ユーザ ガイド

Reflection for Secure IT
Client for Windows

バージョン 8.1

〔 02 版: 2019 年 10 月 31 日 〕

#JMN-02219E-0219J

目次

1. はじめに

1.1 対象製品

本資料は、Windows 用 SSH クライアント製品：**Reflection for Secure IT Client for Windows 8.1** および そのアップデート版 **8.1 Update1/ Update2** について、その仕様と使い方について記述したマニュアルです。

[注記]：

「Reflection for Secure IT」を 以下「RSIT」と省略します。

RSIT 製品には、Windows 向けと UNIX/Linux 向けそれぞれにサーバ版とクライアント版が存在します。

- RSIT Server for Windows
- RSIT Server for UNIX
- **RSIT Client for Windows** << 本製品
- RSIT Client for UNIX

[注記]：

本マニュアルの記述や業務関連資料上、以下の表現が使われますが、皆 同一とお考え下さい。

「RSIT Client for Windows」, 「RSIT Client Windows」,「RSIT Windows Client」,

「RSIT Windows クライアント」,「RSIT CW」等

[注記]：

プログラム上の表示や関連フォルダ内の使用社名について：

かつて"WRQ 社"、"Attachmate 社" と変遷し、バージョン 8.0 から現 Micro Focus 社となりました。

1.2 動作環境

(1) サポート OS

- Windows 10 (x86, x86-64)
- Windows 8, 8.1 (x86, x86-64)
- Windows 7 (x86, x86-64)
- Windows Vista SP2 (x86, x86-64)
- Windows Server 2016 (x86-64)
- Windows Server 2012 R2 (x86-64)
- Windows Server 2012 (x86-64)
- Windows Server 2008 R2 (x86-64)
- Windows Server 2008 (x86, x86-64)
- Citrix XenApp/Presentation Server
- 仮想環境 (VMWare, Microsoft Virtual PC)

[注記]：

動作条件とは別に、脆弱性対策の観点から、最新 SP(サービスパック)や最新 Windows アップデートを適用することを強く推奨します。

2. 概要説明

2.1 導入について

《インストールプログラムファイル名称》：

バージョン	インストールプログラムファイル名称	
8.1　　(8.1 本体)	"rsit-8.1-prod-w32.exe"	64bit/32bit OS 共通
8.1 Update1 (更新差分)	"rsitwincli-8.1.0.731-update-w32.exe"	64bit/32bit OS 共通
8.1 Update2 (更新差分)	"rsitwincli-8.1.754-update-w32.exe"	64bit/32bit OS 共通

《インストール操作上のポイント》：

インストールは、管理者権限を持つローカルユーザアカウントを使い、
インストールプログラムファイルを PC/サーバ上のワークフォルダに 適宜コピーし、起動します。
起動後は表示ウィザード画面内容に従い操作を進め完了します。
更新プログラム 8.1 Update1 は インストール済みの 8.1 に対し上書き適用する形で、更新プログラム 8.1 Update2 は インストール済みの 8.1 あるいは 8.1 Update1 に対し上書き適用する形で導入します。

《表示ウィザード画面上での対応》：

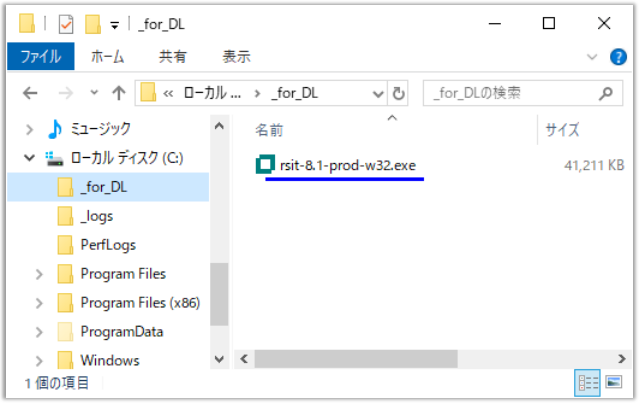
インストール開始後のウィザード画面(*)において、情報入力欄が現れます。ここでは、基本何も入力せずに[インストール]ボタンを押下し先へ進めて下さい。

(*)後述《8.1 インストール操作 表示画面と操作例》⑤各種指定画面の表示 が該当

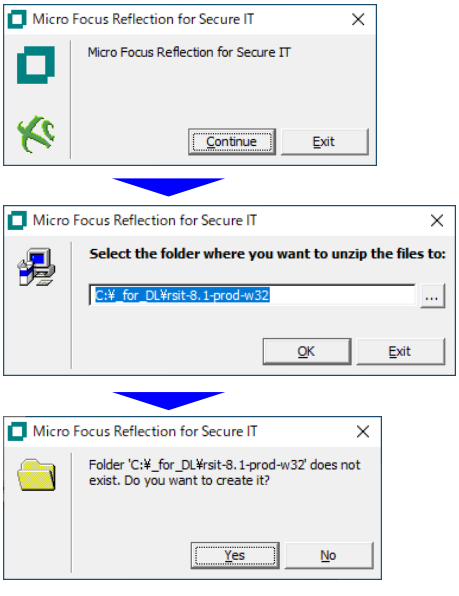
タブの表記	対応操作
[ユーザ情報]	入力不要。(入力内容による動作/表示上への影響は何もありません。)
[ファイルの場所]	基本不要。個別理由によりインストール先を変更する場合にのみ指定します。 既定先 = "C:¥Program Files (x86)¥Micro Focus¥RSecure"
[機能の選択]	操作不要。
[詳細設定]	基本不要。管理者向けのインストールイメージ生成用に使用します。

《 8.1 インストール操作 表示画面と操作例 》：

① インストールプログラムの起動

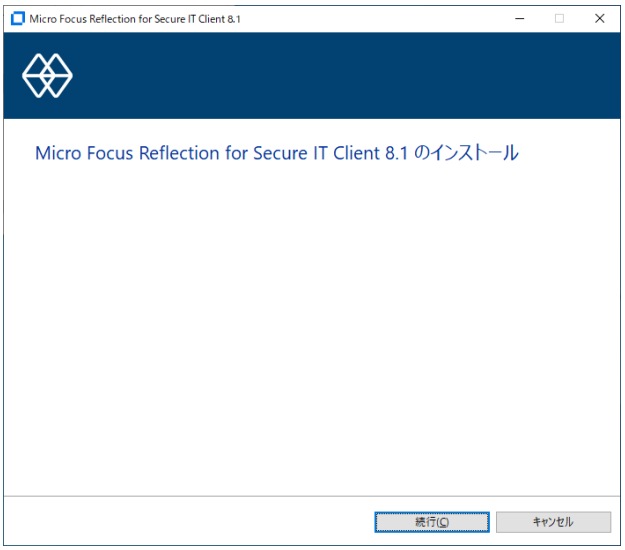


② 表示ガイダンス画面の確認と進行



③ インストール開始画面

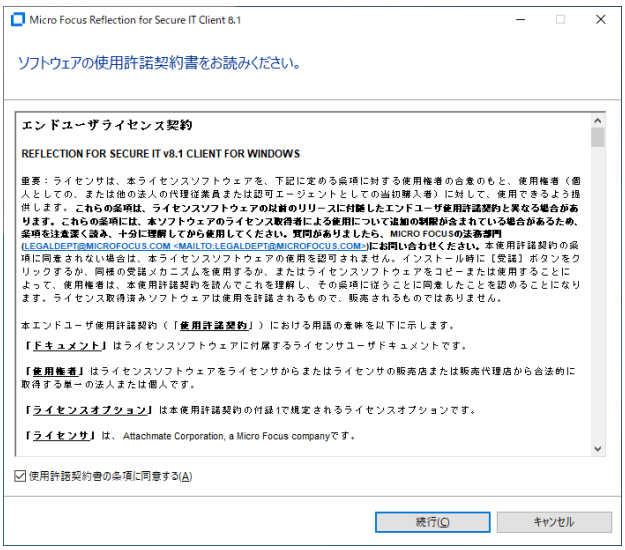
⇒［続行］ボタン押下



④ ソフトウェア使用許諾契約書の表示

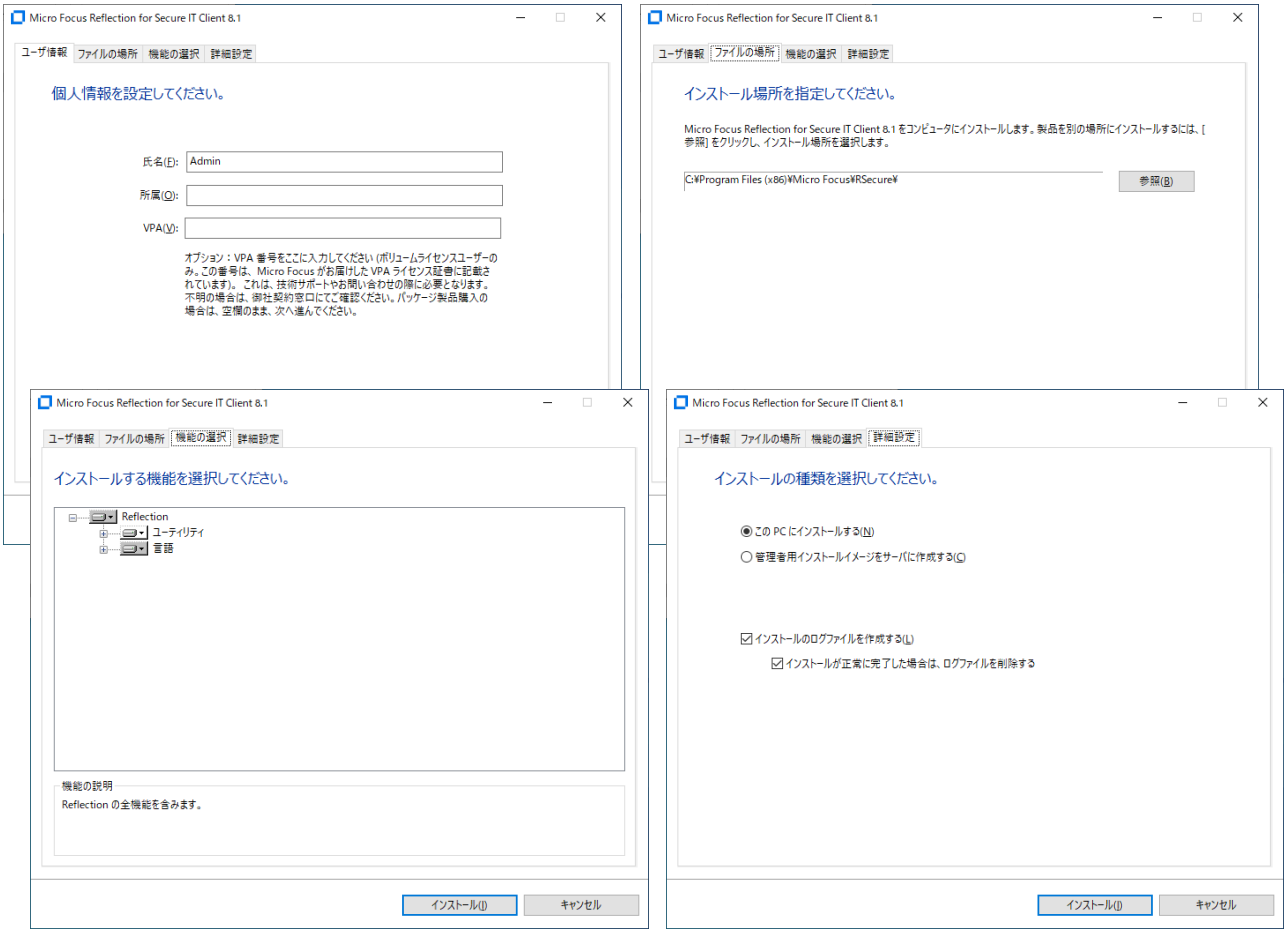
⇒”使用許諾契約書の条項に同意する”をチェック、

［続行］ボタン押下

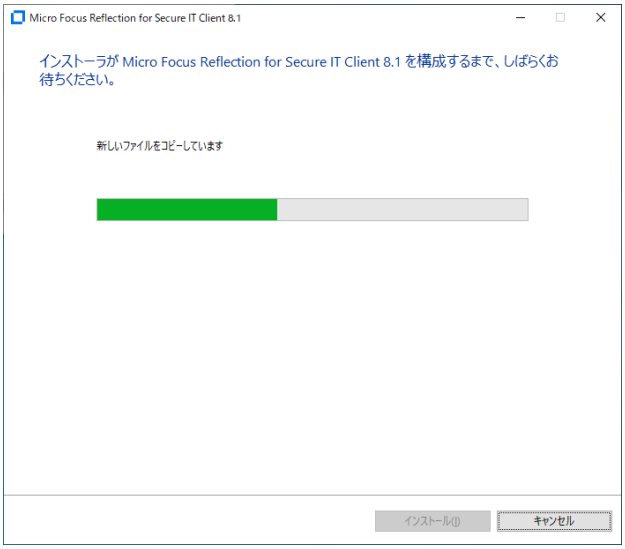


⑤ 各種指定画面の表示

⇒ 内容確認の上、［インストール］ボタン押下　～前述《表示ウィザード画面上での対応》を参照）

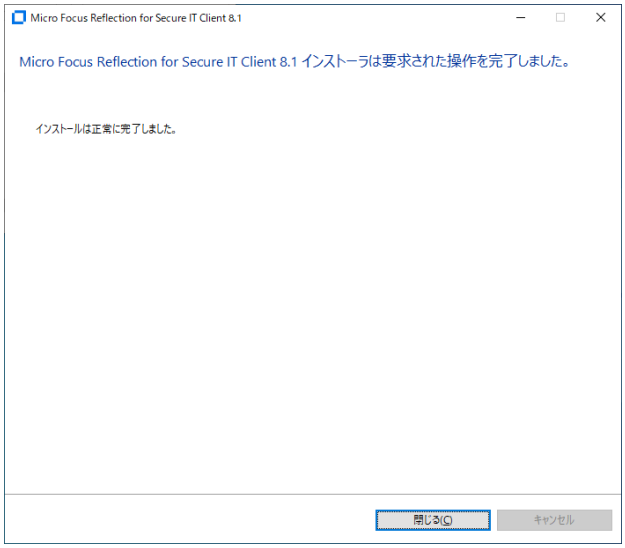


⑥ インストール処理中の表示



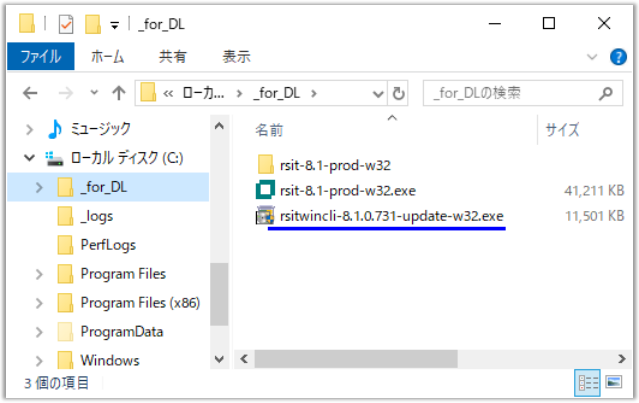
⑦ インストール完了の表示

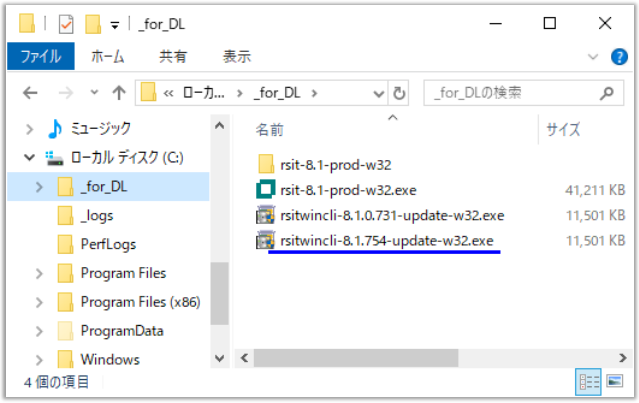
⇒［閉じる］ボタンを押下し終了



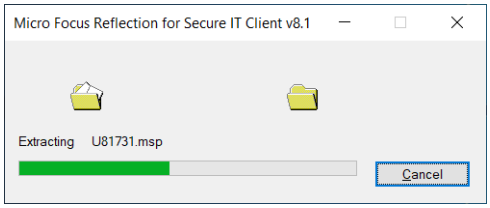
《 8.1 Update1/ Update2 適用操作 表示画面と操作例 》：

① 更新適用プログラム(8.1 Update1)　あるいは　更新適用プログラム(8.1 Update2) の起動





② プログラム展開進行の表示

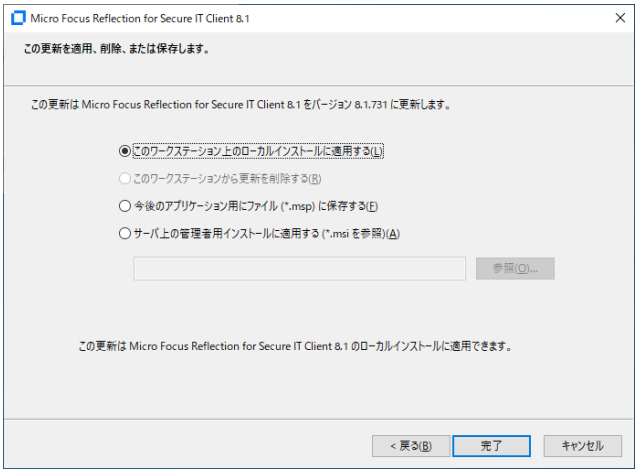


③ ソフトウェア使用許諾契約書の表示

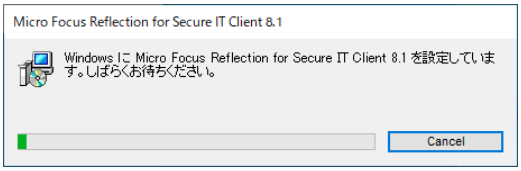
⇒"使用許諾契約書の条項に同意する"をチェック、[次へ]ボタン押下

④ 更新適用確認画面の表示

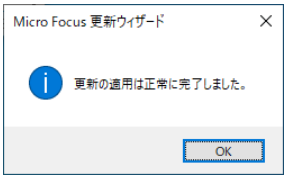
⇒[完了] ボタン押下



⑤ 更新適用処理中の表示



⑥ 更新適用完了の表示



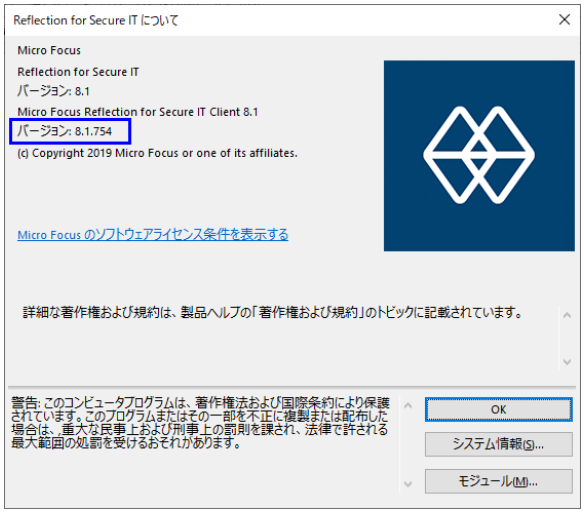
⑦ Update 適用の確認

⇒[Reflection for Secure IT]画面上 "ヘルプ"メニュー > "Reflection のバージョン情報" を選択表示

〈Update 適用前(= 8.1)〉 バージョン 8.1.703

〈Update1 適用後〉 バージョン 8.1.731

〈Update2 適用後〉 バージョン 8.1.754



2.2 使い方の概要について

ssh/sftp クライアントである本製品は、接続先 ssh/sftp サーバとの間で安全な通信路を確立し、次の用途/機能を実現します。

用途	機能	使い方(起動区分)	
		コマンド	GUI 画面操作
ファイル転送	sftp クライアント	sftp (sftp.exe)	Reflection FTP クライアント
	scp クライアント	scp (scp.exe)	—
リモート アクセス	ssh ターミナル接続	ssh (ssh.exe)	SSH クライアント
	ssh リモートコマンド	ssh + 〈コマンド〉	—
	トンネリング (TCP ポート転送)	ssh + 〈設定/オプション〉	SSH クライアント + 〈設定〉

