブジェクト名が続きます。`subject-ldap` 名は要求に文字通りににコピーされます。

一般的には `"C=US,O=SSH,CN=Some Body"` というフォーマットで DN を選択しますが、原則として、結果として得られる証明書に使用できれば何でも指定できます。

`type` の値には `ip`、`email`、`dn`、`dns`、`uri`、及び `rid` を指定できます。

`-u key-usage-name[;key-usage-name]*`

要求された鍵用途のコード。認識されるコードは `digitalSignature`、`nonRepudiation`、`keyEncipherment`、`dataEncipherment`、`keyAgreement`、`keyCertSign`、`cRLSign`、`encipherOnly`、`decipherOnly`、及び `help` です。特殊キーワードの help に、RFC 3280 で定義されていて、サポートされている鍵用途の一覧があります。

`-U extended-key-usage-name[;extended-key-usage-name]*`

要求された拡張鍵用途のコード。ユーザが指定したドット付き OID 値のほか、`serverAuth`、`clientAuth`、`codeSigning`、`emailProtection`、`timeStamping`、`ikeIntermediate`、及び `smartCardLogon` のコードが認識されます。

`access`

CA のアドレスを URL フォーマットで指定します。指定できるアクセス方法は HTTP (`http://host:port/path`) またはプレーンな TCP (`tcp://host:port/path`) です。ホスト・アドレスが IPv6 アドレスの場合、角括弧で囲む必要があります (例: `http://[IPv6-address]:port/`)。

`name`

オプション `-C` を使用して CA 証明書が与えられなかった場合、操作の宛先 CA の名称をオプションで指定します。

例

初期証明書の登録

この例では、電子署名を使用するために初期証明書を登録するためのコマンドについて説明します。このコマンドは、秘密鍵を PKCS#8 のプレーンテキスト・ファイル `initial.prv` に生成し、登録された証明書を `initial-0.crt` というファイルに保存します。ユーザは CA に対して、鍵識別子 (refnum) `62154` と鍵 `ssh` で認証されます。鍵用途フラグとともに、サブジェクト名と代替 IP アドレスが指定されます。CA アドレスは `pki.ssh.com`、ポート番号は `8080`、アクセスする CA の名称は `Test CA 1` です。

```
$ ssh-cmpclient-g3 INITIALIZE \
   -P generate://pkcs8@rsa:2048/initial -o initial \
   -p 62154:ssh \
```

```
   -s 'C=FI,O=SSH,CN=Example/initial;IP=1.2.3.4' \
   -u digitalsignature \
   http://pki.ssh.com:8080/pkix/ \
   'C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities'
```

応答として、コマンドは発行された証明書をユーザに提示し、ユーザはプロンプトで `yes` と入力することでそれを受け入れます。

```
Certificate =
  SubjectName = <C=FI, O=SSH, CN=Example/initial>
  IssuerName = <C=FI, O=SSH Communications Security Corp,
    CN=SSH Test CA 1 No Liabilities>
  SerialNumber= 8017690
  SignatureAlgorithm = rsa-pkcs1-sha1
  Validity = ...
  PublicKeyInfo = ...
  Extensions =
      Viewing specific name types = IP = 1.2.3.4
    KeyUsage = DigitalSignature
    CRLDistributionPoints = ...
    AuthorityKeyID =
      KeyID = 3d:cb:be:20:64:49:16:1d:88:b7:98:67:93:f0:5d:42:81:2e:bd:0c
    SubjectKeyID =
      KeyId = 6c:f4:0e:ba:b9:ef:44:37:db:ad:1f:fc:46:e0:25:9f:c8:ce:cb:da
  Fingerprints =
    MD5 = b7:6d:5b:4d:e0:94:d1:1f:ec:ca:c2:ed:68:ac:bf:56
    SHA-1 = 4f:de:73:db:ff:e8:7d:42:c4:7d:e1:79:1f:20:43:71:2f:81:ff:fa

Do you accept the certificate above? yes
```

鍵の更新

証明書が失効する前に、有効期限を更新した新しい証明書を登録する必要があります。 **ssh-cmpclient-g3** は鍵の更新をサポートしています。鍵の更新時、新しい秘密鍵が生成され、鍵の更新要求が古い (まだ有効な) 証明書で認証されます。古い証明書は新しい証明書を発行するためのテンプレートとして使用されるので、鍵を更新する際にユーザのIDが変更されることはありません。以下のコマンドを実行すると、前述の例で登録した鍵ペアを更新できます。結果として得られる証明書の提示は省かれています。

```
$ ssh-cmpclient-g3 UPDATE \
   -k initial.prv -c initial-0.crt -P \
   generate://pkcs8@rsa:2048/updatedcert -o updatedcert \
   http://pki.ssh.com:8080/pkix/ \
   "C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities"
```

新しい鍵ペアは、 `updatedcert` というプレフィックスを持つファイル内にあります。 **ssh-cmpclient-g3** を `UPDATE` モードで使用する場合、発行元の CA のポリシーで、鍵の自動更新が許可されている必要があります。

ssh-scepclient-g3

ssh-scepclient-g3 — SCEP 登録クライアント

書式

```
ssh-scepclient-g3 command [options] access [name]
```

```
Where command is one of the following:
     GET-CA
     GET-CHAIN
     ENROLL psk keypair template

Most commands can accept the following options:
     -o prefix       Save result into files with prefix.
     -S url          Use this socks server to access CA.
     -H url          Use this HTTP proxy to access CA.

The following identifiers are used to specify options:
     psk      -p key (used as revocationPassword or challengePassword)
     keypair  -P url (private-key URL)
     ca       -C file (CA certificate file)
              -E file (RA encryption certificate file)
              -V file (RA validation certificate file)
     template -T file (certificate template)
              -s subject-ldap[;type=value]
              -u key-usage-name[;key-usage-name]
              -U extended-key-usage-name[;extended-key-usage-name]
     access   URL where the CA listens for requests.

GET-CA and GET-CHAIN take name argument, that is something
interpreted by the CA to specify a CA entity managed by the responder.

Key URLs are either valid external key paths or in the format:
     "generate://savetype:password@keytype:size/save-file-prefix"
     "file://savetype:password@/file-prefix"
     "file://passphrase/file-prefix"
     "file:/file-prefix"
     "key-filename"

The "keytype" for the SCEP protocol has to be "rsa".

The key generation "savetype" can be:
 - ssh2 (Secure Shell 2 key type)
 - ssh1 (Legacy Secure Shell 1 key type)
 - ssh  (Tectia proprietary crypto library format, passphrase-protected)
 - pkcs1 (PKCS#1 format)
 - pkcs8s (passphrase-protected PKCS#8, "shrouded PKCS#8")
 - pkcs8 (plain-text PKCS#8)
 - x509 (Tectia proprietary X.509 library key type)
```

説明

ssh-scepclient-g3 コマンドライン・ツール (Windows では **ssh-scepclient-g3.exe**) は SCEP プロトコルを使用する証明書登録クライアントです。このコマンドライン・ツールでは、RSA の公開鍵ペアを生成し、その公開コンポーネントの証明書を取得できます。SCEP プロトコルは、Cisco 製ルータで使用するために Cisco と Verisign が開発したものです。現在、ほとんどの CA プラットフォームは、クライアント証明書の登録にこのプロトコルをサポートしています。

コマンド

ssh-scepclient-g3 コマンドラインのコマンド・キーワードを以下に示します。3 文字より長ければ、短縮してもコマンドを識別できます。コマンドの大文字と小文字は区別されません。ユーザはコマンドごとに CA アドレス URL を指定する必要があります。ここでいう「ユーザ」とは、ユーザ、プログラム、またはハードウェア・デバイスを指します。

`GET-CA`

CA からの CA または RA 証明書のダウンロードを要求し、CA の検証のために証明書のフィンガープリントを表示します。フフィンガープリントは、何らかの他のメカニズムを使用して CA から受け取る必要があります。

`GET-CHAIN`

CA/RA からトップレベル CA までの証明書チェーンを要求します。

`ENROLL`

CA に新しい証明書を要求します。CA は何らかの他のメカニズムを使用してこの要求を承認します。または、CA から受け取ったパスワードを含めることもできます。

オプション

`-o prefix`

出力された証明書を、指定されたプレフィックスを持つファイルに保存します。プレフィックスにはまず数字が付加され、その後に CA 証明書の場合は `.ca`、ユーザ証明書の場合は `.crt` というファイル拡張子が付加されます。