《コマンド書式 概要》 :　　～ 詳細は、「5.2 コマンド」を参照

・ssh (ユーザ名)@(接続先サーバ)

・sftp (ユーザ名)@(接続先サーバ)　　← 対話型の場合

sftp -B "コマンドファイル" (ユーザ名)@(接続先サーバ)　　← バッチ処理の場合

・scp (コピー元) (コピー先)

(コピー元),(コピー先)の一方に「"ローカルファイルパス"」、

もう一方に「(ユーザ名)@(接続先サーバ):"リモートファイルパス"」を指定します。

＊(接続先サーバ)は、"IP アドレス"か"ホスト名"を指定します。

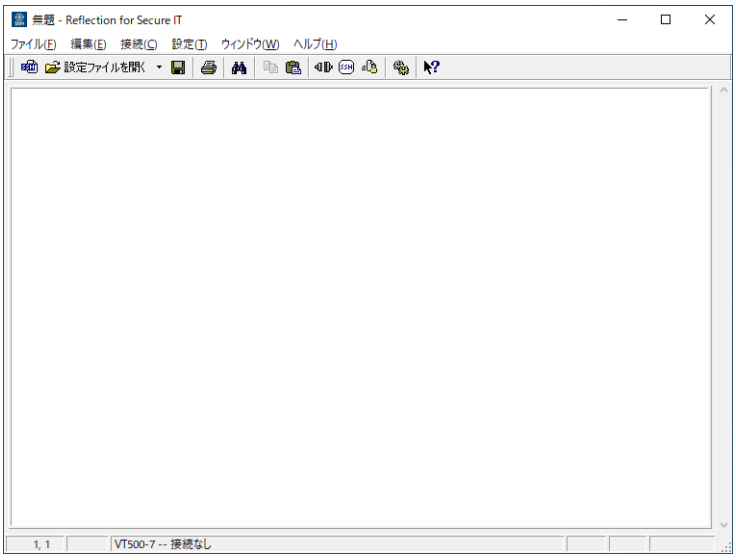
《GUI 画面操作 概要》 :

1) [Reflection for Secure IT] ターミナル画面

a)メイン画面の開き方

・Windows スタートから、「Micro Focus Reflection」>「SSH クライアント」を選択し起動

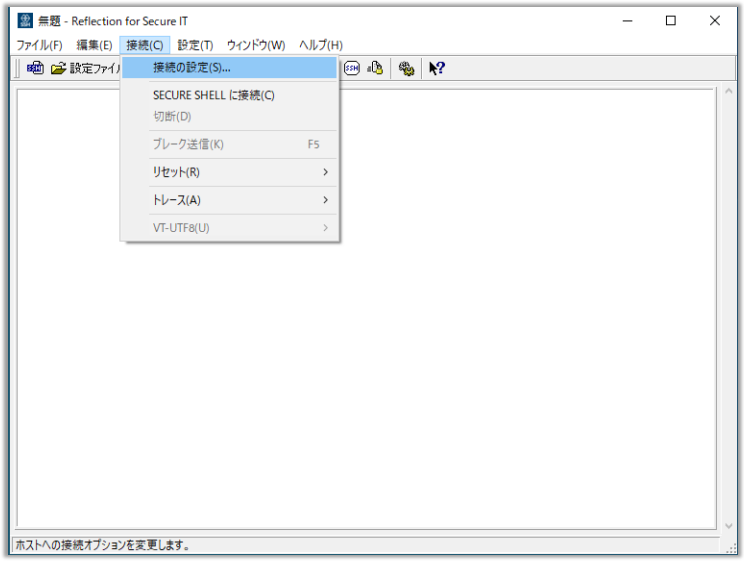
⇒ [Reflection for Secure IT] ターミナル画面を表示

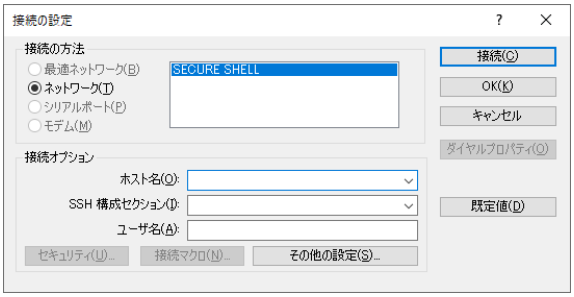


b) 接続画面の開き方

・[Reflection for Secure IT] ターミナル画面において、

"接続"メニュー > "接続の設定"を選択　⇒ [接続の設定]画面を表示

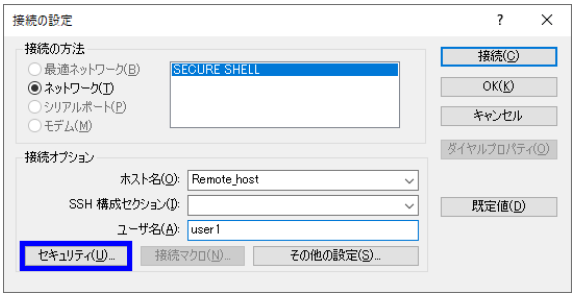


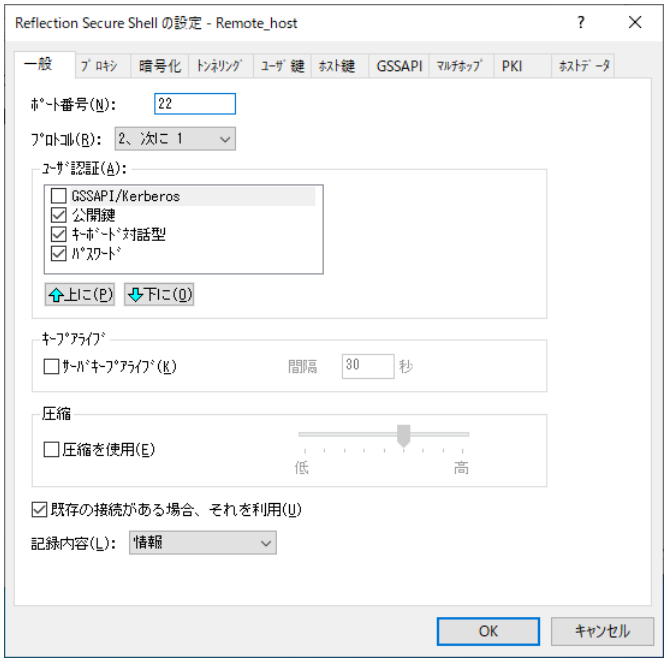


c) 設定画面の開き方

・[接続の設定]画面において、「ホスト名」と「ユーザ名」に名称入力

・有効化した[セキュリティ]ボタンを押下　⇒ [Reflection Secure Shell の設定]画面表示



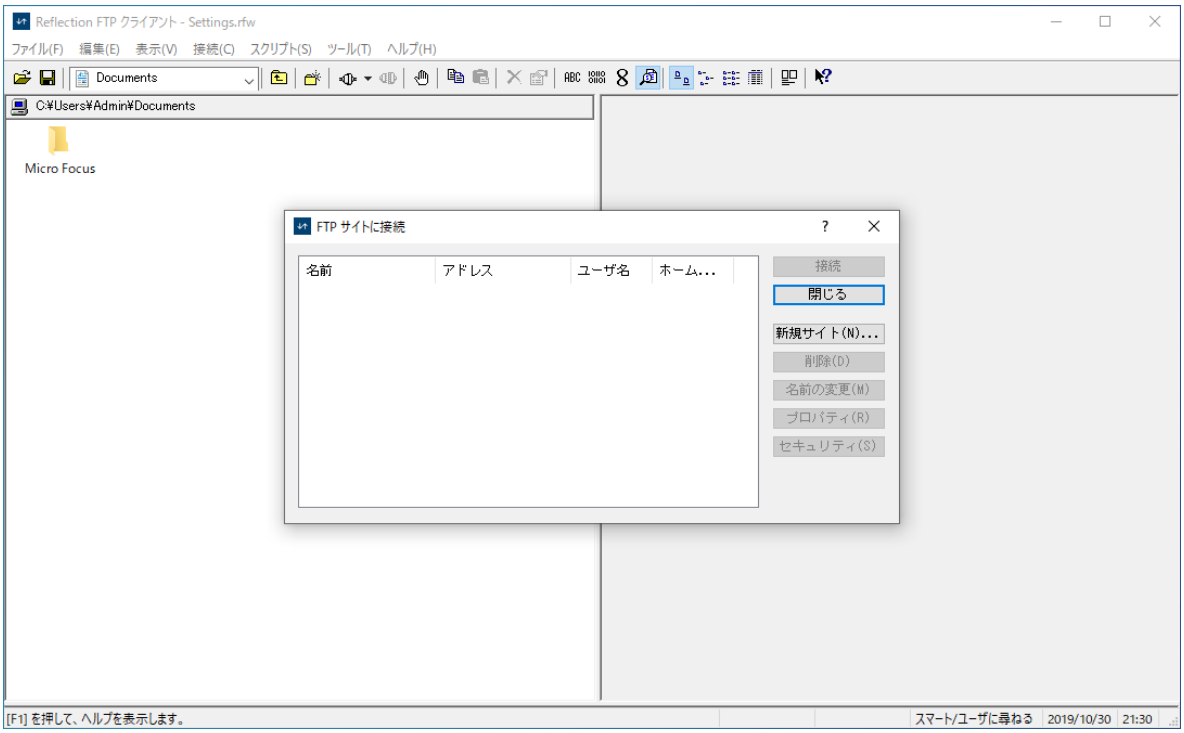


2) [Reflection FTP クライアント] sftp GUI 画面

a) メイン画面の開き方

・Windows スタートから、「Micro Focus Reflection」>「FTP クライアント」を選択し起動

⇒ [Reflection FTP クライアント] sftp GUI 画面を表示



b) 接続画面 および 設定画面の開き方 (新規接続時)

・[FTP サイトに接続]画面において″、[新規サイト]ボタンを押下

・表示ウィザード画面に接続情報を順次入力 (①～④)

・設定画面は、途中 あるいは 今すぐ接続せずに完了後に [セキュリティ]ボタン押下し、
表示[セキュリティのプロパティ]画面内の [構成]ボタン押下にて
表示[Reflection Secure Shell の設定]画面において設定情報を指定可(⑥,⑦)

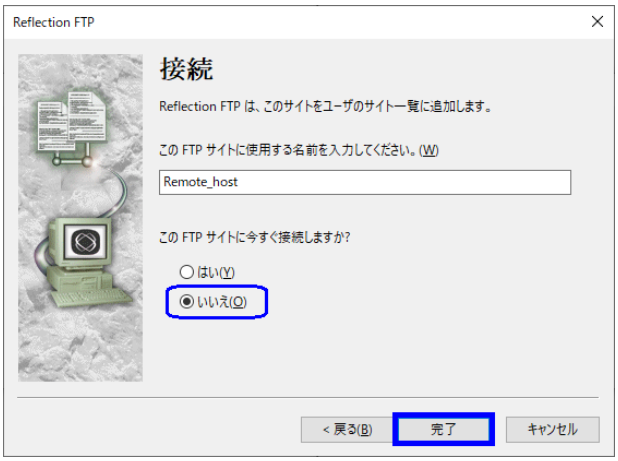
①ウィザード画面 1

②ウィザード画面 2

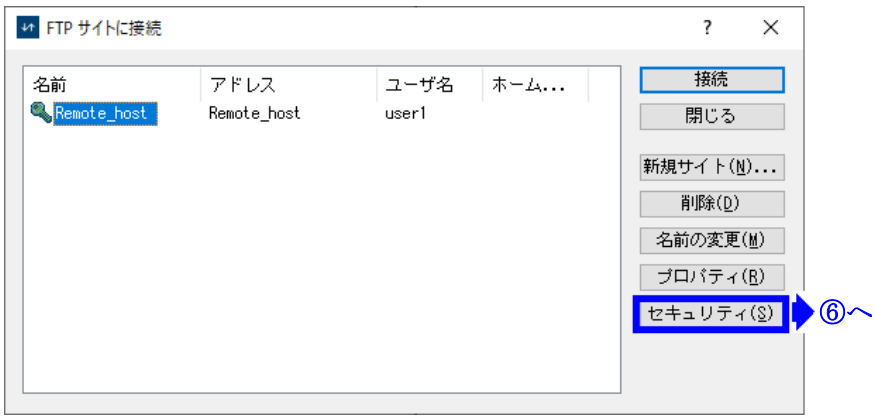
③ウィザード画面 3

④ウィザード画面 4

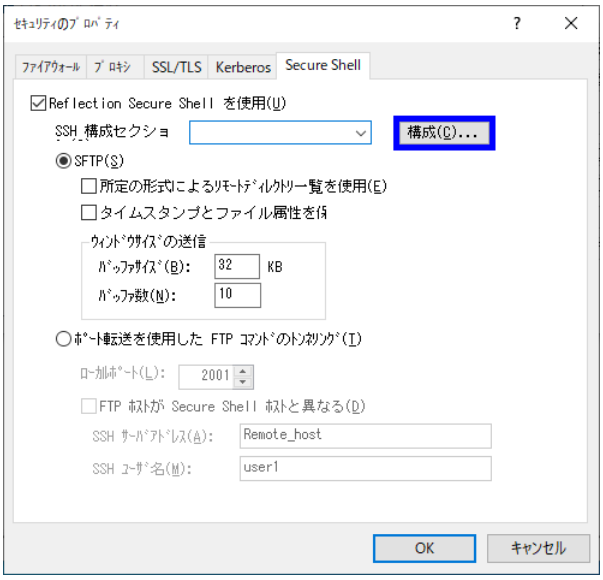
～今すぐ接続せず(「いいえ」選択し)[完了]



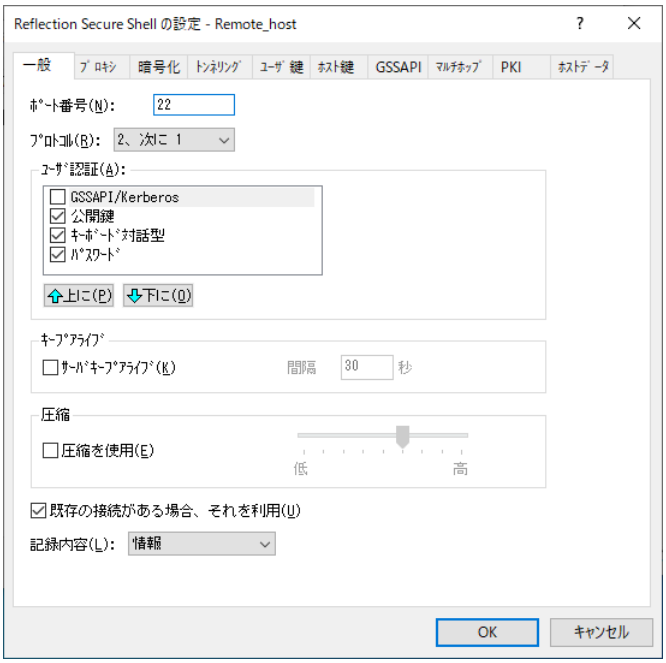
⑤欄内に指定接続先を登録



⑥[セキュリティのプロパティ]画面



⑦[Reflection Secure Shell の設定]画面



2.3 設定について

各種コマンド(ssh, sftp, scp)や GUI 画面からの接続は、設定ファイル[注 1]に従い接続動作を実行します。

[注 1]:

以後「構成ファイル」と呼びます。

《構成ファイルの種類》:

a)ユーザ個別:"config"ファイル

- 適用先:該当するログオンユーザによる起動接続動作にのみ反映
- 所在　:"C:¥Users¥(ユーザ名)¥Documents¥Micro Focus¥Reflection¥.ssh"フォルダ下
- 編集　:GUI 設定画面上の既定値からの変更操作時にプログラムが自動的に反映保存。
  直接手操作により編集することも可。

b)全体共通:"ssh_config"ファイル

- 適用先:PC/サーバ内の("SYSTEM"ユーザ含む)全実行ユーザによる起動接続動作に反映
- 所在　:"C:¥ProgramData¥Micro Focus¥Reflection"フォルダ下
- 編集　:ファイル自体の新規作成含め、手操作により編集要

[注 2]:

フォルダ下に既に参考用の"ssh_config_examples"ファイルが存在しますが、それは使わずに、接続実績あるユーザ個別"config"ファイルと"known_hosts"ファイルをコピーし、それぞれ"config"⇒"ssh_config"、"known_hosts"⇒"ssh_known_hosts"にリネーム保存し、"ssh_config"ファイルを別途編集する方が堅実で容易です。

《構成ファイルの記述仕様》:

接続先サーバ毎に「SSH 構成セクション」と呼ぶ単位でブロック化し指定します。

```
Host (接続先サーバ A)
   設定内容                    ‥「SSH 構成セクション」
   ....
#EndHost
Host (接続先サーバ B)
   設定内容                    ‥「SSH 構成セクション」
   ....
#EndHost
```

Host に続く(接続先サーバ)には、コマンドや GUI 接続画面上に記述する(接続先サーバ)すなわち"IP アドレス"か"ホスト名"を記載します。

[注 3]:

Host に続く(接続先サーバ)名称は文字列にて合致判定をしますので、"IP アドレス"と"ホスト名"が名前解決により実質同じであったとしても、検索処理上はコマンドや GUI 接続画面上に記述する(接続先サーバ)名称と同一にする必要があります。また、大文字/小文字は識別判定されます。

《設定優先順位》：

設定内容は、前述"ユーザ個別構成ファイル"と"全体共通構成ファイル"の他にコマンドのオプション指定も可能です。同じ設定項目が複数指定箇所にて重複存在した場合は、次の優先順位に従い適用されます。

［高優先］ **［低優先］**

コマンドオプション指定 ＞ ユーザ個別構成ファイル ＞ 全体共通構成ファイル

《GUI 接続画面上の保存ファイル》：

GUI 接続画面を起動し使用終了する場合に、構成ファイルとの連携に加え、次の付加情報が合わせて保存情報として保存され、次回以降の GUI 接続画面による接続時に、同一条件下での再利用を可能とします。

	ファイル名称と保存単位	設定保存内容
[Reflection for Secure IT] ssh ターミナル画面	「****.r3w」 "SSH 構成セクション"単位	接続情報、端末エミュレーション、表示設定、キーの割り当て、マウスの構成
[Reflection FTP クライアント] sftp GUI 画面	「****.rfw」 手元ログオンユーザで一つ	接続情報、ディレクトリ表示設定、転送設定

3. 関連技術の解説

本章では、本製品を正しく使用頂く上で前提となる基本的な関連技術について解説します。

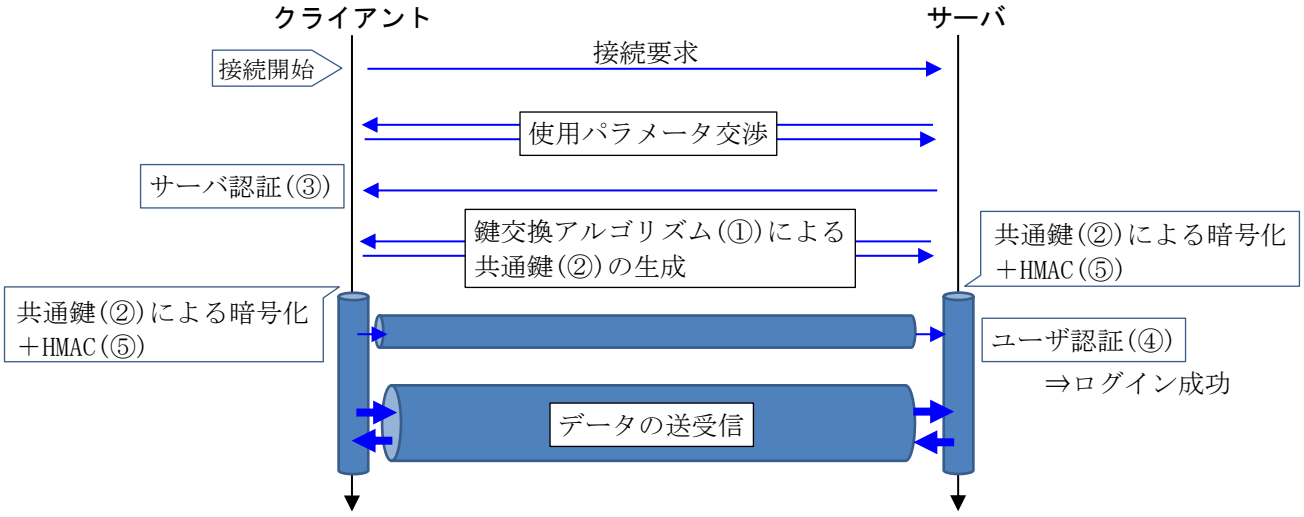
3.1 SSH の概要

SSH(Secure Shell)は、リモート接続先サーバへの接続を、暗号化や認証によりネットワーク上の不正行為リスクから安全に保護するために規定された通信プロトコルです。サーバとクライアントで構成します。

不正行為リスク	対応	関連技術
盗聴	暗号化	①鍵交換アルゴリズム
		②共通鍵暗号アルゴリズム
なりすまし	認証	③サーバ認証
		④ユーザ認証（パスワード認証,公開鍵認証 等）
改ざん	データ完全性保証	⑤HMAC[ハッシュ関数によるメッセージ認証符号]

サーバ側	サーバサービスがクライアントからの接続を待ち受けます。各種機能やサービスを提供し、設定によりクライアントからの接続条件を規定します。
クライアント側	接続開始の起点となります。ユーザインタフェースを提供し、各 GUI 接続画面やコマンドからの入力内容をサーバへ送信し、その結果と状況を表示提供します。

《接続開始時の処理フローイメージ図》：



《SSH 具備機能と生まれた背景》：

SSH としての具備機能	生まれた背景
ssh ターミナル接続	平文 telnet/rlogin 接続等の置き換え
⇒〔セキュアに確立した共通 ssh 通信路をチャネル多重し活用〕	
sftp ファイル転送	平文 FTP ファイル転送の置き換え(類似コマンドを用意)
scp ファイル転送	平文 rcp コマンドの置き換え(類似コマンドを用意)
トンネリング(TCP ポート転送)	TCP アプリ通信や X11 通信をセキュアに保護
ssh リモートコマンド	サーバ上のリモートコマンドを起動

3.2 暗号化技術

ssh 接続では、各種暗号化技術の特性を活かし前述セキュアな通信環境の確立を実現しています。

《各種暗号化技術》：

a)共通鍵暗号アルゴリズム ＝ 対称暗号系

・同一の「共通鍵」を使い、送信データを暗号化し、受信データを復号化します。

〈長所〉・暗号化/復号化処理の計算量が小

〈短所〉・双方への「共通鍵」受け渡し方法が課題

b)公開鍵暗号アルゴリズム ＝ 非対称暗号系

・対となる「秘密鍵」と「公開鍵」を使用。

「公開鍵」による暗号化データは、対となる「秘密鍵」によってのみ復号化可能。

同一の「公開鍵」では復号化不可。その逆(「公開鍵」⇔「秘密鍵」)も真。

〈長所〉・"公開鍵"をネット越しに公開可("秘密鍵"の所有がセキュリティ上肝要)

〈短所〉・暗号化/復号化処理の計算量が大

c)ハッシュ関数

・任意長の入力データに対し規則性のない固定長の"ハッシュ値"を生成。

次の特性をもつ。

＋ "ハッシュ値"生成は高速計算が可能

＋ 生成"ハッシュ値"から元の入力データの推定は困難

＋ 生成"ハッシュ値"と同一"ハッシュ値"を持つ別な入力データ内容を見つけることは困難

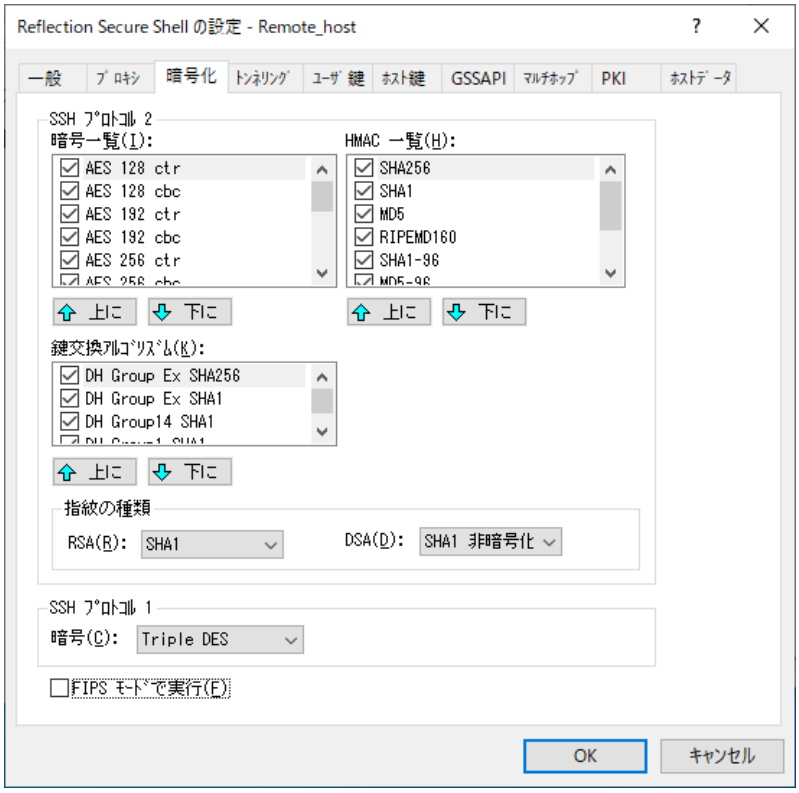
＋ 同じ生成"ハッシュ値"を持つ 2 つの入力データを見つけることは困難

《ssh への適用》：

適用先	適用暗号化技術	採用の背景
通信データの暗号化	共通鍵暗号アルゴリズム	容量大の転送ファイルをリアルタイムに処理(暗号化/復号化)する必要あり
共通鍵の共有 (接続の度に新規生成)	Diffie-Hellman 鍵交換アルゴリズム	サーバ/クライアント間で交換する演算因子情報から生成共有する共通鍵情報を第三者は知り得ないこと
サーバ認証	公開鍵暗号アルゴリズムによる署名	確認合致する「公開鍵」と対となる「秘密鍵」ファイルを所有することが認証上の決め手
ユーザ認証	公開鍵暗号アルゴリズムによる署名 等	登録「公開鍵」と対となる「秘密鍵」ファイルを所有することが認証上の決め手
データ改ざん検知 (=データ完全性保証)	HMAC[ハッシュ関数によるメッセージ認証符号]	共通鍵と送信データとハッシュ関数をもとに計算した HMAC を使いハッシュ関数の特性を利用

《対応するパラメータ》：

	″config″設定 パラメータ	[Reflection Secure Shell の設定] >[暗号化]タブ
通信データの暗号化用 共通鍵暗号アルゴリズム	Ciphers	″暗号一覧″
共通鍵共有用 Diffie-Hellman 鍵交換アルゴリズム	KexAlgorithms	″鍵交換アルゴリズム″
データ改ざん検知用 HMAC	Macs	″HMAC 一覧″



3.3 サーバ認証

クライアントは、接続開始直後のサーバからの応答が、中間なりすまし等の不正な偽装サーバからではなく、真に接続しようとした正規接続先サーバからであることを確認します。このクライアントによるサーバの確認を「サーバ認証」と呼びます。
サーバ認証は、次の 2 条件を満たすことで成立します。

《サーバ認証成立条件》：

a)提示されたサーバホスト鍵公開鍵が確認済みのものと合致（下記①②いずれかにより確認）
　①クライアント保存の"known_hosts"ファイルに登録されたホスト鍵と合致
　②連絡受けたホスト鍵"fingerprint"(Hash によりダイジェストされた出力値コード)と合致
b)提示された公開鍵の対となる秘密鍵を所有している証としての確認手続きの成功

[注記]：

a)の対象となる鍵ファイル： rsa 鍵 あるいは dsa 鍵。および その bit 長が関係
b)の所有確認手続きでは Hash 値も利用。よって、使用する Hash アルゴリズムも関係する。
（使用パラメータ交渉時に決定）

サーバ認証は、クライアントプログラムが登録"known_hosts"ファイル内容を自動的に検索し処理を進めます。よってセキュリティ上は、登録時の確認配慮が必要です。

《サーバホスト鍵の厳格な登録方法》：

・下記いずれかを実施します。
a)入手サーバホスト鍵公開鍵ファイルをインポート保存
b)接続時に表示するサーバホスト鍵"fingerprint"を目視確認の上で保存

[注記]：

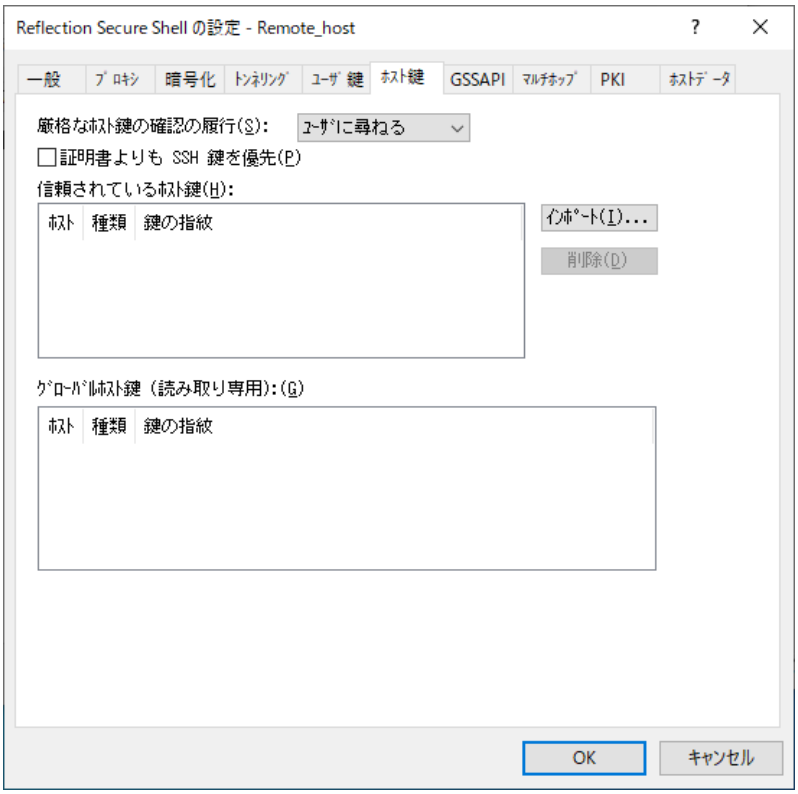
接続先サーバへは、IP アドレスか（名前解決される)ホスト名を使い、接続指定します。
接続操作や("known_hosts"ファイル内の記述を含めた)設定上の何か所かに接続先サーバの指定記述箇所が存在しますが、ある接続先サーバに対する指定は、IP アドレスかホスト名かのいずれか一方を一貫して統一使用下さい。
ネットワーク上の接続はいずれも可能であっても、内部設定上のキーワード情報として使用する場合に両者別物扱いとなり、検索条件不一致に起因して期待通りの動作とならないからです。

《サーバホスト鍵公開鍵 インポート手順》：

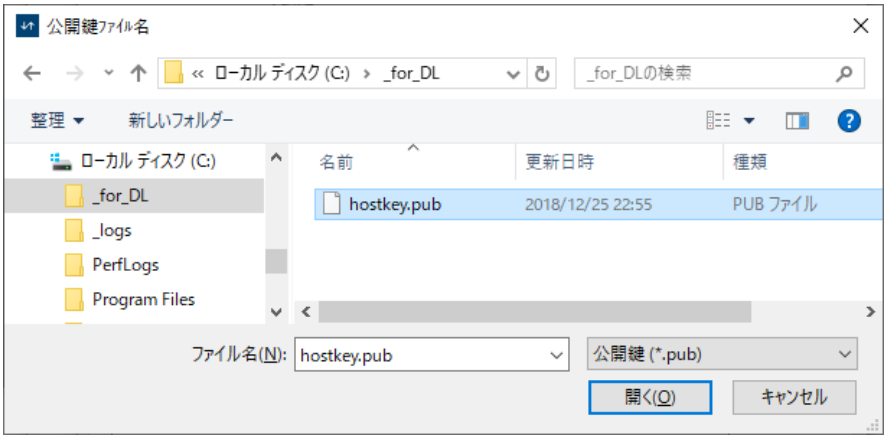
①「SSH クライアント」接続画面から[接続の設定]画面を開く。

②"ホスト名","ユーザ名"を入力し、[セキュリティ]ボタンを押下し[Reflection Secure Shell の設定]画面を開く。

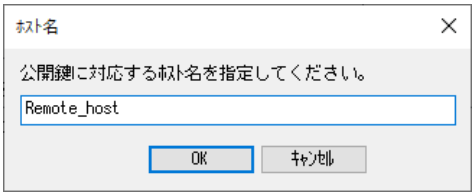
③[ホスト鍵]タブ画面を選択開き、[インポート]ボタンを押下。



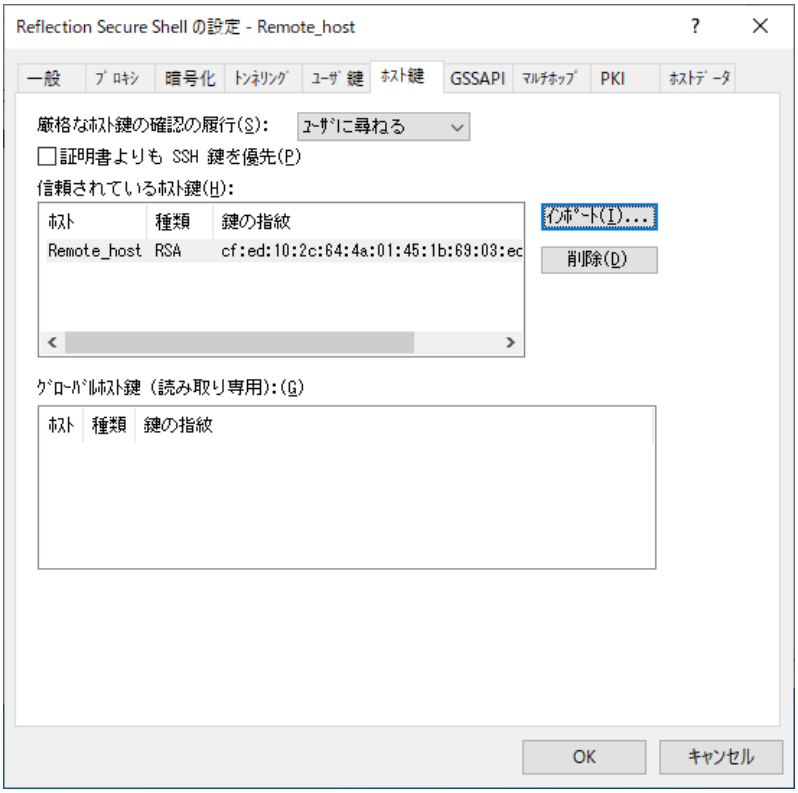
④表示[公開鍵ファイル名]画面にて、保存先フォルダ下の公開鍵ファイルを選択し、[開く]ボタン押下