`-S url`

CA が SOCKS 対応のファイアウォールの背後にある場合、SOCKS URL を指定します。URL のフォーマットは次のようになります: `socks://[username@]server[:port][/network/bits[,network/bits]]`

`-H url`

CA へのアクセスに、指定された HTTP プロキシ・サーバを使用します。URL のフォーマットは次のようになります: `http://server[:port]/`

usage 行は以下のメタ・コマンドを使用します。

`psk`

CA または RA から与えられる事前共有鍵、または発行された証明書の失効をユーザが希望する場合に、クライアントが作成し CA に提供する失効パスワード。この種類と必要性は、CA が使用する PKI プラットフォームに依存します。

`-p key`

認証要求または失効要求の認可のために、CA に転送される認証パスワードまたは失効パスワード (暗号化されたフォーマット)。

`keypair`

証明されるサブジェクトの鍵ペア。

`-P url`

秘密鍵の場所を指定する URL。これは、「書式」 で指定されたフォーマットの外部鍵 URL です。

`ca`

CA/RA 証明書。

`-C file`

登録を行う際、指定されたファイル・パスから CA 証明書を読み込みます。

`-E file`

オプションで、RA 暗号化証明書を指定します。

`-V file`

オプションで、RA 署名証明書を指定します。

`template`

証明されるサブジェクト名とフラグ。

`-T file`

操作のテンプレートとして使用される証明書を含むファイル。サブジェクトを識別するための値はここから読み込まれますが、ユーザは鍵、鍵用途フラグ、またはサブジェクト名を上書きできます。

`-s subject-ldap[;type=value]*`

リバース LDAP フォーマットのサブジェクト名。つまり、最も一般的なコンポーネントから始まり、代替のサブジェクト名が続きます。`subject-ldap` 名は要求に文字通りににコピーされます。

一般的には "C=US,O=SSH,CN=Some Body" というフォーマットで DN を選択しますが、原則として、結果として得られる証明書に使用できれば何でも指定できます。

type の値には ip、email、dn、dns、uri、及び rid を指定できます。

-u key-usage-name[;key-usage-name]*

要求された鍵用途のコード。認識されるコードは digitalSignature、 nonRepudiation、keyEncipherment、 dataEncipherment、 keyAgreement、 keyCertSign、 cRLSign、 encipherOnly、decipherOnly、及び help です。特殊キーワードの help に、 RFC 3280 で定義されていて、サポートされている鍵用途の一覧があります。

-U extended-key-usage-name[;extended-key-usage-name]*

要求された拡張鍵用途のコード。ユーザが指定したドット付き OID 値のほか、serverAuth、 clientAuth、 codeSigning、 emailProtection、 timeStamping、 ikeIntermediate、及び smartCardLogon のコードが認識されます。

access

CA のアドレスを URL フォーマットで指定します。ホスト・アドレスが IPv6 アドレスの場合、角括弧で囲む必要があります (例: http://[IPv6-address]:port/)。

name

宛先 CA の名称を指定します。

例

以下の例では、まず CA 証明書を受け取ります。CA アドレスは pki.ssh.com、ポート番号は 8080、CA の名称は test-ca1.ssh.com です。

```
$ ssh-scepclient-g3 GET-CA \
   -o ca http://pki.ssh.com:8080/scep/ \
   test-ca1.ssh.com

Received CA/RA certificate ca-0.ca:

fingerprint 9b:96:51:bb:29:0d:c9:e0:75:c8:03:0d:0d:92:60:6c
```

次に、RSA 証明書を登録します。ユーザは CA に対して鍵 ssh で認証されます。鍵用途フラグとともに、サブジェクト名と代替 IP アドレスが指定されます。

```
$ ssh-scepclient-g3 ENROLL \
    -C ca-0.ca -p ssh \
    -o subject -P generate://pkcs8:ssh@rsa:2048/subject \
    -s 'C=FI,O=SSH,CN=SCEP Example;IP=1.2.3.4' \
    -u digitalsignature \
    http://pki.ssh.com:8080/scep/

Received user certificate subject-0.crt:
fingerprint 4b:7e:d7:67:27:5e:e0:54:2f:5b:56:69:b5:01:d2:15
```

```
$ ls subject*
subject-0.crt   subject.prv
```

ssh-certview-g3

ssh-certview-g3 — 証明書ビューア

書式

```
ssh-certview-g3
[options...] file
[options...] file ...
```

説明

ssh-certview-g3 プログラム (Windows では **ssh-certview-g3.exe**) は、X.509 証明書、CRL、及び証明書要求をデコードして表示できる、シンプルなコマンドライン・アプリケーションです。このコマンドの出力は標準出力に書き込まれます。

オプション

以下のオプションがあります。

`-h`

簡単なヘルプを表示します。

`-verbose`

より詳細な診断結果を出力します。

`-quiet`

診断結果を出力しません。

`-auto`

次の入力ファイルの種類は自動検出されます (デフォルト)。

`-cert`

次の入力ファイルは証明書です。

`-certpair`

次の入力ファイルはクロス証明書ペアです。

`-crmf`

次の入力ファイルは CRMF の証明書要求です。

`-req`

次の入力ファイルは PKCS #10 の証明書要求です。

`-crl`

次の入力ファイルは CRL です。

`-prv`

次の入力ファイルは秘密鍵です。

`-pkcs12`

次の入力ファイルは PKCS#12 パッケージです。

`-ssh2`

次の入力ファイルは SSH2 公開鍵です。

`-spkac`

次の入力ファイルは、Netscape が生成した SPKAC 要求です。

`-noverify`

入力された証明書の署名の有効性をチェックしません。

`-autoenc`

PEM/DER を自動的に判別します (デフォルト)。

`-pem`

入力ファイルが PEM (ASCII base-64) フォーマットであるとみなします。このオプションには、実際のPEM (ヘッダとフッタあり) とプレーンな base-64 (ヘッダとフッタなし) の両方を使用できます。PEM のヘッダとフッタの例を以下に示します。

```
-----BEGIN CERTIFICATE-----
encoded data
-----END CERTIFICATE-----
```

`-der`

入力ファイルが DER フォーマットであるとみなします。

`-hexl`

入力ファイルが Hexl フォーマットであるとみなします。(Hexl はバイナリ・ファイルを 16 進数で出力するための Unix 共通のツールです。)

`-skip number`

デコードを試みる前に、入力の先頭から `number` バイトをスキップします。これは、実際の内容の前に不要なデータが含まれているファイルに役立ちます。

`-ldap`

LDAP の順序で名前を表示します。

`-utf8`

UTF-8 で名前を表示します。

`-latin1`

ISO-8859-1 で名前を表示します。

`-base10`

大きな数字を基数 10 で出力します (デフォルト)。

`-base16`

大きな数字を基数 16 で出力します。

`-base64`

大きな数字を基数 64 で出力します。

`-width number`

出力幅を (`number` 文字) に設定します。

例

たとえば、`pki.ssh.com` からダウンロードした証明書を使用して、以下のコマンドを指定します。

```
$ ssh-certview-g3 -width 70 ca-certificate.cer
```

以下のように出力されます。

```
Certificate =
  SubjectName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  IssuerName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  SerialNumber= 34679408
  SignatureAlgorithm = rsa-pkcs1-sha1
  Certificate seems to be self-signed.
      * Signature verification success.
  Validity =
    NotBefore = 2003 Dec  3rd, 08:04:27 GMT
    NotAfter  = 2005 Dec  2nd, 08:04:27 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm name (SSH) : if-modn{sign{rsa-pkcs1-md5}}
      Modulus n  (1024 bits) :
```

```
      9635680922805930263476549641957998756341022541202937865240553
      9374740946079473767424224071470837728840839320521621518323377
      3593102350415987252300817926769968881159896955490274368606664
      0759644131690750532665266218696466060377799358036735475902257
      6086098562919363963470926690162744258451983124575595926849551
      903
    Exponent e (  17 bits) :
      65537
Extensions =
  Available = authority key identifier, subject key identifier, key
    usage(critical), basic constraints(critical), authority
    information access
  KeyUsage = DigitalSignature KeyEncipherment KeyCertSign CRLSign
      [CRITICAL]
  BasicConstraints =
    PathLength = 0
    cA         = TRUE
      [CRITICAL]
  AuthorityKeyID =
    KeyID =
      eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
  SubjectKeyID =
    KeyId =
      eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
  AuthorityInfoAccess =
    AccessMethod = 1.3.6.1.5.5.7.48.1
    AccessLocation =
      Following names detected =
        URI (uniform resource indicator)
      Viewing specific name types =
        URI = http://pki.ssh.com:8090/ocsp-1/
Fingerprints =
  MD5 = c7:af:e5:3d:f6:ea:ce:da:07:93:d0:06:8d:c0:0a:f8
  SHA-1 =
  27:d7:19:47:7c:08:3e:1a:27:4b:68:8e:18:83:e8:f9:23:e8:29:85
```

ssh-ekview-g3

ssh-ekview-g3 — 外部鍵ビューア

書式

`ssh-ekview-g3 [options...] provider`

説明

ssh-ekview-g3 プログラム (Windows では **ssh-ekview-g3.exe**) を使用すると、外部鍵プロバイダから証明書をエクスポートできます。エクスポートした証明書は **ssh-certview-g3** でさらに詳しく調べることができます。

これは、証明書認証を許可するためのエントリを `ssh-server-config.xml` ファイルに生成する場合などに便利です。証明書のサブジェクト名を知ることが必要です

ssh-ekview-g3 では、証明書をエクスポートして、証明書から必要な情報を **ssh-certview-g3** で取得できます。

オプション

以下のオプションがあります。

`-h`

簡単なヘルプを表示します。

`-i info`

プロバイダの初期化文字列として `info` を使用します。

`-k`

鍵のパスのみを表示します。

`-e keypath`

`keypath` にある証明書をファイルにエクスポートします。

`-a`

見つかったすべての証明書をファイルにエクスポートします。

`-b base`

整数を表示する際に `base` を使用します。たとえば、10 進数の 10 は base-16 では「a」です。

付録D egrep 構文

Tectia トンネリングのフィルタのルールは、 **egrep** 構文を使用して指定されたホスト名または IP アドレス・パターンに一致させることができます。また、値の範囲を指定する際に、セレクタで正規表現を使用できます。本項では、egrep 構文について説明します。

D.1. egrep のパターン

エスケープ文字はバックスラッシュ (\) です。通常の文字として使用するために、メタ文字をエスケープできます。

以下の例では、リテラル 'E' と 'F' は、パターンあるいは文字でも、あらゆる表現を意味します。

(

キャプチャする部分式を開始します。

)

キャプチャする部分式を終了します。

E|F

選言。E または F のどちらかに一致 (包含)両方が一致する場合、E が優先されます。

E*

Kleene スターとして機能し、E に 0 回以上の一致

E+

閉包。E に 1 回またはそれ以上の一致

E?

オプション。E に0 回または 1 回一致

.

改行文字 (\n, \f, \r) 及び NULL バイトを除く任意の文字に一致

E{n}

E に 厳密に n 回の一致

E{n,} または E{n,0}

E に n 回またはそれ以上の一致

E{,n} または E{0,n}

E に n 回までの一致

E{n,m}

E に n 回以上 m 回 以下の一致

[

文字セットを開始。 D.3 を参照してください。

$

入力の末尾、または行末にある空の文字列に一致

^

入力の先頭、または行頭にある空の文字列に一致

D.2. 正規表現構文 egrep のエスケープされるトークン

\0n..n

8 進数値 n...n のリテラル・バイト

\0

NULL バイト

\[1-9]..x

10 進数値 [1-9]..x のリテラル・バイト

\xn..n または \0xn..n

16 進数値 n...n のリテラル・バイト

\<

単語の先頭にある空の文字列に一致

\>

単語の末尾にある空の文字列に一致

\b

単語境界にある空の文字列に一致

\B

単語境界でない場合、空の文字列に一致

\w

単語構成文字に一致、[a:zA:Z0:9-] と等価

\W

単語構成ではない文字に一致

\a

リテラルのアラーム文字

\e

リテラルのエスケープ文字

\f

リテラルのライン・フィード

\n

C 言語の \n に相当するリテラルの改行。したがって、一文字以上の場合がある。

\r

リテラルのキャリッジ・リターン

\t

リテラルのタブ

上記以外のエスケープ文字はリテラル文字そのものを表します。

D.3. egrep の文字セット

POSIX 文字コード指定子の一部ではなく、'[' の直後に続かない、'[' で始まり、エスケープされていない ']' で終わる文字コード。

以下の文字は特殊な意味を持つので、リテラルとして使用する場合はエスケープする必要があります。

- (マイナス記号)

範囲演算子。ただし、その特別な意味を失うため '[' の直後を除く。

^

開始の '[' の直後であれば補集合を示します。文字セット全体の補集合となります。それ以外の場合は、リテラルの '^'。

[:alnum:]

'isalnum' が true を返す文字 。

[:alpha:]

'isalpha' が true を返す文字 。

[:cntrl:]

'iscntrl' が true を返す文字 。

[:digit:]

'isdigit' が true を返す文字 。

[:graph:]

'isgraph' が true を返す文字 。

[:lower:]

'islower' が true を返す文字 。

[:print:]

'isprint' が true を返す文字 。

[:punct:]

'ispunct' が true を返す文字 。

[:space:]

'isspace' が true を返す文字 。

[:upper:]

'isupper' が true を返す文字 。

[:xdigit:]

'isxdigit' が true を返す文字 .

例: `[[:xdigit:]XY]` **は通常、** `[0123456789ABCDEFabcdefXY]` **に相当します。**

定義済みのエスケープ文字セットを新たに定義した文字セットに含めることもでき、その場合は `[\d\s]` **が数字と空白文字に一致します。**

また、リテラルと評価されるエスケープ・シーケンスは文字セット内部で機能します。

付録E 監査メッセージ

この付録では、接続ブローカーによって生成される監査メッセージを一覧にまとめています。

1000 KEX_failure
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵交換に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名 (最初の鍵交換では存在しない)
Algorithm	鍵交換アルゴリズム名 (アルゴリズムを選択する前に障害が発生した場合は存在しない)
Text	エラーの説明
Session-Id	セッションの識別子

1001 Algorithm_negotiation_failure
レベル: 警告
出所: Tectia Server、接続ブローカー

アルゴリズム・ネゴシエーションに失敗しました。クライアントとサーバのリストに共通のアルゴリズムがありませんでした。

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名 (最初の鍵交換では存在しない)

引数	説明
Algorithm	アルゴリズムの種類
Client algorithms	クライアントのアルゴリズム・リスト
Server algorithms	サーバのアルゴリズム・リスト
Session-Id	セッションの識別子

1002 Algorithm_negotiation_success
レベル: 情報
出所: Tectia Server、接続ブローカー

アルゴリズム・ネゴシエーションに成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名 (最初の鍵交換では存在しない)
Text	ネゴシエーションによるアルゴリズム
Session-Id	セッションの識別子

1003 KEX_success
レベル: 情報
出所: 接続ブローカー

鍵交換に成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Algorithm	鍵交換方法の名前
Session-Id	セッションの識別子
Protocol-session-Id	プロトコル・セッションの識別子

1100 Certificate_validation_failure
レベル: 情報
出所: Tectia Server、接続ブローカー

受信した証明書が、設定されているどの CA でも正しく検証されませんでした。

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名 (最初の鍵交換では存在しない)
Text	設定されているすべての CA の検索状態の結果。
Session-Id	セッションの識別子

1101 Certificate_validation_success
レベル: 情報

出所: Tectia Server、接続ブローカー

受信した証明書が、設定されているいずれかの CA で正しく検証されました。

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名
CA List	ユーザの証明書が正しく検証された CA のリスト。
Session-Id	セッションの識別子

1110 CM_find_started
レベル: 情報
出所: Tectia Server、接続ブローカー

証明書検証サブシステムで低レベルの検索が開始されました。

デフォルトのログ・ファシリティ: normal

引数	説明
Ctx	検索コンテキスト
Search constraints	検索の制約

1111 CM_find_finished
レベル: 情報
出所: Tectia Server、接続ブローカー

使用されたソースをトレースして検索が完了しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Ctx	検索を特定するコンテキスト・ポインタ
Text	使用されたソースを特定する検索トレース

1112 CM_cert_not_in_search_interval
レベル: 情報
出所: Tectia Server、接続ブローカー

要求された期間、証明書が有効ではありません。

デフォルトのログ・ファシリティ: normal

引数	説明
SubjectName	証明書のサブジェクト名
Text	エラーの説明
Ctx	検索コンテキスト

1113 CM_certificate_revoked
レベル: 情報

出所: Tectia Server、接続ブローカー

証明書が失効していることが判明しました。

デフォルトのログ・ファシリティ: normal

引数	説明
SubjectName	証明書のサブジェクト名
Ctx	検索のコンテキスト・ポインタ

1114 CM_cert_search_constraint_mismatch
レベル: 情報
出所: Tectia Server、接続ブローカー

証明書が、検索に設定されている制約を満たしませんでした。

デフォルトのログ・ファシリティ: normal

引数	説明
SubjectName	証明書のサブジェクト名
Text	不一致の説明
Ctx	検索コンテキスト

1115 CM_ldap_search_started
レベル: 情報
出所: Tectia Server、接続ブローカー

CRL またはサブ CA の LDAP 検索を開始しています。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	検索の詳細