⑤表示[ホスト名]画面 入力欄に接続先ホスト名を入力し、[OK]ボタン押下



⑥結果、[ホスト鍵]タブ画面の"信頼されているホスト鍵"欄に、追加鍵ファイルを表示。

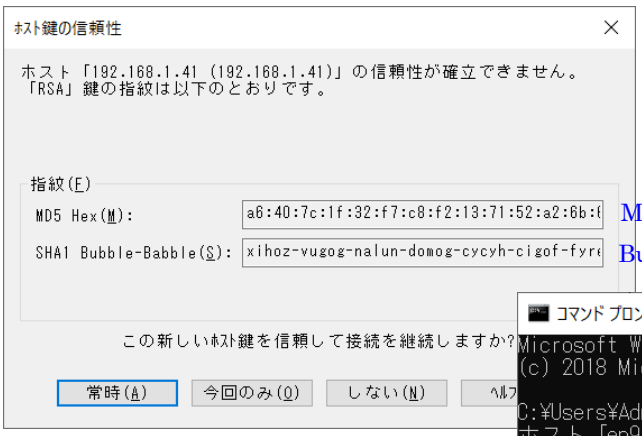


《ホスト鍵"fingerprint"の目視照合確認》 :

RSIT Windows クライアントでは、接続時に目視確認用ホスト鍵公開鍵"fingerprint"として「MD5 Hex 形式」と「Bubble-Babble 形式」で表示します。

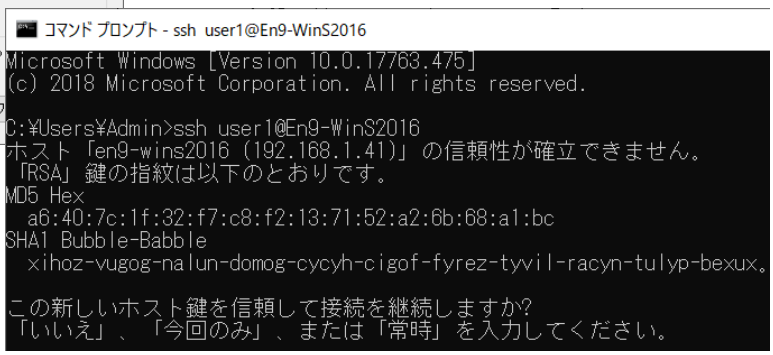
以下に、主な ssh サーバのホスト鍵"fingerprint"情報の確認方法を示します。事前に接続先 ssh サーバのサーバ管理者と連携し、その情報を入手下さい。

1) RSIT Windows クライアント 目視確認用ホスト鍵公開鍵"fingerprint"表示例



MD5 Hex 形式

Bubble-Babble 形式

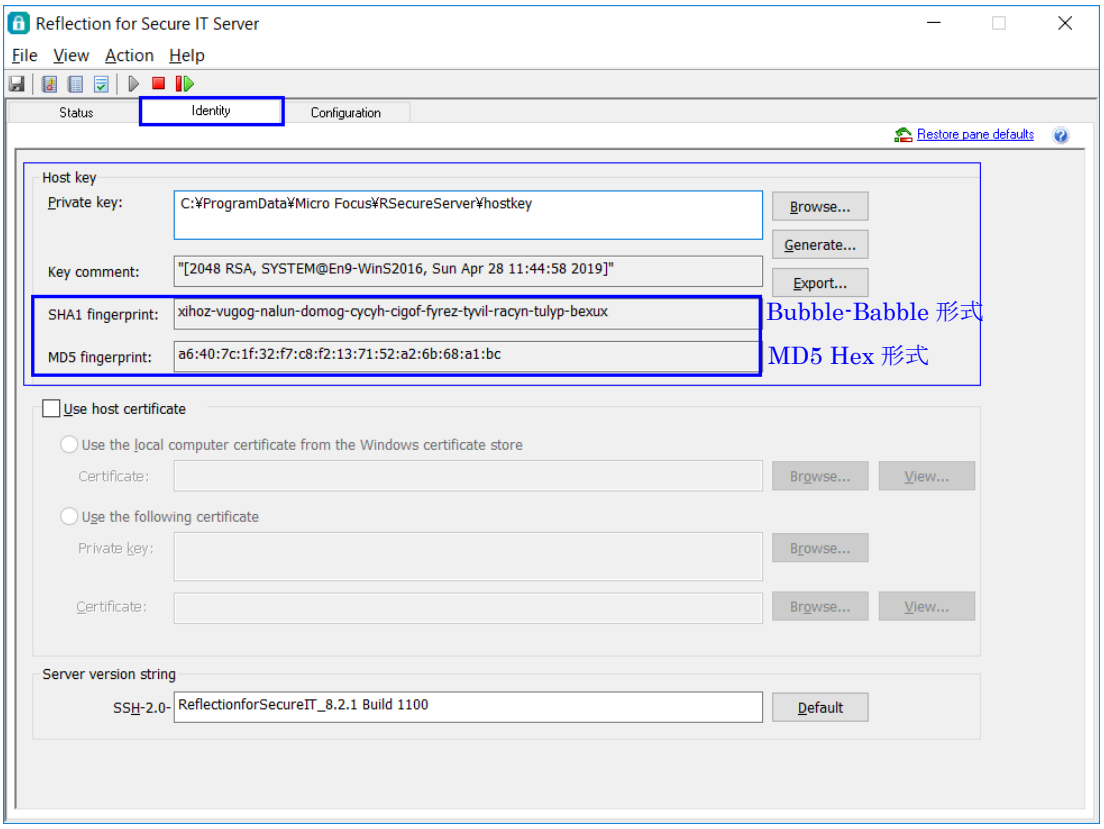


2) 主な ssh サーバ ホスト鍵"fingerprint"情報の確認例

［注記］:

対となる「秘密鍵」/「公開鍵」ファイルの"fingerprint"は同一となります。

a) RSIT Windows Server 8.2 の場合



b) RSIT UNIX Server 8.0.2 の場合

```
# ssh -V
ssh: Reflection for Secure IT 8.0.2.146 on x86_64-redhat-linux-gnu (64-bit).

# cd /etc/ssh2
# ls -l
合計 156
-rw------- 1 root root  1832  4 月 28 14:04 hostkey
-rw-r--r-- 1 root root   540  4 月 28 14:04 hostkey.pub
-r-xr-xr-x 1 root root  7308  8 月  4  2017 migrate.sh
-rw-r--r-- 1 root root 61243  8 月  4  2017 rsit_rhel6.pp
-rw-r--r-- 1 root root 61657  8 月  4  2017 rsit_rhel7.pp
-rw-r--r-- 1 root root  2851  4 月 28 14:04 ssh2_config
-rw-r--r-- 1 root root  2860  8 月  4  2017 ssh2_config.example
-rw-r--r-- 1 root root  3477  4 月 28 14:04 sshd2_config
-rw-r--r-- 1 root root  3486  8 月  4  2017 sshd2_config.example
drwxr-xr-x 2 root root    46  4 月 28 14:04 subconfig

# ssh-keygen -F hostkey
Fingerprint for key:            Bubble-Babble 形式
xenol-rivof-vydis-hasub-vyfaz-sadag-sanic-rocub-sucur-kidan-pixux
```

c) OpenSSH_7.4p1 の場合

```
# ssh -V
OpenSSH_7.4p1, OpenSSL 1.0.2k-fips  26 Jan 2017

# cd /etc/ssh
# ls -l
合計 604
-rw-r--r--. 1 root root     581843 11 月 24  2017 moduli
-rw-r--r--. 1 root root       2276 11 月 24  2017 ssh_config
-rw-r-----. 1 root ssh_keys    227  1 月 30 14:18 ssh_host_ecdsa_key
-rw-r--r--. 1 root root        162  1 月 30 14:18 ssh_host_ecdsa_key.pub
-rw-r-----. 1 root ssh_keys    387  1 月 30 14:18 ssh_host_ed25519_key
-rw-r--r--. 1 root root         82  1 月 30 14:18 ssh_host_ed25519_key.pub
-rw-r-----. 1 root ssh_keys   1675  1 月 30 14:18 ssh_host_rsa_key
-rw-r--r--. 1 root root        382  1 月 30 14:18 ssh_host_rsa_key.pub
-rw-------  1 root root       3907  5 月  9 10:37 sshd_config

# ssh-keygen -B -f ssh_host_rsa_key            Bubble-Babble 形式
2048 xefil-kanem-catec-ganid-hukat-mycib-memus-vepit-mygyr-sisov-kaxyx ssh_host_rsa_key.pub (RSA)
# ssh-keygen -l -E md5 -f ssh_host_rsa_key     MD5 Hex 形式
2048 MD5:de:24:62:41:3a:fa:5a:be:29:91:66:09:2f:9c:18:a4 ssh_host_rsa_key.pub (RSA)
```

d) OpenSSH_5.3p1 の場合

```
# ssh -V
OpenSSH_5.3p1, OpenSSL 1.0.1e-fips 11 Feb 2013

# cd /etc/ssh
# ls -l
合計 156
-rw-------  1 root root 125811  3 月 20 20:25 2019 moduli
-rw-r--r--  1 root root   2047  3 月 20 20:25 2019 ssh_config
-rw-------. 1 root root    668 12 月 14 00:05 2018 ssh_host_dsa_key
-rw-r--r--. 1 root root    590 12 月 14 00:05 2018 ssh_host_dsa_key.pub
-rw-------. 1 root root    963 12 月 14 00:05 2018 ssh_host_key
-rw-r--r--. 1 root root    627 12 月 14 00:05 2018 ssh_host_key.pub
-rw-------. 1 root root   1675 12 月 14 00:05 2018 ssh_host_rsa_key
-rw-r--r--. 1 root root    382 12 月 14 00:05 2018 ssh_host_rsa_key.pub
-rw-------  1 root root   3879  3 月 20 20:25 2019 sshd_config

# ssh-keygen -B -f ssh_host_rsa_key            Bubble-Babble 形式
2048 xohas-femub-kuber-tasap-lifop-dulac-ryrah-nezuh-lyfuz-peviv-kexex ssh_host_rsa_key.pub (RSA)
# ssh-keygen -l -f ssh_host_rsa_key            MD5 Hex 形式
2048 7e:7d:d3:6c:41:c2:e6:88:31:50:69:53:2b:11:d7:ba ssh_host_rsa_key.pub (RSA)
```

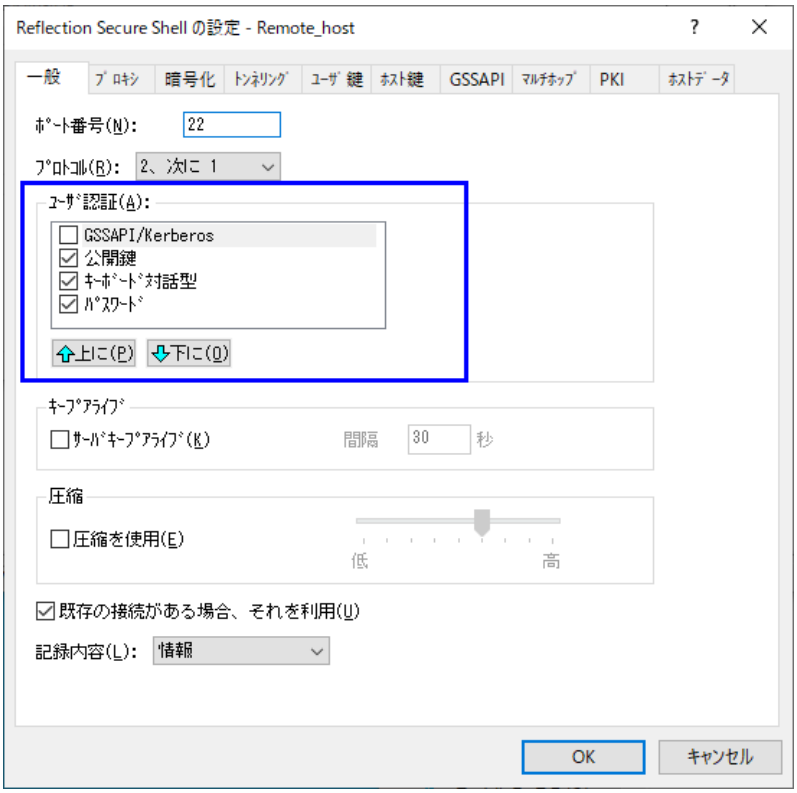
3.4 ユーザ認証

接続先 ssh サーバが規定する「ユーザ認証」手段に従い、対応した認証にパスして初めてログイン/ログオンが許可されます。

クライアント側において指定する「ユーザ認証」項目は、あくまでもクライアント側が使用可能な候補を提示することを意味します。

《RSIT Windows クライアントの指定と ssh サーバ認証方式の対応》：

RSIT Windows クライアント設定 〔上段〕：「GUI 設定画面表示」 〔下段〕："config 設定パラメータ"	対応する ssh サーバ側認証方式 および その解説
「パスワード」 "PasswordAuthentication"	『パスワード認証』： ・サーバからの入力要求に対し該当ユーザのパスワードを入力。
「キーボード対話型」 "KbdInteractiveAuthentication"	次のサーバ側認証方式に対応： 『PAM 認証』,『RSA SecurID トークン』,『RADIUS 認証』 ・サーバからの入力要求に対しキー入力情報を応答する認証方式。 (将来拡張性を備え、サーバ側での新方式採用変更時も、クライアント側設定変更は不要となる長所を持つ。)
「公開鍵」 "PubkeyAuthentication"	『公開鍵認証』： ・クライアント側：鍵ペアを生成し、使用秘密鍵を指定。 ・サーバ側：ssh サーバ登録仕様に従いクライアント公開鍵を登録。 『証明書認証』：(証明書署名が公開鍵アルゴリズムに基づく) ・クライアント側：発行された利用者証明書を所定格納場所に保存。 ・サーバ側：外部認証局との連携環境を構築。(例えば、RSIT サーバ環境に"Reflection PKI Service Manager"を追加導入する)
「GSSAPI/Kerberos」 "GssapiAuthentication"	『Kerberos 認証』： ・汎用的な GSS-API(Generic Security Services)を介して対応。



3.5 公開鍵認証 (ユーザ認証)

公開鍵認証は、公開鍵暗号アルゴリズムの特性を活かした認証方式で、ssh サーバ側に登録された公開鍵の対となる秘密鍵を所有し利用出来るクライアントユーザのみがログインに成功します。
また、クライアント側に保存する秘密鍵保存用パスフレーズを空白とすることで、バッチ処理自動実行時に人手操作を介することなくプログラム的な自動処理が可能な、便利で信頼性の高い認証方式です。
その準備として、クライアント側では、鍵ペアを生成保存し、使用秘密鍵を指定します。サーバ側では、ssh サーバの登録仕様に従いログインユーザ配下へクライアント生成の公開鍵を登録します。

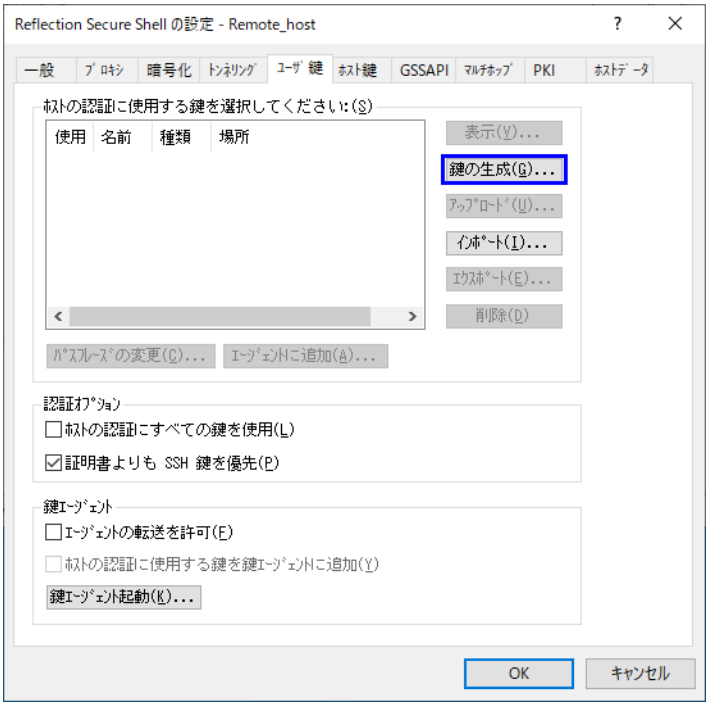
《鍵ペアファイル名称》：

- 「秘密鍵」ファイル = 拡張子なし（"xxxxx"）　＊ ファイル名称自体は任意
- 「公開鍵」ファイル = ".pub"拡張子（"xxxxx.pub"）

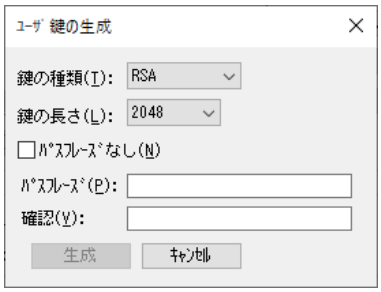
公開鍵認証用ユーザ鍵の管理について、以下記します。
GUI 画面上の操作 あるいは"ssh-keygen"コマンドにより、鍵ペアを新規生成したり、外部生成秘密鍵をインポート保存することが可能です。
鍵の保存先は、既定では"C:¥Users¥(ユーザ名)¥Documents¥Micro Focus¥Reflection¥.ssh"フォルダ下となり、変更も可能です。

《鍵ペアの生成 - GUI 画面上の操作手順》：

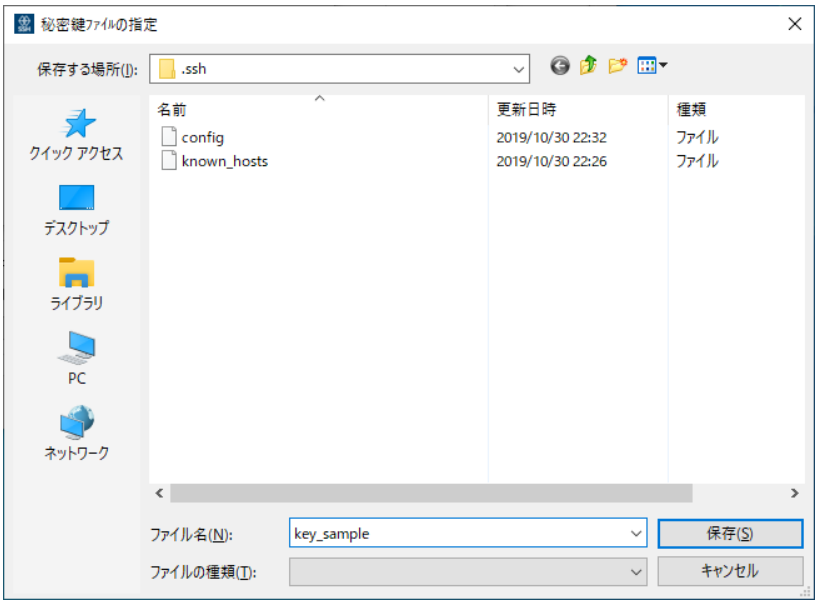
①[Reflection Secure Shell の設定]>[ユーザ鍵]タブ画面において、[鍵の生成]ボタンを押下



②表示[ユーザ鍵の生成]画面において、鍵の種類と長さ,(秘密鍵保存用の)パスフレーズ情報を指定し、[生成]ボタンを押下
自動実行処理用には「パスフレーズなし」を選択

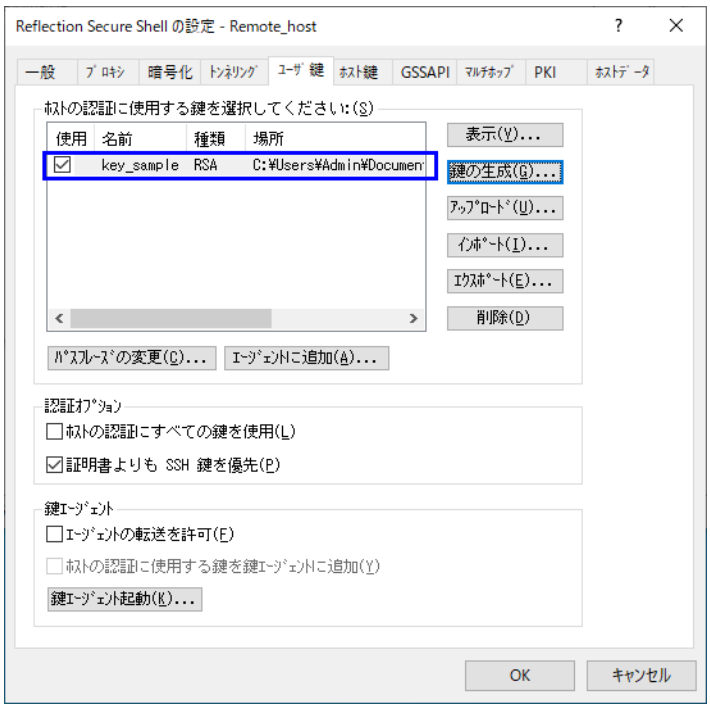


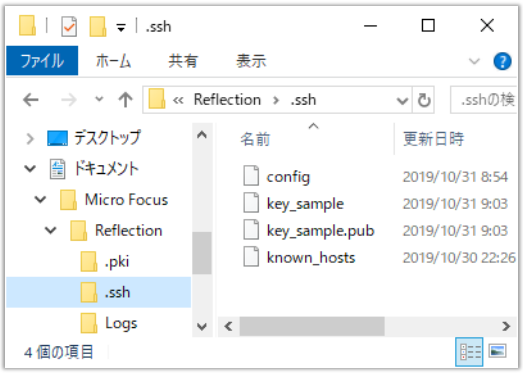
③表示[秘密鍵ファイルの指定]画面において、
保存場所とファイル名を指定し、[保存] ボタン押下



④成功メッセージを確認し[閉じる]

⑤上記操作結果、[ユーザ鍵]タブ画面の欄内に、生成鍵(秘密鍵)の"名前","種類","場所"が追加される。
合わせて"config"ファイルへ設定が自動反映されます。





<"config"内容 例>

```
Host Remote_host
    IdentityFile "C:¥Users¥Admin¥Documents¥Micro Focus¥Reflection¥.ssh¥key_sample"
#EndHost
```

《鍵ペアの生成 - "ssh-keygen"コマンド操作例》：

コマンドプロンプト画面を開き、下記操作をします。

①"ssh-keygen"コマンドを入力

[注記]：

システム環境変数"Path"にプログラムパスは登録済みですので、任意場所から"ssh-keygen"指定のみでコマンドは機能します。

②表示に従いパスフレーズを入力

自動実行処理用には「パスフレーズなし」とするため空白のまま Enter

③"config"ファイル上への生成秘密鍵の反映

・"config"ファイルへは、手操作にて"IdentityFile"指定を編集するか、下記操作を実施します。

・"C:¥Users¥(ユーザ名)¥Documents¥Micro Focus¥Reflection¥.ssh"フォルダ下に生成した場合、次回[Reflection Secure Shell の設定]画面を開いた際に、[ユーザ鍵]タブ画面の欄内に読み込まれます。(但しそのままでは"使用"チェックマークは付きませんので)人手操作にて"使用"チェックマークを付け、[Reflection Secure Shell の設定]画面内容を終了保存することで、"config"ファイルへ自動的に反映されます。

《秘密鍵のインポート - GUI 画面上の操作手順》：

既存サーバ/PC を更改する時や複製時に、既存公開鍵認証環境を流用可能です。

(= ssh サーバに登録した公開鍵はそのままで、クライアント側の既存秘密鍵をコピー流用可能です。)

<手順 1>

①流用秘密鍵ファイルを既定"C:¥Users¥(ユーザ名)¥Documents¥Micro Focus¥Reflection¥.ssh"フォルダ下に保存。

②("ssh-keygen"コマンド生成時の説明と同様に)[Reflection Secure Shell の設定]>[ユーザ鍵]タブ画面を開き、自動的に読み込まれた該当ファイルの"使用"欄にチェックマークを付け終了保存。

(以上で、"config"ファイルへも設定内容が自動的に反映されます。)

<手順 2>

①流用秘密鍵ファイルを任意のフォルダ下に保存。

②[Reflection Secure Shell の設定]>[ユーザ鍵]タブ画面を開き、[インポート]ボタンを押下。

③表示[秘密鍵ファイルの選択]画面より、保存流用秘密鍵ファイルを選択し[開く]

⇒結果、"C:¥Users¥(ユーザ名)¥Documents¥Micro Focus¥Reflection¥.ssh"フォルダ下にコピー保存され、鍵ファイル表示欄に"使用"欄チェックマーク付きで追加される。

④[Reflection Secure Shell の設定]画面を終了保存時に"config"ファイルへ設定内容が反映されます。

《公開鍵認証用 秘密鍵ファイルの指定方法》：

・下記いずれかの指定方法があります。

a)"config"にて 指定パラメータ"IdentityFile"を使い指定

b)コマンド発行時に "-i"オプションにて指定

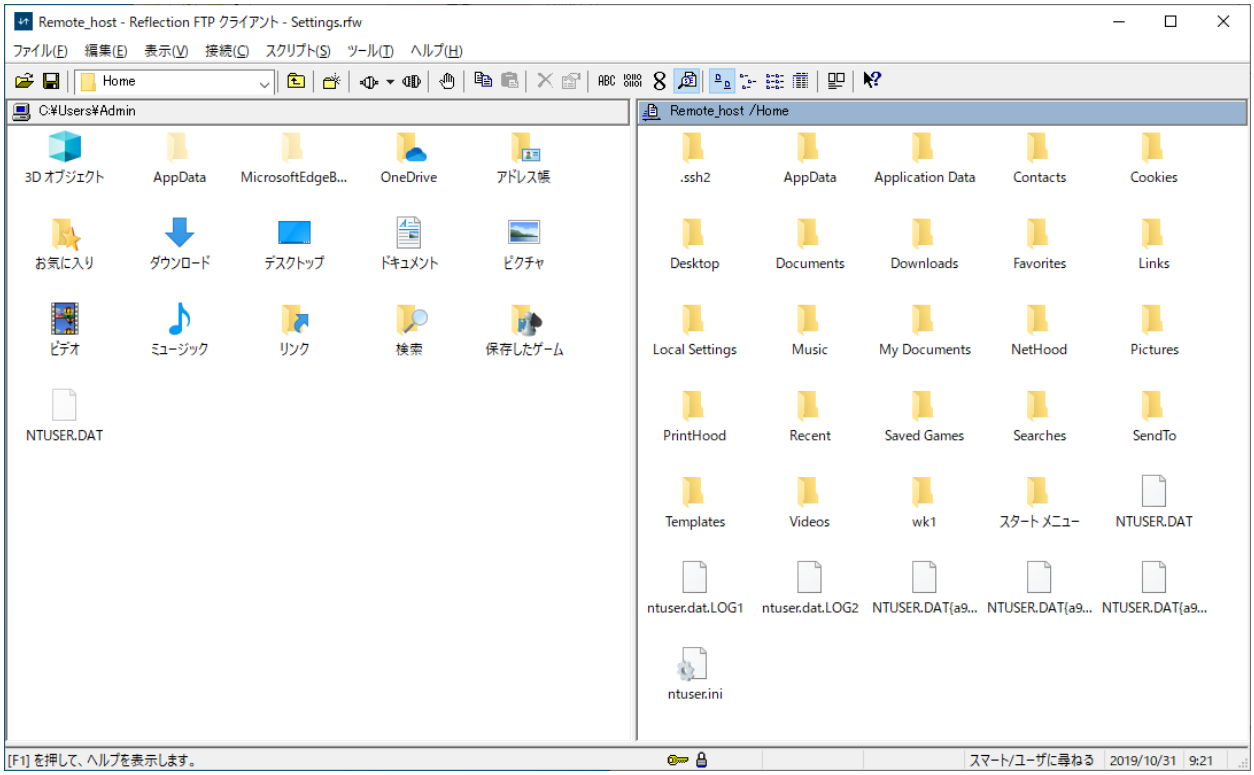
3.6 ファイル転送

RSIT Windows クライアントでは、ファイル転送用に次のユーザインタフェースを用意しています。
利用者から見たユーザインタフェースこそ違いますが、内部処理上はクライアントプログラムが sftp プロトコルに準拠した形で内部展開し、sftp サーバとの間で制御パケットと転送データ情報のやり取りをしながら処理を遂行しています。

1) GUI 操作画面

a) [Reflection FTP クライアント]:

～ インタラクティブな人手操作を前提としたエクスプローラー ライクな GUI 画面



2) コマンド指示　　(*) 詳細は「5.2 コマンド」を参照

a) sftp コマンド:

～ FTP コマンドの置き換えを意識した類似コマンド書式/オプションを用意

b) scp コマンド:

～ rcp コマンドの置き換えを意識した類似コマンド書式を用意

3.7 トンネリング (TCP ポート転送)

ssh では誕生当初からその利用価値を活かすために、平文のまま通信する TCP 上のアプリケーション通信や FTP ファイル転送、X11 通信を、確立した ssh 接続のコネクションを通じて暗号化通信するトンネリング(TCP ポート転送)機能を備えています。

ここでは、設定上の理解の一助となるよう、そのシステム構成イメージと設定の対応を示します。

《システム構成イメージ図》：

1) 従来のアプリケーション間通信

～ Client アプリは、アプリケーションサーバ[=IP アドレス(A)]上の Server アプリ[=ポート番号(a)]へ接続し通信します。

2) ローカルポート転送

〈ケース a〉：同一アプリケーションサーバ内に SSH Server が存在

〈ケース b〉：アプリケーションサーバと SSH サーバが別

～① RSIT Windows Client （＋設定）からアプリケーションサーバ[=IP アドレス(A)] あるいは SSH サーバ[=IP アドレス(B)]上の SSH Server[=ポート番号 22]へ ssh 接続

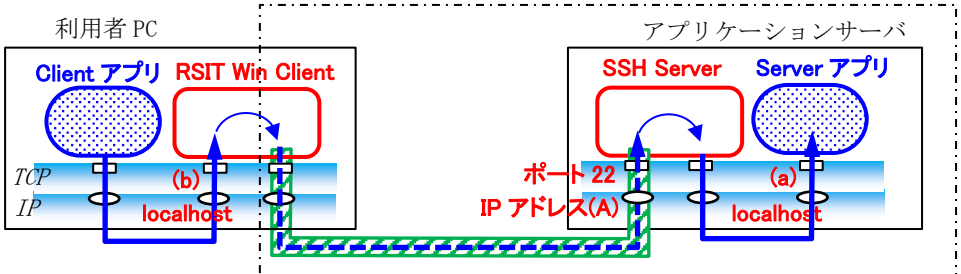
② Client アプリは、localhost 上のポート番号(b)[= RSIT Windows Client 設定値]へ接続

③ ポート番号(b)で待ち受けしている RSIT Windows Client は、ポートからの受信データを接続確立した ssh 接続内のあるチャネルへ転送します。

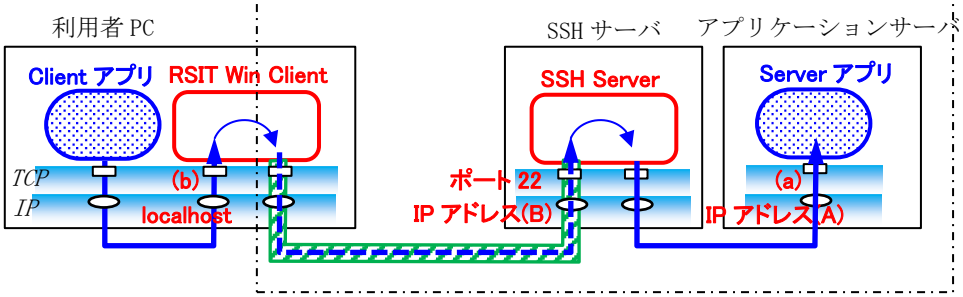
④ SSH Server は、接続確立した ssh 接続から受信したデータを、SSH Server から見た時の指定転送先 IP アドレス/ポート番号へポート転送します。

	転送先 IP アドレス	転送先ポート番号
〈ケース a〉	localhost	ポート番号(a)
〈ケース b〉	IP アドレス(A)	ポート番号(a)

〈ケース a〉



〈ケース b〉

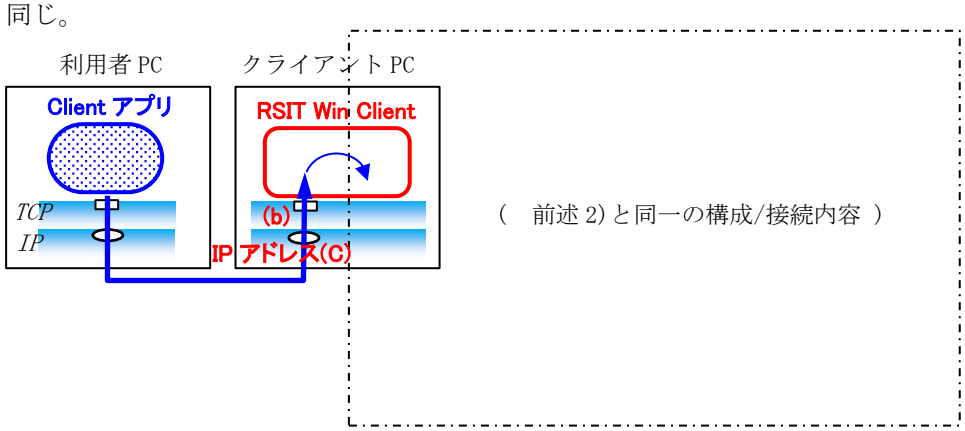


3) ローカルポート転送（ゲートウェイポート利用時）

～① クライアント PC 上の RSIT Windows Client(＋設定)からアプリケーションサーバ[=IP アドレス(A)]
あるいは SSH サーバ[=IP アドレス(B)]上の SSH Server[=ポート番号 22]へ ssh 接続

② Client アプリは、クライアント PC[=IP アドレス(C)]上のポート番号(b)[= RSIT Windows Client 設定値]へ接続

③④ は、2)と同じ。

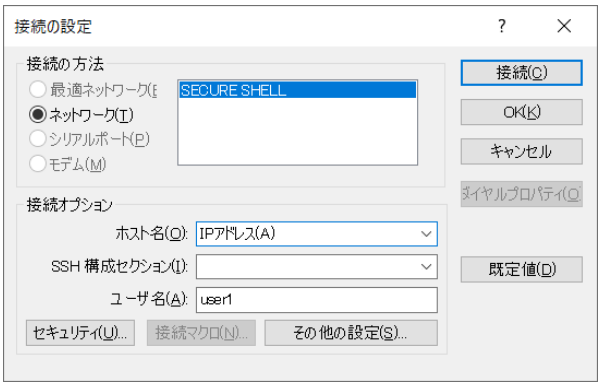


《対応する設定内容》 ：

2) ローカルポート転送

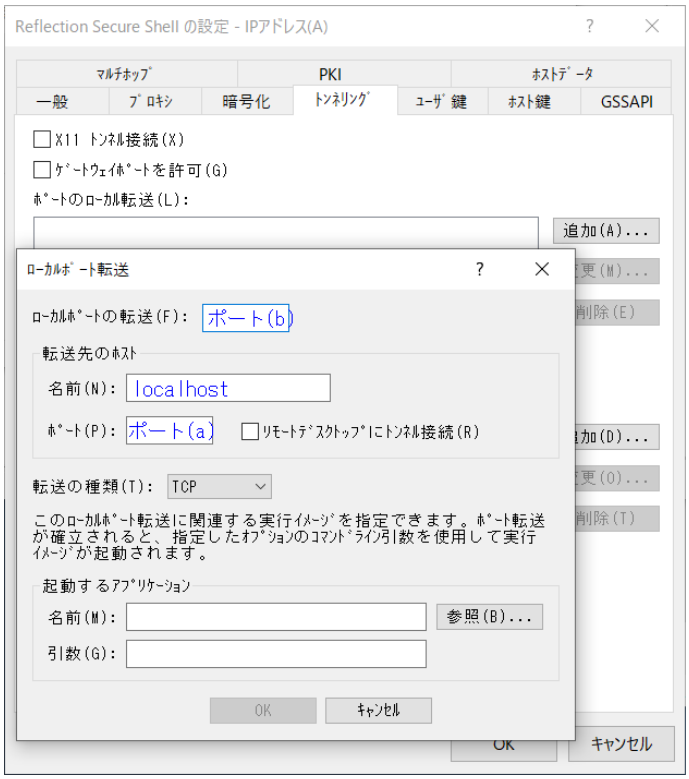
〈ケース a〉

①[接続の設定]画面：



「ホスト名」＝"IP アドレス(A)"
(= アプリケーションサーバ IP アドレス)