1116 CM_ldap_search_success
レベル: 情報
出所: Tectia Server、接続ブローカー

CRL またはサブ CA の LDAP 検索が正常に完了しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	検索の詳細

1117 CM_ldap_search_failure
レベル: 情報
出所: Tectia Server、接続ブローカー

LDAP サーバとの通信に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	エラーの詳細

1118 CM_http_search_started
レベル: 情報
出所: Tectia Server、接続ブローカー

証明書検証サブシステムは、HTTP プロトコルにより CRL またはサブ CA の検索を開始しています。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	検索ターゲット

1119 CM_http_search_success
レベル: 情報
出所: Tectia Server、接続ブローカー

CRL またはサブ CA の HTTP 要求が正常に完了しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	取得した内容を詳しく示すステータス・メッセージ

1120 CM_http_search_failure
レベル: 情報
出所: Tectia Server、接続ブローカー

CRL またはサブ CA の HTTP 要求に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	エラーの詳細

1121 CM_crl_added
レベル: 情報
出所: Tectia Server、接続ブローカー

新しい CRL が証明書検証サブシステムに正常に追加されました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	CRL の発行元と有効期間

1122 Certificate_end_point_id_check_success
レベル: 情報

出所: 接続ブローカー

エンド・ポイント同一性チェックに成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Server	ホスト名
Text	説明メッセージ

1123 Certificate_end_point_id_check_warning
レベル: 情報
出所: 接続ブローカー

証明書のエンド・ポイント同一性チェックの警告

デフォルトのログ・ファシリティ: normal

引数	説明
Server	ホスト名
Text	警告メッセージ

1124 Certificate_end_point_id_check_failure
レベル: 情報
出所: 接続ブローカー

証明書のエンド・ポイント同一性チェックの失敗

デフォルトのログ・ファシリティ: normal

引数	説明
Server	ホスト名
Text	エラー・メッセージ

1200 Key_store_create
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵ストアが作成されました。

デフォルトのログ・ファシリティ: normal

1201 Key_store_create_failed
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵ストアの作成に失敗しました。

デフォルトのログ・ファシリティ: normal

1202 Key_store_destroy

レベル: 情報
出所: Tectia Server、接続ブローカー

鍵ストアが破壊されました。

デフォルトのログ・ファシリティ: normal

1204 Key_store_add_provider
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵ストアにプロバイダが追加されました。

デフォルトのログ・ファシリティ: normal

引数	説明
Type	プロバイダの種類

1205 Key_store_add_provider_failed
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵ストアへのプロバイダの追加に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Type	プロバイダの種類
EK error	エラー・メッセージ

1206 Key_store_remove_provider
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵ストアからプロバイダが削除されました。

デフォルトのログ・ファシリティ: normal

引数	説明
Init info	プロバイダ名

1208 Key_store_decrypt
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵を使用した復号化に成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス

引数	説明
Fwd path	転送パス

1209 Key_store_decrypt_failed
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵を使用した復号化に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス
Fwd path	転送パス
Crypto error	エラー文字列

1210 Key_store_sign
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵を使用した署名に成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス
Fwd path	転送パス

1211 Key_store_sign_failed
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵を使用した署名に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス
Fwd path	転送パス
Crypto error	エラー文字列

1212 Key_store_sign_digest
レベル: 情報
出所: Tectia Server、接続ブローカー

鍵を使用したダイジェストへの署名に成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス

引数	説明
Fwd path	転送パス

1213 Key_store_sign_digest_failed
レベル: 警告
出所: Tectia Server、接続ブローカー

鍵を使用したダイジェストへの署名に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス
Fwd path	転送パス
Crypto error	エラー文字列

1214 Key_store_ek_provider_failure
レベル: 警告
出所: Tectia Server、接続ブローカー

外部鍵プロバイダの障害

デフォルトのログ・ファシリティ: normal

引数	説明
Key path	鍵のパス
Text	鍵ラベル
Text	エラーの説明

1220 Key_store_certificate_issued
レベル: 情報
出所: Tectia Server、接続ブローカー

内部 CA が X.509 証明書を発行しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	CA 名
Text	プリンシパル名
Text	有効期限
Text	証明書の SHA-256 ハッシュ

1221 Key_store_certificate_revoked
レベル: 情報
出所: Tectia Server、接続ブローカー

内部 CA が証明書を失効させました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	CA 名
Text	プリンシパル名
Text	有効期限
Text	証明書の SHA-256 ハッシュ

1300 Channel_inbound_statistics
レベル: 情報
出所: 接続ブローカー、Tectia Server

チャネルのインバウンド側 (ネットワークから到着するトラフィック) の統計

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名
Session-Id	セッションの識別子
Channel Id	ローカル・チャネルの ID
Packet count	プロトコル・パケットのカウント
Packet size	プロトコル・パケットの平均ペイロード・サイズ

1301 Channel_outbound_statistics
レベル: 情報
出所: 接続ブローカー、Tectia Server

チャネルのアウトバウンド側 (ネットワークに向かうトラフィック) の統計

デフォルトのログ・ファシリティ: normal

引数	説明
Username	ユーザのログイン名
Session-Id	セッションの識別子
Channel Id	ローカル・チャネルの ID
Packet count	プロトコル・パケットのカウント
Packet size	プロトコル・パケットの平均ペイロード・サイズ
Packet size	アウトバウンド・チャネル・バッファの最終サイズ

3000 Sft_client_start
レベル: デバッグ
出所: Tectia Secure File Transfer clients

ファイル転送クライアント・プログラムが起動しました。

デフォルトのログ・ファシリティ: user

3001 Sftc_create_file

レベル: デバッグ
出所: Tectia Secure File Transfer clients

新しいファイルが作成されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
File	ターゲット・ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3002 Sftc_truncate_file
レベル: デバッグ
出所: Tectia Secure File Transfer clients

ファイルが切り詰められました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
File	ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3003 Sftc_modify_file_attrs
レベル: 情報
出所: Tectia Secure File Transfer clients

ファイル属性が変更されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名

引数	説明
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
File	ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3004 Sftc_delete_file
レベル: 通知
出所: Tectia Secure File Transfer clients

ファイルが削除されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
File	ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3005 Sftc_create_dir
レベル: デバッグ
出所: Tectia Secure File Transfer clients

ディレクトリが作成されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
Dir	ディレクトリへのパス

引数	説明
Text	(オプション) エラー・メッセージ及び/または追加情報

3006 Sftc_remove_dir
レベル: 通知
出所: Tectia Secure File Transfer clients

ディレクトリが削除されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
Dir	ディレクトリへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3007 Sftc_copy_dir_start
レベル: 通知
出所: Tectia Secure File Transfer clients

ディレクトリのコピーが開始されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Source Connection	ソース接続
From	ソース・ディレクトリへのパス
Target Connection	ターゲット接続
To	ターゲット・ディレクトリへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3008 Sftc_copy_dir_finished
レベル: 通知
出所: Tectia Secure File Transfer clients

ディレクトリのコピーが完了しました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Source Connection	ソース接続
From	ソース・ディレクトリへのパス
Target Connection	ターゲット接続
To	ターゲット・ディレクトリへのパス
Duration	ディレクトリをコピーする期間
Files	コピーされたファイル数
Data	コピーされたデータ量 (バイト単位)
Speed	コピー速度 (KiB/s 単位)
Text	(オプション) エラー・メッセージ及び/または追加情報

3009 Sftc_move_dir_start
レベル: 通知
出所: Tectia Secure File Transfer clients

ディレクトリの移動が開始されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Source Connection	ソース接続
From	ソース・ディレクトリへのパス
Target Connection	ターゲット接続
To	ターゲット・ディレクトリへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3010 Sftc_move_dir_finished
レベル: 通知
出所: Tectia Secure File Transfer clients

ディレクトリの移動が完了しました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Source Connection	ソース接続
From	ソース・ディレクトリへのパス
Target Connection	ターゲット接続
To	ターゲット・ディレクトリへのパス
Duration	ディレクトリを移動する期間
Files	移動されたファイル数
Data	転送されたデータ量 (バイト単位)
Speed	転送速度 (KiB/s 単位)
Text	(オプション) エラー・メッセージ及び/または追加情報

3011 Sftc_copy_file_start
レベル: 情報
出所: Tectia Secure File Transfer clients

ファイルのコピーが開始されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Source Connection	ソース接続
From	ソース・ファイルへのパス
Target Connection	ターゲット接続
To	ターゲット・ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3012 Sftc_copy_file_finished
レベル: 通知
出所: Tectia Secure File Transfer clients

ファイルのコピーが完了しました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Source Connection	ソース接続
From	ソース・ファイルへのパス
Target Connection	ターゲット接続
To	ターゲット・ファイルへのパス
Duration	期間
Data	コピーされたデータ量 (バイト単位)
Speed	コピー速度 (KiB/s 単位)
Text	(オプション) エラー・メッセージ及び/または追加情報

3013 Sftc_move_file_start
レベル: 情報
出所: Tectia Secure File Transfer clients

ファイルの移動が開始されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Source Connection	ソース接続
From	ソース・ファイルへのパス
Target Connection	ターゲット接続
To	ターゲット・ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3014 Sftc_move_file_finished
レベル: 通知
出所: Tectia Secure File Transfer clients

ファイルの移動が完了しました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名

引数	説明
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Source Connection	ソース接続
From	ソース・ファイルへのパス
Target Connection	ターゲット接続
To	ターゲット・ファイルへのパス
Duration	ファイルを移動する期間
Data	転送されたデータ量 (バイト単位)
Speed	転送速度 (KiB/s 単位)
Text	(オプション) エラー・メッセージ及び/または追加情報

3015 Sftc_rename_file
レベル: 情報
出所: Tectia Secure File Transfer clients

ファイルの名前が変更されました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
File	ファイルへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

3017 Sft_client_command
レベル: デバッグ
出所: Tectia Secure File Transfer clients

ファイル転送のクライアント・コマンド

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム

引数	説明
Pid	クライアント・プロセス ID
OpID	操作 ID
Text	コマンドラインで指定されたコマンド、またはバッチ・ファイルで定義されたコマンド

3018 Sftc_open_dir
レベル: デバッグ
出所: Tectia Secure File Transfer clients

ディレクトリが開かれました。

デフォルトのログ・ファシリティ: user

引数	説明
Local username	ローカルのユーザ名
Program	セキュアなファイル転送クライアント・プログラム
Pid	クライアント・プロセス ID
OpID	操作 ID
Status	結果 (SUCCESS または FAILED)
Connection	ターゲット接続
Dir	ディレクトリへのパス
Text	(オプション) エラー・メッセージ及び/または追加情報

6000 Broker_client_connect
レベル: 情報
出所: 接続ブローカー

クライアントがブローカーに接続しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	処理 ID
Local username	ローカルのユーザ名

6001 Broker_client_connect_failed
レベル: 警告
出所: 接続ブローカー

クライアントによるブローカーへの接続の試みが失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名

引数	説明
Pid	処理 ID
Local username	ローカルのユーザ名
Text	理由

6002 Broker_client_disconnect
レベル: 情報
出所: 接続ブローカー

クライアントがブローカーとの接続を切断しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	処理 ID
Local username	ローカルのユーザ名

6004 Broker_exec_channel_open
レベル: 情報
出所: 接続ブローカー

ブローカーが exec チャネルを開きました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Command	コマンド
Text	Exec パラメータ
Channel Id	チャネル ID
Session-Id	セッション ID

6005 Broker_exec_channel_open_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがクライアントのために exec チャネルを開くのに失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID

引数	説明
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Command	コマンド
Text	Exec パラメータ
Channel Id	チャネル ID
Text	理由
Session-Id	セッション ID

6006 Broker_tunnel_open
レベル: 情報
出所: 接続ブローカー

ブローカーがクライアントのためにトンネルを開きました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Dst	接続先ホスト
Dst Port	接続先ポート
Tunnel type	トンネルの種類
Session-Id	セッション ID

6007 Broker_tunnel_open_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがクライアントのためにトンネルを開くのに失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Dst	接続先ホスト
Dst Port	接続先ポート

引数	説明
Tunnel type	トンネルの種類
Text	理由
Session-Id	セッション ID

6008 Broker_tunnel_listener_open
レベル: 情報
出所: 接続ブローカー

ブローカーがクライアントのためにトンネル・リスナを開きました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Listener	リスナ・ホスト
Listener Port	リスナ・ポート
Dst	接続先ホスト
Dst Port	接続先ポート
Tunnel type	トンネルの種類
Text	トンネル・リスナのパラメータ
Session-Id	セッション ID

6009 Broker_tunnel_listener_open_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがクライアントのためにトンネル・リスナを開くのに失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Server	サーバ・ホスト
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Listener	リスナ・ホスト
Listener Port	リスナ・ポート
Dst	接続先ホスト
Dst Port	接続先ポート
Tunnel type	トンネルの種類

引数	説明
Text	トンネル・リスナのパラメータ
Text	理由
Session-Id	セッション ID

6010 Broker_channel_fd_strip
レベル: 情報
出所: 接続ブローカー

ブローカーがチャネル・オブジェクトを破壊しました (さらに、原因となっている fd をクライアントに返しました)。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Text	チャネルは恒久的かどうか
Local username	ローカルのユーザ名
Session-Id	セッション ID

6011 Broker_channel_fd_strip_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがチャネル・オブジェクトの破壊 (及び、原因となっている fd のクライアントへの戻し) に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Text	チャネルは恒久的かどうか
Local username	ローカルのユーザ名
Text	理由
Session-Id	セッション ID

6012 Broker_channel_control
レベル: 情報
出所: 接続ブローカー

ブローカーがチャネル制御メッセージを送信しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名

引数	説明
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Command	コマンド
Args	引数
Local username	ローカルのユーザ名
Session-Id	セッション ID

6013 Broker_channel_control_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがチャネル制御メッセージの送信に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Command	コマンド
Args	引数
Local username	ローカルのユーザ名
Text	理由
Session-Id	セッション ID

6014 Broker_channel_close
レベル: 情報
出所: 接続ブローカー

ブローカーがチャネルを閉じました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Exit Value	終了値
Local username	ローカルのユーザ名
Session-Id	セッション ID

6015 Broker_channel_close_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがチャネルを閉じるのに失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Local username	ローカルのユーザ名
Text	理由

6018 Broker_server_version_request
レベル: 情報
出所: 接続ブローカー

ブローカーがサーバ・バージョンを要求 (及び取得) しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Ver	バージョン
Local username	ローカルのユーザ名
Session-Id	セッション ID

6019 Broker_server_version_request_failed
レベル: 警告
出所: 接続ブローカー

ブローカーがサーバ・バージョンの取得に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Local username	ローカルのユーザ名
Text	理由
Session-Id	セッション ID

6020 Broker_channel_process_exit
レベル: 情報
出所: 接続ブローカー

チャネル・プロセス終了要求が成功しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Client	クライアント名

引数	説明
Pid	クライアント・プロセス ID
Local username	ローカルのユーザ名
Session-Id	セッション ID

6021 Broker_channel_process_exit_failed
レベル: 警告
出所: 接続ブローカー

チャネル・プロセス終了要求に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Text	理由
Local username	ローカルのユーザ名
Session-Id	セッション ID

6025 Broker_connector_license_check_failed
レベル: 警告
出所: 接続ブローカー

Connector のライセンス・チェックに失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	エラー・メッセージ
Session-Id	セッション ID

6026 Broker_server_rekey
レベル: 通知
出所: 接続ブローカー

ブローカーが鍵更新を要求し、成功しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Local username	ローカルのユーザ名
Session-Id	セッション ID

6027 Broker_server_rekey_failed
レベル: 警告
出所: 接続ブローカー

ブローカーが鍵更新を要求しましたが、失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Channel Id	チャネル ID
Local username	ローカルのユーザ名
Text	理由
Session-Id	セッション ID

6035 Broker_publickey_upload
レベル: 情報
出所: 接続ブローカー

公開鍵がアップロードされました。

デフォルトのログ・ファシリティ: normal

引数	説明
Client	クライアント名
Pid	クライアント・プロセス ID
Local username	ローカルのユーザ名
Public key hash	公開鍵のハッシュ
Public key hash (SHA-256)	公開鍵のハッシュ (SHA-256)
Server	サーバ名
Server Port	サーバ・ポート
Remote username	リモートのユーザ名
File name	公開鍵のファイル名