②[ローカルポート転送] 画面：



「ローカルポートの転送」="ポート(b)"
(= RSIT Windows Client が待ち受けるポート番号)

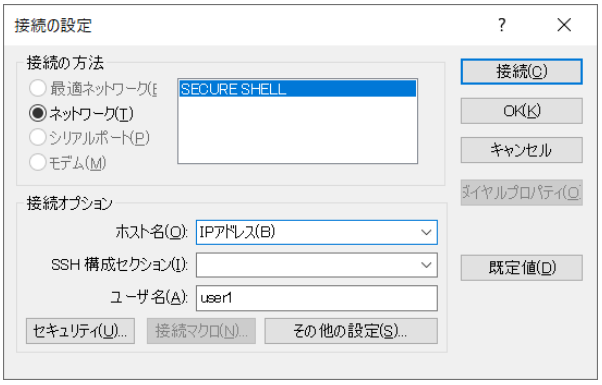
#"転送先のホスト"

「名前」="localhost"
(= SSH Server から見た 転送先 Server アプリの IP アドレス)

「ポート」="ポート(a)"
(= 転送先 Server アプリのポート番号)

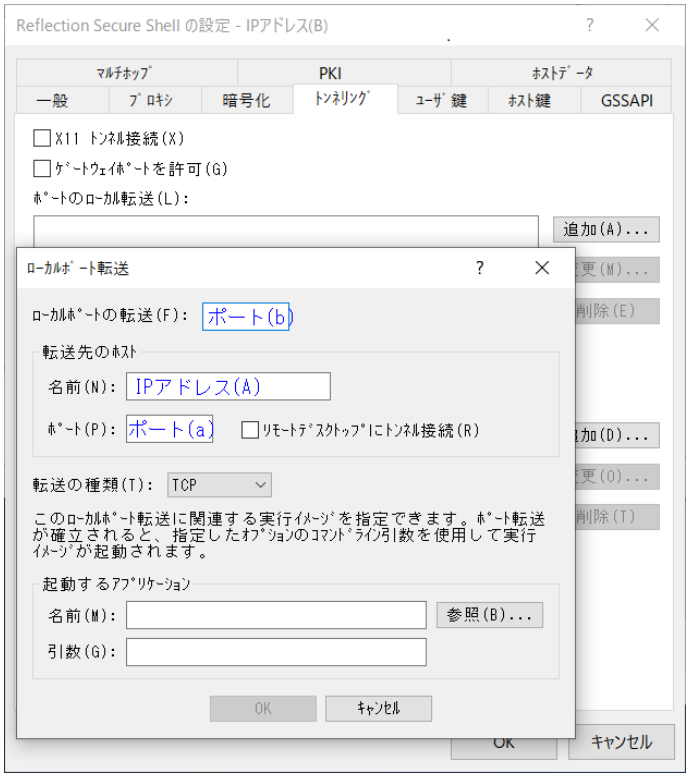
〈ケース b〉

①[接続の設定]画面：



「ホスト名」= "IP アドレス(B)"
(= SSH サーバ IP アドレス)

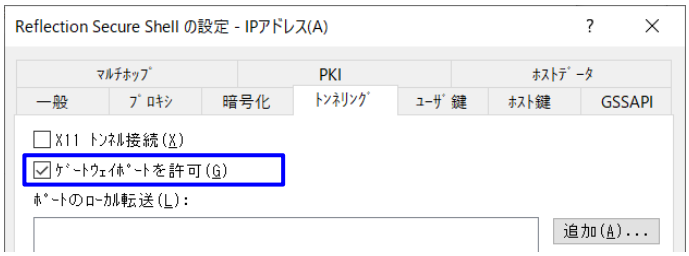
②[ローカルポート転送] 画面：



「ローカルポートの転送」="ポート(b)"
(= 利用者 PC 上の RSIT Windows クライアントが
待ち受けるポート番号)
#"転送先のホスト"
「名前」="IP アドレス(A)"
(= SSH Server から見た 転送先 Server アプリの IP アドレス)
「ポート」="ポート(a)"
(= 転送先 Server アプリのポート番号)

3) ローカルポート転送（ゲートウェイポート利用時）

・[Reflection Secure Shell の設定] > [トンネリング]タブ画面：



「ゲートウェイポートを許可」にチェックマーク付与。

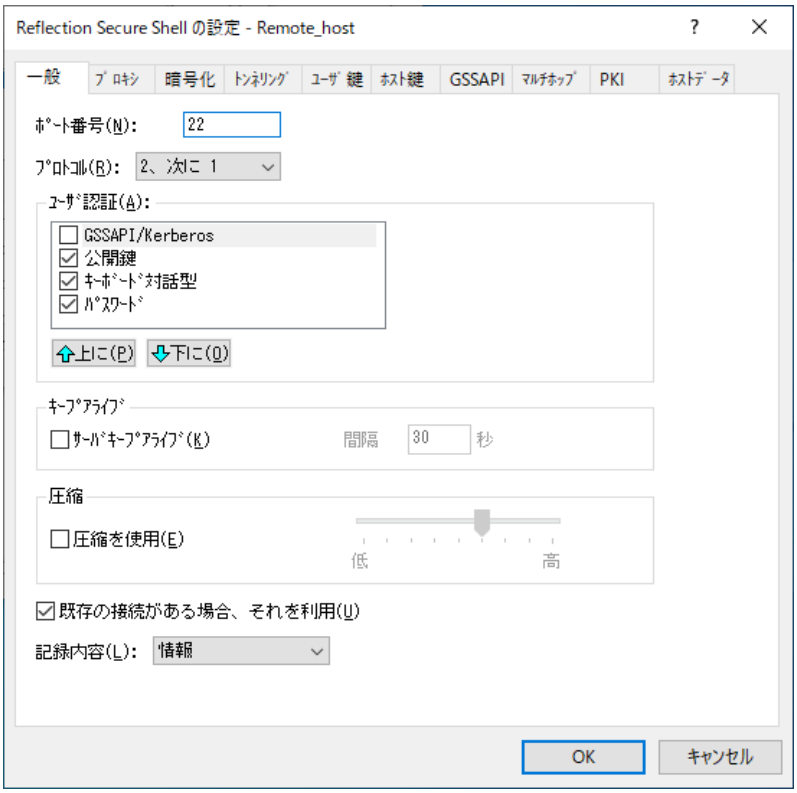
後は上記 2) ローカルポート転送 〈ケース a〉,〈ケース b〉と同じ。

5. 仕様一覧

5.1 設定画面

5.1.1 [Reflection Secure Shell の設定]画面

1) [一般]タブ



ポート番号

サーバに接続するポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。

プロトコル

ホストへの接続の確立時に Reflection が使用する Secure Shell プロトコルのバージョンを指定します。この設定で最も安全な値は [2 のみ] です。

ユーザ認証

いずれかの認証方式の横にあるボックスをクリックして、その方式をオフにするか、オンにします。少なくとも 1 つの認証方式を選択する必要があります。プロトコル 2 接続の場合は、矢印キーを使って優先順位を指定できます。Reflection は、それぞれの方式を先頭から順番に試行します。

サーバキープアライブ

[サーバキープアライブ] チェックボックスをオンにすると、Reflection は、指定された間隔で、安全なトンネルを介して NOOP メッセージをサーバへ送信します。この設定は、サーバとの接続を維持するために使用します。[間隔] を使用して、サーバがキープアライブメッセージを送信する頻度を指定します。この設定が有効になっていないと、サーバが停止するかネットワーク接続が失われた場合に Secure Shell 接続が終了されません。この設定は、TCP セッションのみを転送する接続がサーバによって時間切れになるのを防ぐためにも使用できます。サーバは、SSH トラフィックが存在しないことの検出を理由にこれらの接続を時間切れにする場合があるからです。

Secure Shell の［サーバキープアライブ］の設定は、ファイアウォールによってすべての TCP/IP 接続が時間切れにならないようにする、Windows のレジストリで設定できる TCP キープアライブの設定とは関係ありません。TCP/IP キープアライブの動作を変更するには、Windows レジストリを編集する必要があります。

圧縮を使用

［圧縮を使用］チェックボックスをオンにすると、クライアントはすべてのデータの圧縮を要求します。圧縮はモデム回線やほかの低速接続に適していますが、高速のネットワークでは応答速度の低下を招くだけです。圧縮レベルの設定はプロトコルバージョン 1 でのみ可能で、プロトコルバージョン 2 接続では適用されません。

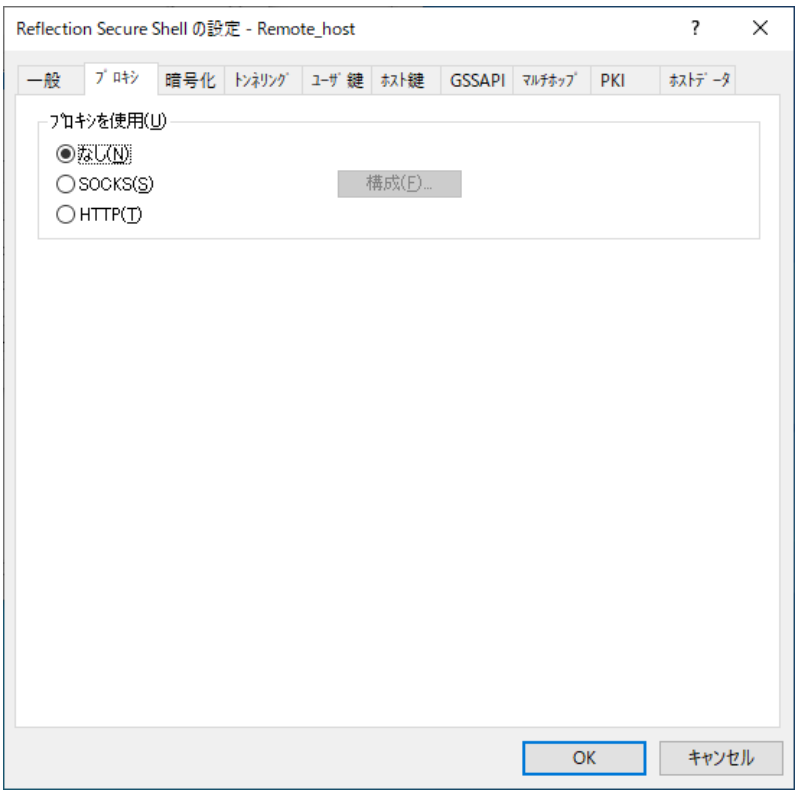
既存の接続がある場合、それを利用

既定で、同一ホストへの複数のセッションでは元の Secure Shell 接続が再利用されます。このため、再認証が不要となります。［既存の接続がある場合、それを再利用する］チェックボックスをオフにすると、Reflection はセッションごとに新しい接続を確立します。つまり、新しい接続ごとに認証プロセスが繰り返されます。

記録内容

Secure Shell のログファイルに書き込む情報レベルを決定します。

2) ［プロキシ］タブ



#プロキシを使用

なし

プロキシは設定されていません。（デフォルト値）

SOCKS

［SOCKS］を選択すると、SOCKS プロキシを介した Secure Shell 接続を構成できます。

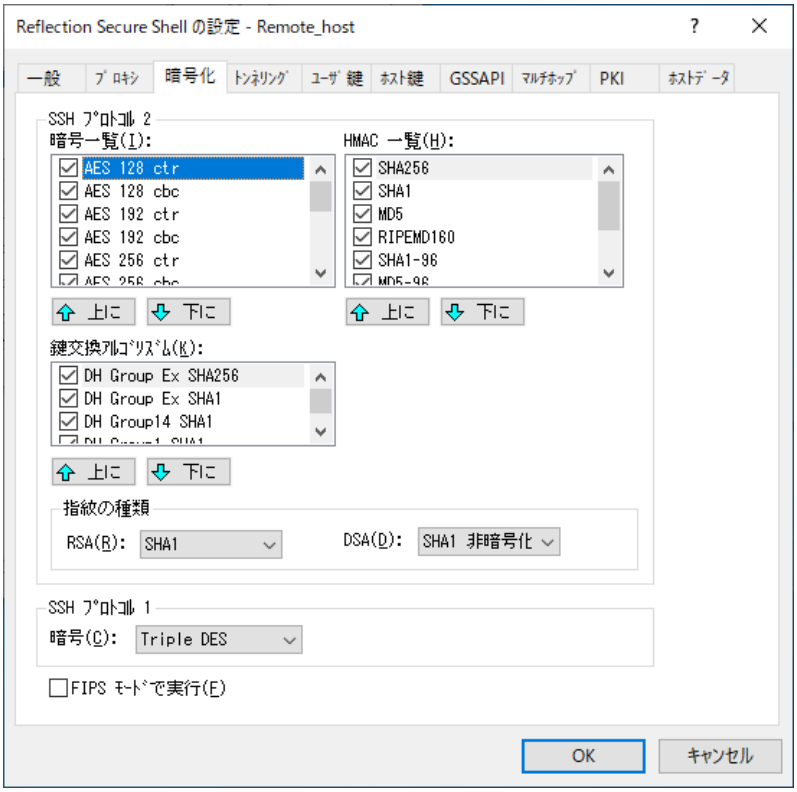
HTTP

［HTTP］を選択すると、HTTP プロキシを介した Secure Shell 接続を構成できます。

［構成］ボタン

プロキシサーバの設定を構成します。

3) ［暗号化］タブ



SSH プロトコル 2

暗号一覧

この一覧は、現在のホストとのプロトコル 2 接続に使用可能にする暗号を選択するために使用します。複数の暗号を選択すると、Secure Shell クライアントは、指定した順番で先頭から暗号の使用を試みます。順序を変更するには、一覧から Cipher を選択して上向きまたは下向きの矢印をクリックします。特定のセッション用に使用された暗号は、この一覧の最初のアイテムであり、サーバもこの一覧に対応しています。

HMAC 一覧

使用可能にする HMAC（ハッシュメッセージ認証コード）方式を指定します。このハッシュは、サーバと交換されるすべてのデータパケットの整合性を検証するために使用されます。複数の HMAC を選択すると、Secure Shell クライアントは、指定した順番で先頭の HMAC からサーバと交渉します。順序を変更するには、一覧から HMAC を選択して上向きまたは下向きの矢印をクリックします。

鍵交換アルゴリズム

クライアントが対応する鍵交換アルゴリズムと優先順位を指定します。

場合によっては、DH Group14 SHA1 を他の値より前に配置するために、鍵交換アルゴリズムの順番を変更する必要があります。hmac-sha512 MAC を使用する場合、または鍵交換中に次のエラーが表示される場合は、この操作が必要です。"fatal: dh_gen_key: group too small: 1024 (2*need 1024)"

その他に 2 つの暗号化アルゴリズム (gss-group1-sha1-*) に対応していますが、これらは使用可能な鍵交換アルゴリズムの一覧には表示されません。これらの 2 つのアルゴリズムは、[全般] タブ ([ユーザ認証] の下) で GSSAPI/Kerberos を使用可能にし、[GSSAPI] タブで [Reflection Kerberos] をオンにすると、クライアントによって自動的に提案されます。

指紋の種類

秘密鍵の所有を提供する過程でクライアントが使用するハッシュアルゴリズムを指定します。公開鍵ユーザ認証時にこのハッシュが使用されます。RSA 鍵で使用されるハッシュを指定するには RSA を使用し、DSA 鍵で使用されるハッシュを指定するには DSA を使用します。

 # SSH プロトコル 1

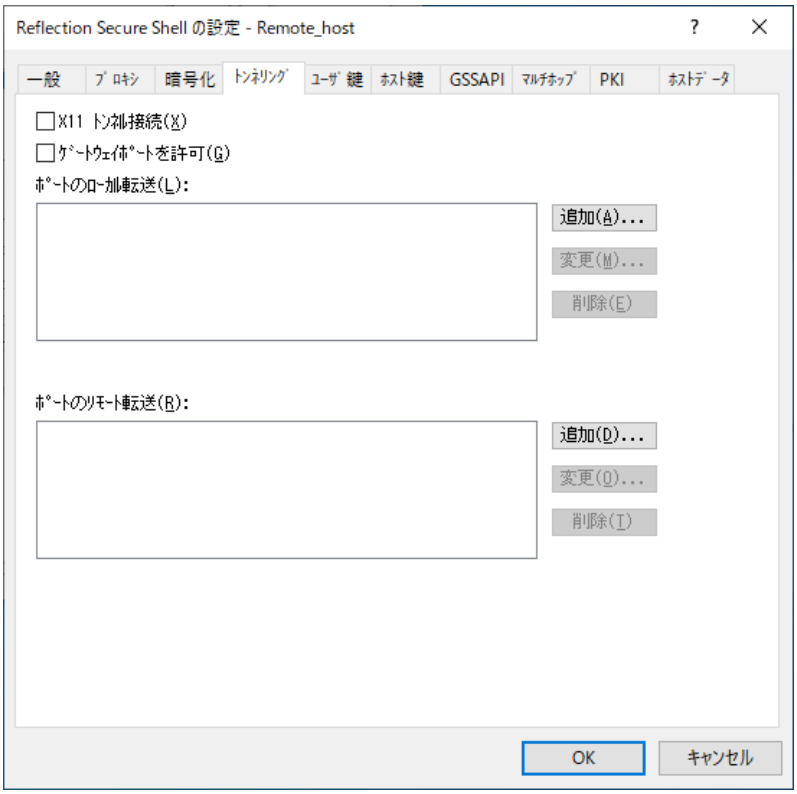
暗号

この設定は、現在のホストとのプロトコル 1 接続に使用したい暗号を選択するために使用します。既定値は [Triple DES] で、これが推奨されるオプションです。

FIPS モードで実行

[FIPS モードで実行] チェックボックスをオンにすると、Reflection は、その接続に対してアメリカ合衆国政府の連邦情報処理標準 (FIPS) 140-2 を施行します。[FIPS モードで実行] チェックボックスをオンにした場合、この標準に適合していない [暗号化] タブのオプションは使用不可になります。