6100 Broker_starting
レベル: 通知
出所: 接続ブローカー

ブローカーが起動しています。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名

6101 Broker_start_failed
レベル: 警告
出所: 接続ブローカー

ブローカーの起動に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名
Success | Error	エラー・コード
Text	エラー・メッセージ

6102 Broker_running
レベル: 通知
出所: 接続ブローカー

ブローカーが実行されています。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名
Text	メッセージ文

6104 Broker_stopping
レベル: 通知
出所: 接続ブローカー

ブローカーを停止しています。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名

6106 Broker_reconfig_started
レベル: 通知
出所: 接続ブローカー

再構成が始まりました。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名

6108 Broker_reconfig_finished
レベル: 通知
出所: 接続ブローカー

再構成が終了しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Local username	ローカルのユーザ名
Success | Error	エラー・コード

6114 Broker_config_deprecated_element
レベル: 警告
出所: 接続ブローカー

ブローカーの設定に非推奨のエレメントが含まれています。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	イベントの説明

6200 Broker_tcp_connect
レベル: 情報
出所: 接続ブローカー

ブローカーの TCP 接続の試みが成功しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Src Port	転送元ポート
Local username	ローカルのユーザ名

6201 Broker_tcp_connect_failed
レベル: 警告
出所: 接続ブローカー

ブローカーの TCP 接続の試みが失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名
NIO error	NIO エラー

6204 Broker_transport_connect
レベル: 情報
出所: 接続ブローカー

TCP 経由でトランスポートが接続されました。

デフォルトのログ・ファシリティ: discard

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Remote username	リモートのユーザ名

引数	説明
Src Port	転送元ポート
Local username	ローカルのユーザ名
Session-Id	セッション ID

6206 Broker_transport_gateway_connect
レベル: 情報
出所: 接続ブローカー

ゲートウェイ・ハンドル経由でトランスポートが接続されました。

デフォルトのログ・ファシリティ: discard

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Remote username	リモートのユーザ名
Local username	ローカルのユーザ名
Session-Id	セッション ID

6208 Broker_connection_connect
レベル: 情報
出所: 接続ブローカー

ブローカーが Secure Shell 接続に成功しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名
Remote username	リモートのユーザ名
Uses gateway?	これはゲートウェイ・ハンドルを経由していますか
Session-Id	セッション ID

6209 Broker_connection_connect_failed
レベル: 警告
出所: 接続ブローカー

ブローカーが Secure Shell 接続に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名

引数	説明
Remote username	リモートのユーザ名
Uses gateway?	これはゲートウェイ・ハンドルを経由していますか
Session-Id	セッション ID
Text	エラー・コード

6210 Broker_connection_disconnect
レベル: 情報
出所: 接続ブローカー

ブローカーによって開始された Secure Shell 接続が切断されました。

デフォルトのログ・ファシリティ: discard

引数	説明
Local username	ローカル・ユーザ
Session-Id	セッションの識別子
Dst	接続先ホスト
Dst Port	接続先ポート
Remote username	リモートのユーザ名

6211 Broker_unknown_hostkey_accepted
レベル: 警告
出所: 接続ブローカー

設定により、* ブローカーがユーザに通知することなく未知のホスト鍵を受け入れました。*

デフォルトのログ・ファシリティ: normal

引数	説明
Text	鍵ダイジェスト
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名
Remote username	リモートのユーザ名
Text	SHA-256 鍵ダイジェスト

6212 Broker_new_hostkey
レベル: 警告
出所: 接続ブローカー

* サーバへの最初の接続、またはこのサーバのホスト鍵が以前に * 保存されたことがありません。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	鍵ダイジェスト

引数	説明
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名
Remote username	リモートのユーザ名
Text	SHA-256 鍵ダイジェスト

6213 Broker_hostkey_changed
レベル: 警告
出所: 接続ブローカー

* サーバのホスト鍵が、保存されているホスト鍵と異なっています。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	鍵ダイジェスト
Dst	接続先ホスト
Dst Port	接続先ポート
Local username	ローカルのユーザ名
Remote username	リモートのユーザ名
Text	SHA-256 鍵ダイジェスト

6301 Broker_userauth_failure
レベル: 警告
出所: 接続ブローカー

ユーザ認証に失敗しました。

デフォルトのログ・ファシリティ: normal

引数	説明
Text	理由
Session-Id	セッションの識別子

6302 Broker_userauth_method_success
レベル: 情報
出所: 接続ブローカー

ユーザ認証方法に成功しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Text	認証方法
Session-Id	セッションの識別子

6303 Broker_userauth_method_failure
レベル: 警告

出所: 接続ブローカー

ユーザ認証方法に失敗しました。

デフォルトのログ・ファシリティ: discard

引数	説明
Text	認証方法
Text	理由
Session-Id	セッションの識別子

6401 Connector_filter_rule
レベル: 情報
出所: 接続ブローカー

FTP_CAPTURE がトンネリングしていません

デフォルトのログ・ファシリティ: discard

引数	説明
Connector	Connector の動作
Dst	アドレス
Dst Port	ポート

付録F デフォルト及びサポートされる SSH アルゴリズム

ここでは Tectia 製品がサポートする SSH アルゴリズムについて説明します。

F.1. 暗号

表F.1 デフォルトの暗号 (クライアント側での優先順位による)

XMLでの名称	GUIでの名称	FIPS
crypticore128@ssh.com	CryptiCore (Tectia)	
aes128-gcm@openssh.com	AES-128-GCM (OpenSSH)	•
aes256-gcm@openssh.com	AES-256-GCM (OpenSSH)	•
AEAD_AES_128_GCM	AEAD_AES_128_GCM	•
AEAD_AES_256_GCM	AEAD_AES_256_GCM	•
aes128-ctr	AES-128-CTR	•
aes192-ctr	AES-192-CTR	•
aes256-ctr	AES-256-CTR	•

表F.2 サポートされる全ての暗号

XMLでの名称	GUIでの名称	FIPS
3des-cbc	3DES	•
AEAD_AES_128_GCM	AEAD_AES_128_GCM	•
AEAD_AES_256_GCM	AEAD_AES_256_GCM	•

XMLでの名称	GUIでの名称	FIPS
aes128-cbc	AES-128-CBC	•
aes128-ctr	AES-128-CTR	•
aes128-gcm@openssh.com	AES-128-GCM (OpenSSH)	•
aes192-cbc	AES-192-CBC	•
aes192-ctr	AES-192-CTR	•
aes256-cbc	AES-256-CBC	•
aes256-ctr	AES-256-CTR	•
aes256-gcm@openssh.com	AES-256-GCM (OpenSSH)	•
arcfour	Arcfour	
blowfish-cbc	Blowfish	
crypticore128@ssh.com	CryptiCore (Tectia)	
seed-cbc@ssh.com	SEED (Tectia)	
twofish128-cbc	Twofish-128	
twofish192-cbc	Twofish-192	
twofish256-cbc	Twofish-256	
twofish-cbc	Twofish	

F.2. 鍵交換アルゴリズム

表F.3 デフォルトの鍵交換 (クライアント側での優先順位による)

XMLでの名称	GUIでの名称	FIPS
mlkem1024nistp384-sha384	PQC: mlkem1024nistp384-sha384	•
mlkem768nistp256-sha256	PQC: mlkem768nistp256-sha256	•
mlkem768x25519-sha256	PQC: mlkem768x25519-sha256	•
ecdh-nistp521-kyber1024-sha512@ssh.com	PQC: ecdh-nistp521-kyber1024-sha512 (Tectia)	•
curve25519-frodokem1344-sha512@ssh.com	PQC: curve25519-frodokem1344-sha512 (Tectia)	•
sntrup761x25519-sha512@openssh.com	PQC: sntrup761x25519-sha512 (OpenSSH)	•
diffie-hellman-group-exchange-sha256	DH-GEX-SHA256	•
diffie-hellman-group16-sha512	DH-Group16-SHA512	•
diffie-hellman-group18-sha512	DH-Group18-SHA512	•
diffie-hellman-group14-sha256	DH-Group14-SHA256	•
diffie-hellman-group14-sha256@ssh.com	DH-Group14-SHA256 (Tectia)	•
curve25519-sha256	Curve25519-sha256	•