4) [トンネリング]タブ



X11 トンネル接続

X11 リモートポートから送信されるすべてのデータが安全なトンネルを介して自動的に正しいローカルポートへ転送されるように指定します。

ゲートウェイポートを許可

ゲートウェイポートを有効にします。リモートホストは、ローカル転送ポートへの接続を許可されます。既定では、Reflection Secure Shell はローカルポート転送をループバックアドレスにバインドします（ローカルのホストを使用する場合と同等）。これによって、ほかのリモートホストが、転送ポートに接続できないようにしています。[ゲートウェイポートを許可] を使用すると、Reflection Secure Shell がローカルポート転送をローカルの Ethernet アドレス（IP アドレス、URL、DNS 名など）にバインドし 、転送されたポートにリモートホストが接続できるように指定できます。

この設定を有効にする場合は注意を必要とします。この設定によって、リモートホストで、認証なしにシステムで転送されたポートを使用できるようになるため、ネットワークと接続の安全性が低下します。

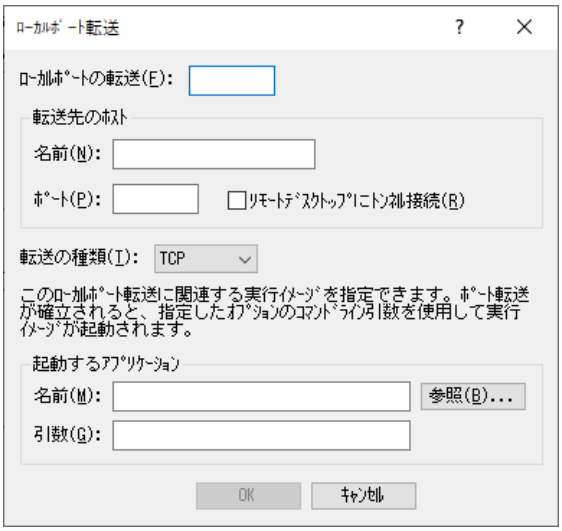
ポートのローカル転送

構成済みのローカルポート転送を表示します。[追加] をクリックすると、[ローカルポート転送] ダイアログボックスが開きます。

ポートのリモート転送

構成済みのリモートポート転送を表示します。[追加] をクリックすると、[リモートポート転送] ダイアログボックスが開きます。

4-1)［ローカルポート転送］ダイアログボックス



ローカルポートの転送

PC の使用可能なポートを指定します。このポートに送信されたデータは、SSH トンネルを介して転送されます。

#転送先のホスト

名前

データの送信先ホストコンピュータを指定します(localhost を指定して、すでに Secure Shell 接続を確立した同じリモートホストの異なるポートにデータを転送できます)。

ポート

データの送信先ホストコンピュータを指定します([リモートデスクトップにトンネル接続] を選択した場合は、正しいリモートポートが自動的に設定され、このボックスは選択できなくなります)。

リモートデスクトップにトンネル接続

Windows リモートデスクトップセッションにトンネル接続する場合は、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するとその他のオプションが選択できなくなり、自動的にセッション転送設定が正しく設定されます。

転送の種類

［TCP］と［FTP］の 2 つのオプションがあります。FTP クライアントとサーバ間で通信を転送する場合を除いて、TCP を使用してください。

設定可能なオプション

#起動するアプリケーション

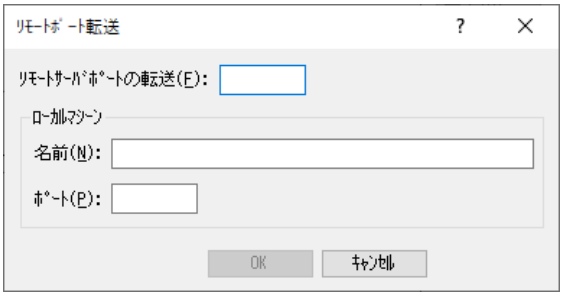
名前

Secure Shell 接続確立後に自動的に起動するアプリケーション（メールクライアント、FTP クライアント、または Web ブラウザ）の名前を入力します。安全なトンネルを使用するには、［ローカルポートの転送］に設定したポートに接続するようにアプリケーションを設定する必要があります。一部のアプリケーションでは、コマンドライン引数を使用してこのように設定できます。引数は［引数］テキストボックスに指定します。

引数

指定したアプリケーションが起動した時に使用するオプションのコマンドライン引数を指定します。

4-2) ［リモートポート転送］ダイアログボックス



リモートサーバポートの転送

ホストコンピュータのポートを指定します。このポートから送信されたデータは、SSH トンネルを介して PC に転送されます。

#ローカルマシーン

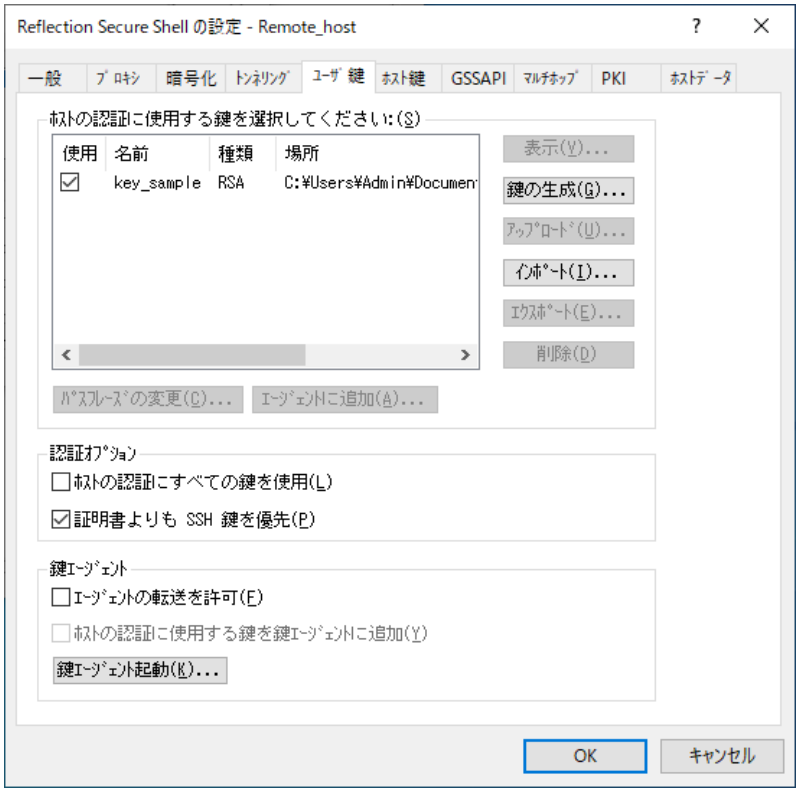
名前

データの送信先のローカルコンピュータを指定します。

ポート

データの送信先のリモートコンピュータを指定します。

5) [ユーザ鍵]タブ



#ホストの認証に使用する鍵を選択してください：

[表示]ボタン

選択した鍵または証明書の内容を表示します。

[鍵の生成]ボタン

[ユーザ鍵の生成] ダイアログボックスを開きます。このダイアログボックスは、ユーザ鍵認証用の公開鍵と秘密鍵のペアを構成するために使用できます。

[アップロード]ボタン

現在指定されているホストへ公開鍵をアップロードします。ユーティリティは自動的にホストの種類を検出し、既定でこのホストにとって適切な設定を使用して鍵をアップロードします。ホストへの安全な接続が確立されると、ダイアログボックスが開き、この鍵をアップロードするホスト上の場所に関する情報が表示されます。通常は、この設定を変更する必要はありません。

ユーティリティによって決定されるホストまたはキータイプが間違っている場合は、Secure Shell 構成ファイルで ServerKeyFormat と ServerStyle キーワードを設定することで、キーのアップロードのためのホスト固有の値を構成できます。

公開鍵は、安全な SFTP プロトコルを使用して転送されます。公開鍵をアップロードするには、パスワード認証（または別の認証方式）を使用できる必要があります。公開鍵のアップロードに成功すると、その他の認証方式を無効にできます。

[インポート]ボタン

使用可能な鍵の一覧に秘密鍵を追加します。この機能を使用すると、他のアプリケーションを使用して作成した鍵に簡単にアクセスできます。鍵をインポートすると、Secure Shell フォルダにコピーされます。

[エクスポート]ボタン

公開鍵をエクスポートするか、公開鍵と秘密鍵のペアをエクスポートします。

[削除]ボタン

選択したキーを削除します。

[パスフレーズの変更]ボタン

選択した鍵の保護に使用するパスフレーズを変更します。

[エージェントに追加]ボタン

選択した鍵を 鍵エージェントに追加します。鍵エージェントを起動したことがない場合や、鍵エージェントがロックされている場合は、鍵エージェントのパスフレーズの入力を要求されます。また、鍵をエージェントに追加できるようにする場合は、秘密鍵のパスフレーズの入力を要求されます。

#認証オプション

ホストの認証にすべての鍵を使用

このオプションをオンにすると、クライアントは ([使用] チェックボックスがオンになっているかどうかに関わらず) 表示されているすべての鍵を使用して認証を行います。

証明書よりも SSH 鍵を優先

この設定によって、公開鍵の認証時にクライアントがサーバに提示する証明書署名の種類の順序が決定されます。この設定がオン (既定) の場合、クライアントは最初に標準の ssh 鍵署名 (ssh-rsa または ssh-dss) を使用して鍵を送信します。それに失敗すると、クライアントは再び証明書署名 (x509-sign-rsa または x509-sign-dss) の使用を試みます。

このオプションがオフの場合、クライアントは証明書署名を最初に提示します。これは、証明書鍵の種類が必要であり、サーバがクライアントに対して署名の種類が異なる同じ鍵を 2 回目の認証で使用することを許可していない場合に便利です。

#鍵エージェント

エージェントの転送を許可

鍵エージェント接続の転送を有効にします。エージェントの転送を有効にする場合は注意を必要とします。(エージェントの UNIX ドメインソケットに関して) リモートホスト上でファイルのアクセス許可をバイパスする能力を持つユーザは、転送された接続を介してローカルエージェントにアクセスできます。攻撃者は、エージェントから鍵類を取得できませんが、鍵に関して、エージェントに読み込まれた ID による認証を可能にする操作を実行できます。

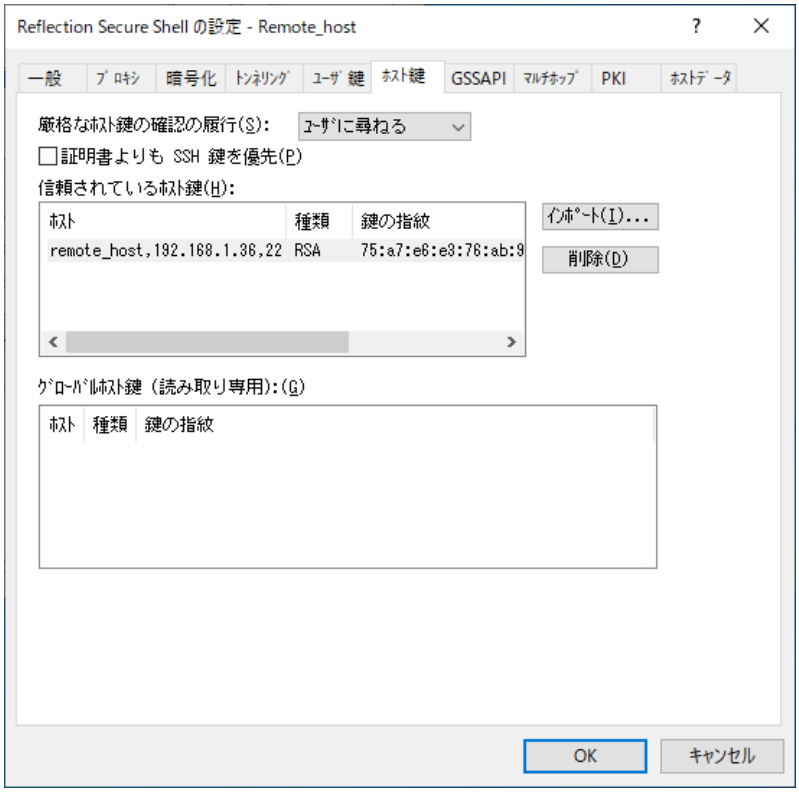
ホストの認証に使用する鍵を鍵エージェントに追加

この設定は、[エージェントの転送を許可する] が有効な場合に使用できます。これがオンになっており、サーバに対する公開鍵認証が成功すると、認証に使用された鍵または証明書が自動的に 鍵エージェントに追加されます。この鍵は鍵エージェントには保存されませんが、鍵エージェントが実行されている間は使用できます。

[鍵エージェント起動]ボタン

鍵エージェントを起動します。

6) [ホスト鍵]タブ



厳格なホスト鍵の確認の履行

不明ホストに接続する際に、Reflection がホスト鍵の確認を処理する方法を指定します。オプションは次のとおりです。

「**ユーザに尋ねる**」(既定値)

不明なホストに接続すると、[ホスト鍵の信頼性] 確認のダイアログボックスが表示されます。

「**はい**」(安全性が最も高い)

ホストが信頼できるホストでない場合、接続は拒否されます。接続前に、ホスト鍵を信頼するホスト鍵の一覧に追加する必要があります。

「**いいえ**」(安全性が最も低い)

ホストキーのチェックは行われません。この接続は、認証ダイアログボックスを表示せずに行われます。信頼する鍵の一覧にホスト鍵を追加しません。

証明書よりも SSH 鍵を優先

ホスト鍵アルゴリズムの優先順位を指定します。この設定を選択しない (既定値にする) と、Reflection はホスト鍵の前にホスト証明書を要求します。この設定を選択すると、Reflection はホスト証明書の前にホスト鍵を要求します。

この設定は、証明書と標準ホスト鍵認証の両方をサーバに構成する場合に便利です。SSH プロトコルでは、ホストの認証試行を 1 回しか許可しません。ホストが証明書を提示し、証明書を使用したホスト認証がクライアントに構成されていない場合、接続に失敗します(これは、複数の認証試行に対応しているユーザ認証とは異なります)。

信頼されているホスト鍵

現在の Windows ユーザに信頼されているホストの一覧を表示します。[インポート] と [削除] を使用してこの一覧の内容を変更できます。

既定では、この一覧に含まれないホストに接続しようとすると、新しいホスト鍵を信頼するかどうかを確認されます。この要求への応答で [常時] を選択すると、ホストが [信頼されているホスト鍵] 一覧に追加されます。

[インポート]ボタン

ホストの公開鍵を [信頼されているホスト鍵] 一覧に追加します。

[削除]ボタン

選択した鍵を [信頼されているホスト鍵] 一覧から削除します。

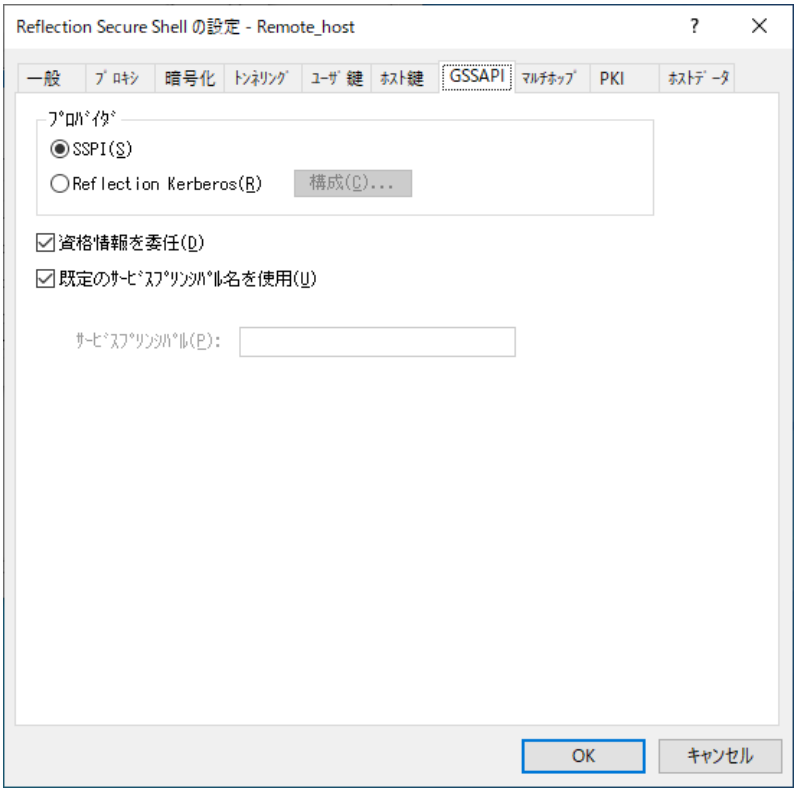
注意:確認プロンプトは表示されません。また、この操作を取り消すことはできません。

グローバルホスト鍵

コンピュータのすべてのユーザが使用できる信頼されているホスト鍵の一覧を表示します。この一覧の項目は、表示はできますが編集はできません。

システム管理者は、グローバルの既知のホストファイルを使用して [グローバルホスト鍵] 一覧を変更できます。

7) [GSSAPI]タブ



#プロバイダ

SSPI

Microsoft の Security Services Provider Interface (SSPI) を使用すると、Secure Shell サーバに対する認証に Windows ドメインのログイン資格情報を使用できます。この設定を行うと、Reflection Kerberos クライアントの構成が不要になるため、セットアップが簡単になります。

Reflection Kerberos

Kerberos/GSSAPI 認証のために Reflection Kerberos クライアントを使用します。Reflection Kerberos クライアントを使用して接続する前に、使用するコンピュータで Reflection Kerberos を構成する必要があります。

[構成]ボタン

Reflection Kerberos クライアントを構成します。このボタンは、[Reflection Kerberos] が GSSAPI プロバイダとして選択されている場合のみ使用できます。