XMLでの名称	GUIでの名称	FIPS
curve25519-sha256@libssh.org	Curve25519-sha256 (libssh)	•

表F.4 サポートされる全ての鍵交換

XMLでの名称	GUIでの名称	FIPS
curve25519-frodokem1344-sha512@ssh.com	PQC: curve25519-frodokem1344-sha512 (Tectia)	•
curve25519-sha256	Curve25519-sha256	•
curve25519-sha256@libssh.org	Curve25519-sha256 (libssh)	•
curve448-kyber1024-sha512@ssh.com	PQC: curve448-kyber1024-sha512 (Tectia)	•
diffie-hellman-group14-sha1	DH-Group14-SHA1	•
diffie-hellman-group14-sha224@ssh.com	DH-Group14-SHA224 (Tectia)	•
diffie-hellman-group14-sha256	DH-Group14-SHA256	•
diffie-hellman-group14-sha256@ssh.com	DH-Group14-SHA256 (Tectia)	•
diffie-hellman-group15-sha256@ssh.com	DH-Group15-SHA256 (Tectia)	•
diffie-hellman-group15-sha384@ssh.com	DH-Group15-SHA384 (Tectia)	•
diffie-hellman-group16-sha384@ssh.com	DH-Group16-SHA384 (Tectia)	•
diffie-hellman-group16-sha512	DH-Group16-SHA512	•
diffie-hellman-group16-sha512@ssh.com	DH-Group16-SHA512 (Tectia)	•
diffie-hellman-group18-sha512	DH-Group18-SHA512	•
diffie-hellman-group18-sha512@ssh.com	DH-Group18-SHA512 (Tectia)	•
diffie-hellman-group1-sha1	DH-Group1-SHA1	
diffie-hellman-group-exchange-sha1	DH-GEX-SHA1	
diffie-hellman-group-exchange-sha224@ssh.com	DH-GEX-SHA224 (Tectia)	•
diffie-hellman-group-exchange-sha256	DH-GEX-SHA256	•
diffie-hellman-group-exchange-sha384@ssh.com	DH-GEX-SHA384 (Tectia)	•
diffie-hellman-group-exchange-sha512@ssh.com	DH-GEX-SHA512 (Tectia)	•
ecdh-sha2-nistp256	ECDH-NISTP256	•
ecdh-sha2-nistp384	ECDH-NISTP384	•
ecdh-sha2-nistp521	ECDH-NISTP521	•

XMLでの名称	GUIでの名称	FIPS
mlkem1024nistp384-sha384	PQC: mlkem1024nistp384-sha384	•
mlkem768nistp256-sha256	PQC: mlkem768nistp256-sha256	•
mlkem768x25519-sha256	PQC: mlkem768x25519-sha256	•
sntrup761x25519-sha512@openssh.com	PQC: sntrup761x25519-sha512 (OpenSSH)	•

F.3. メッセージ認証符号 (MAC)

表F.5 デフォルトのMAC (クライアント側での優先順位による)

XMLでの名称	GUIでの名称	FIPS
crypticore-mac@ssh.com	CryptiCore (Tectia)	
hmac-sha2-256	HMAC-SHA2-256	•
hmac-sha256-2@ssh.com	HMAC-SHA256-2 (Tectia)	•
hmac-sha2-512	HMAC-SHA2-512	•
hmac-sha512@ssh.com	HMAC-SHA512 (Tectia)	•
hmac-sha1	HMAC-SHA1	•

表F.6 サポートされる全てのMAC

XMLでの名称	GUIでの名称	FIPS
crypticore-mac@ssh.com	CryptiCore (Tectia)	
hmac-md5	HMAC-MD5	
hmac-md5-96	HMAC-MD5-96	
hmac-md5-96-etm@openssh.com	HMAC-MD5-96-ETM (OpenSSH)	
hmac-md5-etm@openssh.com	HMAC-MD5-ETM (OpenSSH)	
hmac-sha1	HMAC-SHA1	•
hmac-sha1-96	HMAC-SHA1-96	
hmac-sha1-96-etm@openssh.com	HMAC-SHA1-96-ETM (OpenSSH)	
hmac-sha1-etm@openssh.com	HMAC-SHA1-ETM (OpenSSH)	•
hmac-sha224@ssh.com	HMAC-SHA224 (Tectia)	
hmac-sha2-256	HMAC-SHA2-256	•
hmac-sha2-256-etm@openssh.com	HMAC-SHA2-256-ETM (OpenSSH)	•
hmac-sha2-512	HMAC-SHA2-512	•
hmac-sha2-512-etm@openssh.com	HMAC-SHA2-512-ETM (OpenSSH)	•
hmac-sha256@ssh.com	HMAC-SHA256 (Tectia/Old)	
hmac-sha256-2@ssh.com	HMAC-SHA256-2 (Tectia)	
hmac-sha384@ssh.com	HMAC-SHA384 (Tectia)	
hmac-sha512@ssh.com	HMAC-SHA512 (Tectia)	

F.4. ホスト鍵及び公開鍵署名アルゴリズム

表F.7 デフォルトのホスト鍵アルゴリズム (クライアント側での優先順位による)

XMLでの名称	GUIでの名称	FIPS
rsa-sha2-512	rsa-sha2-512	•
rsa-sha2-256	rsa-sha2-256	•
ssh-rsa-sha256@ssh.com	ssh-rsa-sha256 (Tectia)	•
ecdsa-sha2-nistp521	ecdsa-sha2-nistp521	•
ecdsa-sha2-nistp384	ecdsa-sha2-nistp384	•
ecdsa-sha2-nistp256	ecdsa-sha2-nistp256	•
x509v3-sign-rsa-sha256@ssh.com	x509v3-sign-rsa-sha256 (Tectia)	•
x509v3-ecdsa-sha2-nistp256	x509v3-ecdsa-sha2-nistp256	•
x509v3-ecdsa-sha2-nistp384	x509v3-ecdsa-sha2-nistp384	•
x509v3-ecdsa-sha2-nistp521	x509v3-ecdsa-sha2-nistp521	•
x509v3-rsa2048-sha256	x509v3-rsa2048-sha256	•
ssh-ed25519	ssh-ed25519	•
ecdsa-sha2-nistp256-cert-v01@openssh.com	ecdsa-sha2-nistp256-cert-v01@openssh.com	•
ecdsa-sha2-nistp384-cert-v01@openssh.com	ecdsa-sha2-nistp384-cert-v01@openssh.com	•
ecdsa-sha2-nistp521-cert-v01@openssh.com	ecdsa-sha2-nistp521-cert-v01@openssh.com	•
ssh-ed25519-cert-v01@openssh.com	ssh-ed25519-cert-v01@openssh.com	•
rsa-sha2-256-cert-v01@openssh.com	rsa-sha2-256-cert-v01@openssh.com	•
rsa-sha2-512-cert-v01@openssh.com	rsa-sha2-512-cert-v01@openssh.com	•

表F.8 サポートされる全てのホスト鍵と公開鍵署名アルゴリズム

XMLでの名称	GUIでの名称	FIPS
ecdsa-sha2-nistp256	ecdsa-sha2-nistp256	•
ecdsa-sha2-nistp256-cert-v01@openssh.com	ecdsa-sha2-nistp256-cert-v01@openssh.com	•
ecdsa-sha2-nistp384	ecdsa-sha2-nistp384	•
ecdsa-sha2-nistp384-cert-v01@openssh.com	ecdsa-sha2-nistp384-cert-v01@openssh.com	•
ecdsa-sha2-nistp521	ecdsa-sha2-nistp521	•
ecdsa-sha2-nistp521-cert-v01@openssh.com	ecdsa-sha2-nistp521-cert-v01@openssh.com	•
rsa-sha2-256	rsa-sha2-256	•
rsa-sha2-256-cert-v01@openssh.com	rsa-sha2-256-cert-v01@openssh.com	•

XMLでの名称	GUIでの名称	FIPS
rsa-sha2-512	rsa-sha2-512	•
rsa-sha2-512-cert-v01@openssh.com	rsa-sha2-512-cert-v01@openssh.com	•
ssh-dss	ssh-dss	
ssh-dss-cert-v01@openssh.com	ssh-dss-cert-v01@openssh.com	
ssh-dss-sha224@ssh.com	ssh-dss-sha224 (Tectia)	•
ssh-dss-sha256@ssh.com	ssh-dss-sha256 (Tectia)	•
ssh-dss-sha384@ssh.com	ssh-dss-sha384 (Tectia)	•
ssh-dss-sha512@ssh.com	ssh-dss-sha512 (Tectia)	•
ssh-ed25519	ssh-ed25519	•
ssh-ed25519-cert-v01@openssh.com	ssh-ed25519-cert-v01@openssh.com	•
ssh-rsa	ssh-rsa	
ssh-rsa-cert-v01@openssh.com	ssh-rsa-cert-v01@openssh.com	
ssh-rsa-sha224@ssh.com	ssh-rsa-sha224 (Tectia)	•
ssh-rsa-sha256@ssh.com	ssh-rsa-sha256 (Tectia)	•
ssh-rsa-sha384@ssh.com	ssh-rsa-sha384 (Tectia)	•
ssh-rsa-sha512@ssh.com	ssh-rsa-sha512 (Tectia)	•
x509v3-ecdsa-sha2-nistp256	x509v3-ecdsa-sha2-nistp256	•
x509v3-ecdsa-sha2-nistp384	x509v3-ecdsa-sha2-nistp384	•
x509v3-ecdsa-sha2-nistp521	x509v3-ecdsa-sha2-nistp521	•
x509v3-rsa2048-sha256	x509v3-rsa2048-sha256	•
x509v3-sign-dss	x509v3-sign-dss	
x509v3-sign-dss-sha224@ssh.com	x509v3-sign-dss-sha224 (Tectia)	•
x509v3-sign-dss-sha256@ssh.com	x509v3-sign-dss-sha256 (Tectia)	•
x509v3-sign-dss-sha384@ssh.com	x509v3-sign-dss-sha384 (Tectia)	•
x509v3-sign-dss-sha512@ssh.com	x509v3-sign-dss-sha512 (Tectia)	•
x509v3-sign-rsa	x509v3-sign-rsa	
x509v3-sign-rsa-sha224@ssh.com	x509v3-sign-rsa-sha224 (Tectia)	•
x509v3-sign-rsa-sha256@ssh.com	x509v3-sign-rsa-sha256 (Tectia)	•
x509v3-sign-rsa-sha384@ssh.com	x509v3-sign-rsa-sha384 (Tectia)	•
x509v3-sign-rsa-sha512@ssh.com	x509v3-sign-rsa-sha512 (Tectia)	•
x509v3-ssh-dss	x509v3-ssh-dss	
x509v3-ssh-rsa	x509v3-ssh-rsa	

付録G Tectia Client からの OpenSSL の削除

G.1. 背景情報

G.1.1. OpenSSL ライブラリは削除した方が良いですか?

FIPS 準拠モードで Tectia Client を使用しない場合(詳細については、 3.6 を参照)、 OpenSSL 暗号化ライブラリは不要なので、削除して問題ありません。

FIPS モードを使用しておらず、Tectia Client のインストールから OpenSSL を削除する場合は、後述の項で説明しているプラットフォームごとの方法に従ってください。

G.1.2. OpenSSL ライブラリを削除するとどうなりますか?

OpenSSL 暗号化ライブラリを削除すると、Tectia Client が FIPS 準拠モードで動作するように設定されている場合、起動が拒否されるようになります。

G.2. OpenSSL 暗号化ライブラリの削除

G.2.1. Linux と Solaris

Unix で OpenSSL 暗号化ライブラリを Tectia Client から削除するには、もし設定で有効になっている場合は初めに FIPS モードを停止してください。

以下のコマンドで penSSL FIPS ライブラリを削除します。

```
# /opt/tectia/sbin/ssh-modeset fips-mode off
```

```
# /opt/tectia/sbin/ssh-modeset fips-remove
```

G.2.2. Windows

Windows で OpenSSL 暗号化ライブラリを Tectia Client から削除するには、もし設定 GUI で有効になっている場合は初めに FIPS モードを停止してください。

注意

Tectia Client と Tectia Server の両方がインストールされている場合は、ユーザ固有の 接続ブローカー 設定で FIPS モードが無効になっていることとシステム全体の Tectia FIPSMODE スイッチファイルが削除されていることを確認してください。FIPSMODE ファイルは Tectia Server 設定 GUI から FIPS モードが無効にされると自動的に削除されます。 (詳しくは、 3.6).

Windows で Tectia のオプション・インストール・モジュールを変更します

Windows 環境では、以下の手順で Tectia Client 及び Server オプション FIPS モジュールを変更します。

1. Windows の [**スタート**] メニューから、 [**コントロール パネル**] を開き [**プログラムと機能**] をクリックします。
2. インストールされたプログラムの一覧から **Tectia Client** を選択し [**変更**] をクリックします。
3. インストーラで [**Modify**] をクリックします。
4. [**Tectia Client**] > [**FIPS**] オプション・モジュールを選択し、 [Entire feature will be unvailable] に変更します。
5. [**Next**] 及び [**Install**] をクリックして Tectia の FIPS サポートモジュールを削除する インストールの変更 を続行します。

Windows では、 FIPS サポート・モジュールが削除されると Tectia Client から OpenSSL ファイルが削除されます。

`<INSTALLDIR>` は 64ビット Windows バージョンでのインストール・ディレクトリを示します: `C:\Program Files (x86)\SSH Communications Security\SSH Tectia`

- `<INSTALLDIR>\SSH Tectia AUX\Plugins\<x>.<y>.<z>.<b>\sshcrypto1.dll`

  (`<x>`、`<y>`、`<z>`、及び `<b>` は Tectia Client のバージョン及びビルド番号を示しています (例: `6.6.5.123`))

- `<INSTALLDIR>\SSH Tectia AUX\fips\fips.dll`
- `<INSTALLDIR>\SSH Tectia AUX\fips\openssl.cnf`

- <INSTALLDIR>\SSH Tectia AUX\libcrypto-3.dll
- <INSTALLDIR>\SSH Tectia Broker\libcrypto-3.dll
- <INSTALLDIR>\SSH Tectia Client\libcrypto-3.dll

付録H オープン・ソース・ソフトウェアのライセンス謝辞

SSH Communications Security Corporation は、Tectia クライアント/サーバ・ソリューションで使用されている以下のオープン・ソース・ソフトウェアに謝意を表します。

BSD Software

> This product includes software developed by the University of California, Berkeley and its contributors.

DES

> This product includes software developed by Eric Young `eay@cryptsoft.com`.

ICU

> Copyright © 1995-2016 International Business Machines Corporation and others
>
> All rights reserved.
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

OpenSSL

OpenSSL 3.0 is licensed under the Apache License 2.0.

Apache License Version 2.0, January 2004 https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR

PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

PCRE

This software includes PCRE library. Copyright © 1997-2015 University of Cambridge. All rights reserved.

C++ wrapper functions. Copyright © 2007-2015, Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

XFree86

This Software contains portions of XFree86 software and the delivery of XFree86 software or portions of the said software is subject to the acknowlegement of the following copyright notice and permission notice of The Open Group:

Copyright © 1988, 1998 The Open Group

Permission to use, copy, modify, distribute, and sell XFree86 software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation.

THE XFREE86 SOFTWARE IS PROVIDE "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE XFREE86 SOFTWARE OR THE USE OR OTHER DEALINGS IN THE XFREE86 SOFTWARE.

Except as contained in this notice, the name of The Open Group shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from The Open Group.

ZLIB

This software incorporates zlib data compression library by Jean-loup Gailly and Mark Adler.

liboqs

Licensed under MIT. Copyright (c) 2016-2021 Open Quantum Safe project.

liboqs includes some third party libraries or modules that are licensed differently, including:

- BSD 3-Clause License
- Apache License v2.0
- public domain

- BSD-like CRYPTOGAMS license
- CC0
- Custom license for rand_nist.c:

  Created by Bassham, Lawrence E (Fed) on 8/29/17.

  Copyright © 2017 Bassham, Lawrence E (Fed). All rights reserved.

  NIST-developed software is provided by NIST as a public service. You may use, copy, and distribute copies of the software in any medium, provided that you keep intact this entire notice. You may improve, modify, and create derivative works of the software or any portion of the software, and you may copy and distribute such modifications or works. Modified works should carry a notice stating that you changed the software and should note the date and nature of any such change. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

  NIST-developed software is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED, IN FACT, OR ARISING BY OPERATION OF LAW, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, AND DATA ACCURACY. NIST NEITHER REPRESENTS NOR WARRANTS THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED. NIST DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE SOFTWARE OR THE RESULTS THEREOF, INCLUDING BUT NOT LIMITED TO THE CORRECTNESS, ACCURACY, RELIABILITY, OR USEFULNESS OF THE SOFTWARE.

  You are solely responsible for determining the appropriateness of using and distributing the software and you assume all risks associated with its use, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and the unavailability or interruption of operation. This software is not intended to be used in any situation where a failure could cause risk of injury or damage to property. The software developed by NIST employees is not subject to copyright protection within the United States.

  SPDX-License-Identifier: Unknown

  Modified for liboqs by Douglas Stebila

索引

シンボル

A

C

D

E

F

G

H

I

J

K

L

M

O

P

R

S

T

U

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

よ

ら

り

れ

ろ

わ