資格情報を委任

GSSAPI がホストへ Kerberos 発券許可チケット (TGT) を転送するかどうかを指定します。

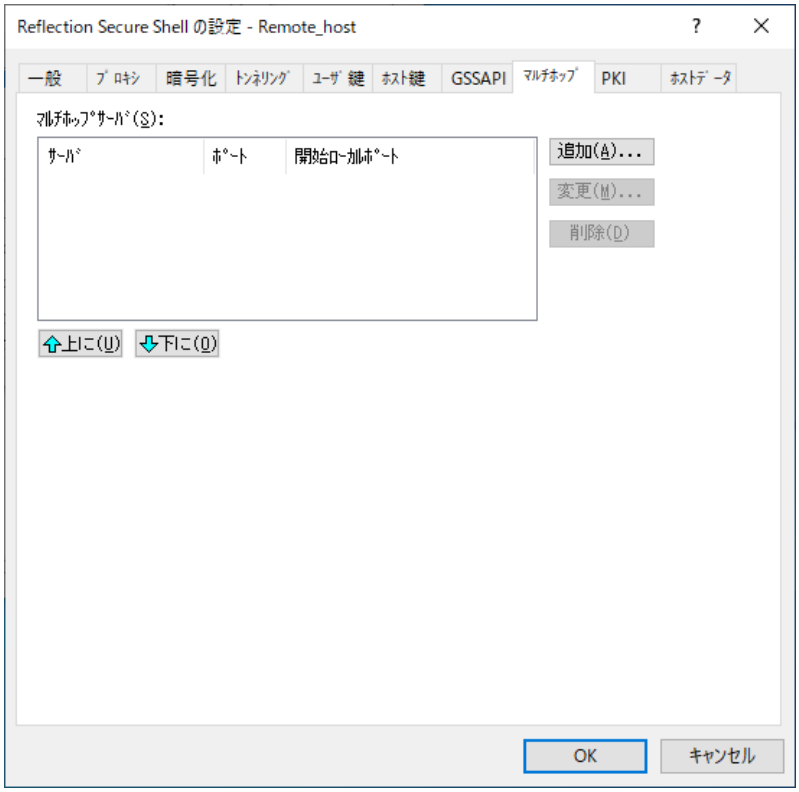
既定のサービスプリンシパル名を使用

Kerberos Key Distribution Center (KDC) へサービスチケット要求を送信するのに使用する名前を指定します。ホスト名の値は、接続先の Secure Shell サーバの名前になります。レルムの値は、選択した GSSAPI プロバイダによって異なります。

サービスプリンシパル

既定ではないサービスプリンシパル値を指定します。

8) [マルチホップ]タブ



マルチホップサーバ

マルチホップシーケンスに含まれるサーバを表示します。Reflection は、指定したローカルポートから、リモートサーバ上の指定したポートへ新しい SSH トンネルを確立します。一覧にある各接続は、その上に表示されている接続によって設定されたトンネルを経由して送信されます。一覧内の順序を変更するには、矢印ボタンを使用します。

[追加]ボタン

[マルチホップサーバの構成] ダイアログボックスを使って新しいサーバを一覧へ追加します。

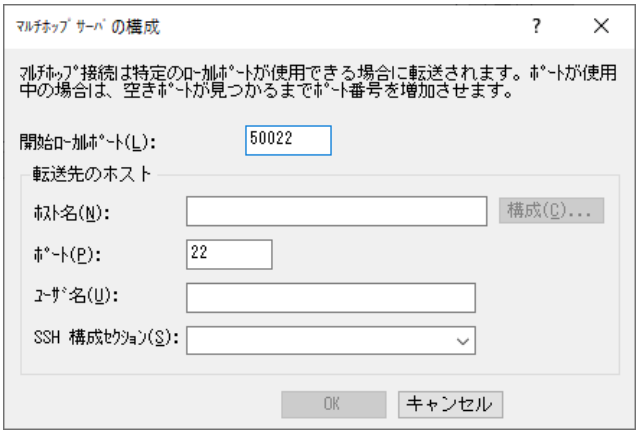
[変更]ボタン

選択したサーバを変更します。

[削除]ボタン

選択したサーバを削除します。

8-1) [マルチホップサーバの構成]ダイアログボックス



開始ローカルポート

これは、ローカルの Windows ワークステーションのポートです。マルチホップ接続は、指定のポート（使用可能な場合）から転送されます。ポートが使用中の場合は、使用可能なポートが見つかるまでポート番号が増分されます。

#転送先のホスト

ホスト名

送信データを通過させるホストコンピュータを指定します。

[構成]ボタン

[Reflection Secure Shell の設定] ダイアログボックスを開きます。このダイアログボックスでは、このトンネルに既定以外の設定を構成できます。注意: 既定以外の設定を構成する別の方法として、この接続に [SSH 構成セクション] を指定するやり方もあります。

ポート

データの送信先リモートホストのポートを指定します。既定では、多くの SSH サーバで使用されているポートである「22」が設定されています。

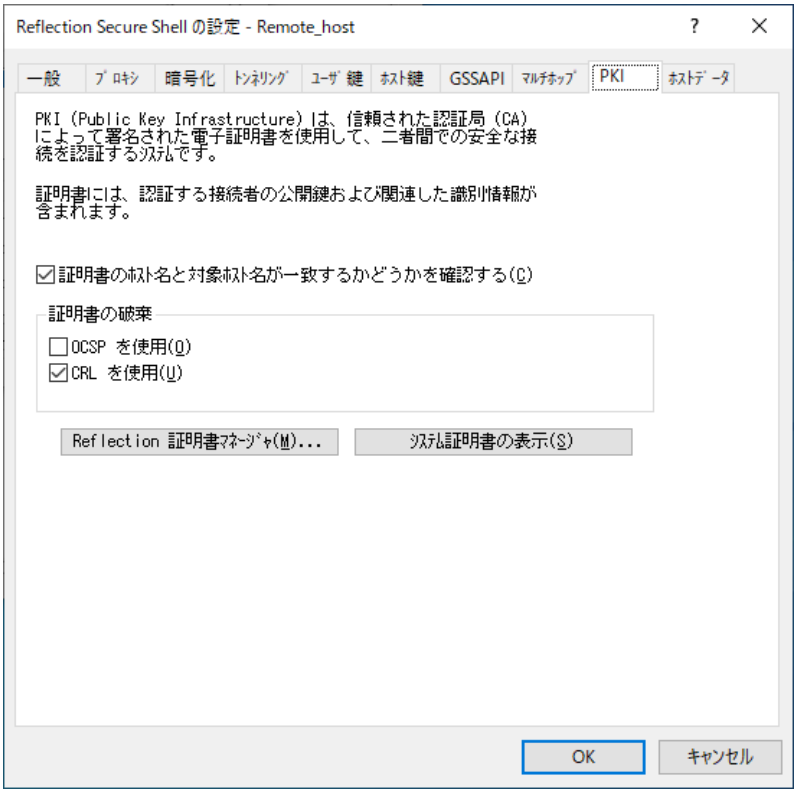
ユーザ名

（オプション）元の接続用に指定したユーザとは異なるユーザ名がホストで必要な場合、ここで名前を指定します。

SSH 構成セクション

（オプション）この接続に使用する SSH 構成セクションを指定します(Reflection では、既定でホスト名が使用されます)。

9) [PKI]タブ



証明書のホスト名と対象ホスト名が一致するかどうかを確認する

ホスト証明書の確認時にホスト名の一致が必要かどうかを指定します。この設定が有効な場合 (既定値)、セッションのために構成したホスト名が、証明書の [CommonName] フィールドまたは [SubjectAltName] フィールドに入力されているホスト名に一致していなければなりません。

#証明書の破棄

OCSP を使用

ホストの証明書の検証時に、OCSP (Online Certificate Status Protocol) レスポンダを使用して Reflection で証明書の失効を確認するかどうかを指定します。証明書自体の AIA 拡張に OCSP レスポンダが指定される場合もあります。Reflection 証明書マネージャの [OCSP] タブを使用して OCSP レスポンダを指定することもできます。

CRL を使用

ホストの証明書の検証時に、CRL (Certificate Revocation Lists) を使用して Reflection で証明書の失効を確認するかどうかを指定します。証明書自体の CDP 拡張に CRL が指定される場合もあります。Reflection 証明書マネージャの [LDAP] タブを使用して CRL を指定することもできます。

注意: この設定の既定値は、CRL の確認に関する現在のシステム設定に従います。システム設定を表示して編集するには、Internet Explorer を起動して、[ツール] - [インターネットオプション] - [詳細設定] に進みます。[セキュリティ] の下の [サーバー証明書の取り消しを確認する] を探します。

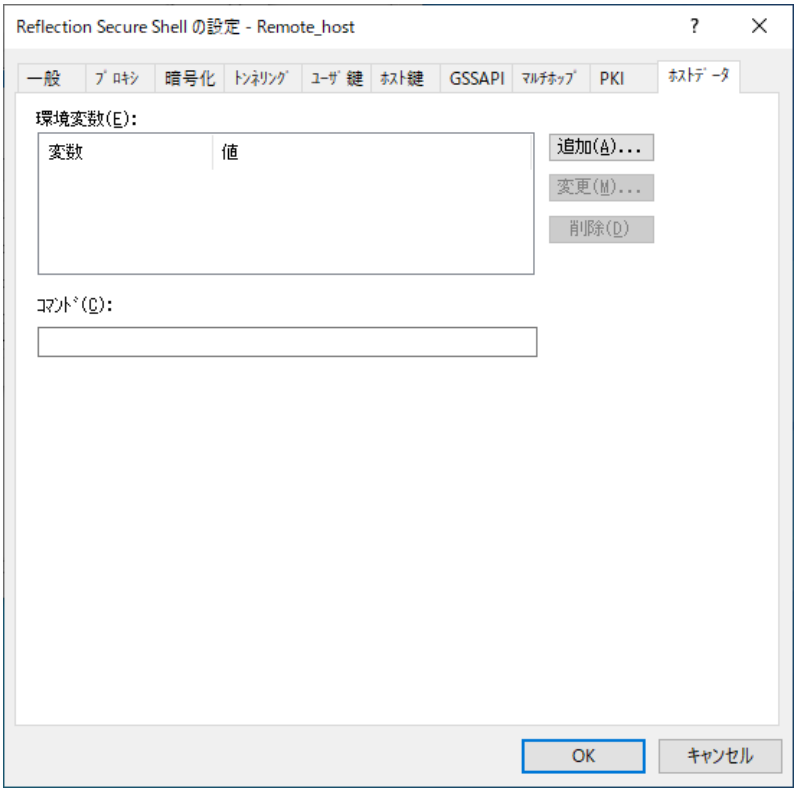
[Reflection 証明書マネージャ]ボタン

Reflection 証明書マネージャを開きます。ここで Reflection の証明書管理者格納場所内の証明書を管理し、PKI 構成を指定することができます。

[システム証明書の表示]ボタン

Windows 証明書マネージャを開きます。システムの格納場所にある証明書を管理することができます。

10) ［ホストデータ］タブ



環境変数

［追加］ボタン

［新規環境変数］ダイアログボックスが開き、新しい変数と値を指定できます。

［変更］ボタン

選択した変数を編集します。

［削除］ボタン

選択した変数を削除します。

コマンド

リモートサーバで実行する 1 つまたは複数のコマンドを指定します。UNIX サーバへの接続時には、セミコロン（;）を使用して複数のコマンドを区切ります。Windows サーバへの接続時には、アンパサンド記号（&）を使用してコマンドを区切ります。接続の確立後、サーバは指定したコマンドを実行（または実行を試行）し、セッションが終了します。サーバは、クライアントから受信したコマンドの実行を許可するよう構成されている必要があります。

コマンドは、正しい形式でサーバに指定する必要があります。例えば、UNIX サーバに一覧表示されているディレクトリをキャプチャするには、以下のように指定する必要があります。

```
ls > list.txt
```

Windows サーバで同等のコマンドは、サーバの構成方法に応じて以下のいずれかになります。

```
dir > list.txt
cmd /c dir > list.txt
```

5.2 コマンド

5.2.1 ssh コマンドラインユーティリティ

構文： ssh [options] [user@]hostname [host command]

オプション：

-A

認証エージェント転送を有効にします。これは、ホスト単位で構成ファイルに指定することもできます。エージェントの転送を有効にする場合は注意を必要とします。リモートホストでファイル権限を回避できるユーザは、転送された接続を介してローカルエージェントにアクセスできます。攻撃者は、エージェントから鍵類を取得できませんが、鍵に関して、エージェントに読み込まれた ID による認証を可能にする操作を実行できます。

-a

認証エージェント転送を無効にします（デフォルト値）

-b *bind_address*

複数インタフェースまたは別名アドレスを使用して、マシンから送信するインタフェースを指定します。

-c *cipher_spec*

優先順に指定した暗号のカンマ区切りリスト。既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour」です。接続が FIPS モードで行われるように設定されている場合、既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc」です。
プロトコルバージョン 1（廃止される可能性があるため、推奨されません）では、単一の暗号の指定が許可されています。対応する値は「3des」、「blowfish」、「des」です。

-C

すべての送信データの圧縮を有効にします。圧縮はモデム回線やほかの低速接続に適していますが、高速のネットワークでは応答速度の低下を招くだけです。

-e *escape_character*

端末セッションのエスケープ文字を設定します。既定の文字はチルダ（~）です。エスケープ文字を「none」に設定すると、使用できるエスケープ文字はなくなり、チルダは他の文字と同様に機能します。次のエスケープシーケンスを使用できます。（波型符号を指定した escape_character で置換します）。
~. 接続を終了します。
~R rekey を要求します（SSH プロトコル 2 のみ）。
~# 転送された接続を一覧表示します。
~? 使用可能なエスケープシーケンスを表示します。
~~ エスケープ文字をホストに送信します。

-E *provider*

指定したプロバイダを外部鍵プロバイダとして使用します。

-f

コマンドが実行される直前に、クライアントをバックグラウンドに配置します。

-F *config_file*

この接続に使用する代替構成ファイルを指定します。構成ファイルをコマンドラインで指定した場合、ほかの構成ファイルは無視されます。

-g

ゲートウェイポートを有効にします。リモートホストは、ローカル転送ポートへの接続を許可されます。

-h

コマンドラインオプションに関する簡単な説明を表示します。

-H *scheme*

この接続に使用する SSH 構成セクションを指定します。

-i *key_file*

鍵認証に使用する秘密鍵を指定します。鍵ファイルは、ホスト単位で構成ファイルに指定することもできます。複数の -i オプション（および構成ファイルに指定した複数の鍵）を指定できます。ファイルまたはパスが空白を含む場合、引用符を使用します。

-k *directory*

config、ホスト鍵、ユーザ鍵ファイルの代替位置を指定します。注意: -k が使用されていると、既知のホストファイルが指定した場所にすでに存在していない場合のみ、その場所からホスト鍵の読み取りと書き込みが行われます。既知のホストファイルが見つからない場合、既定の場所で既知のホストファイルにホスト鍵の読み取りと書き込みが行われます。

-l *login_name*

リモートコンピュータでのログインに使用する名前を指定します。これを構成ファイルに指定することもできます。

-L *localport:remotehost:hostport*

指定されたローカルポートからのデータを、安全なトンネルを介して、指定された宛先ホストおよびポートにリダイレクトします。ローカルポート転送.ポート転送を構成ファイルに指定することもできます。管理者としてログインしないかぎり、権限ポート（ポート番号 1024 以下）を転送できません。IPv6 アドレスは、port/host/hostport というもう 1 つの構文を使用して指定できます。

-m *mac_spec*

この接続に使用する 1 つまたは複数のカンマ区切り MAC（メッセージ認証コード）アルゴリズムを指定します。アルゴリズムを優先順に指定します。既定値は「hmac-sha1,hmac-sha256,hmac-sha512,hmac-md5,hmac-ripemd160,hmac-sha1-96,hmac-md5-96」です。接続が FIPS モードで実行するように設定されている場合、既定値は「hmac-sha1,hmac-sha256,hmac-sha512」です。

-N

リモートコマンドを実行しません。転送するポートのみを構成する場合に便利です(プロトコルバージョン 2 のみ)。

-o *option*

構成ファイルで対応するオプションを指定します。次に例を示します。

ssh ″-o FIPSMode=yes″ myuser@myhost

-p *port*

サーバに接続するポートを指定します。既定値 (22) は、Secure Shell 接続の標準ポートです。これは、構成ファイルにホスト単位で指定できます。

-q

クワイエットモードを有効にします。 このモードでは、バナーを含むすべての警告および診断メッセージが表示されません。

-R *localport:remotehost:hostport*

(Secure Shell サーバを実行するコンピュータ上の) 指定されたリモートポートからのデータを、安全なトンネルを介して、指定された宛先ホストおよびポートにリダイレクトします。リモートポート転送.ポート転送を構成ファイルに指定することもできます。管理者としてログインしないかぎり、権限ポート (ポート番号 1024 以下) を転送できません。IPv6 アドレスは、port/host/hostport というもう 1 つの構文を使用して指定できます。

-S

シェルを実行しないでください。

-t

コマンドが指定されている場合も TTY を強制的に割り当てます。

-T

pseudo-tty 割り当てを無効にします。

-v

デバッグレベルを冗長モードに設定します。 これは、デバッグレベルを 2 に設定することと同じです。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-x

X11 接続の転送を無効にします。

-X

X11 接続の転送を有効にし、X11 クライアントを信頼されないものとして扱います。信頼されないリモート X11 クライアントは、信頼される X11 クライアントに属するデータを不正に変更できません。

X11 転送を有効にする場合は注意が必要です。ユーザの X 認可データベースのリモートホストでファイル権限を回避できるユーザは、転送された接続を介してローカル X11 ディスプレイにアクセスできます。攻撃者は、キーストローク監視などの行動を実行できる可能性があります。

-Y

X11 接続の転送を有効にし、X11 クライアントを信頼関係があるクライアントとして扱います。

X11 転送を有効にする場合は注意が必要です。ユーザの X 認可データベースのリモートホストでファイル権限を回避できるユーザは、転送された接続を介してローカル X11 ディスプレイにアクセスできます。攻撃者は、キーストローク監視などの行動を実行できる可能性があります。

-1

ssh がプロトコルバージョン 1 のみを試行するようにします。プロトコルバージョン 1 は廃止される可能性があるため、推奨されません。

-2

ssh にプロトコルバージョン 2 のみを試行させます。

-4

IPv4 アドレスのみを使用して接続させます。

-6

IPv6 アドレスのみを使用して接続させます。

5.2.2 sftp コマンドラインユーティリティ

構文： sftp [options] [user@]host[#port]:source_file
[user@]host[#port][:destination_file]

コマンドラインオプション：

-a

ASCII モードでファイルを転送します。

-b *buffersize*

1 つの要求の最大バッファサイズを設定します。有効な値は 1024 ～ 32768 です。

-B *batchfile*

ログイン成功後、指定したバッチファイルで各コマンドを実行し、接続を終了します。例えば、以下のコマンドでは myname を使用して myhost に接続し、myfile でコマンドを実行します。ファイル内のすべてのコマンドを実行後、接続が終了します。

sftp -B c:¥mypath¥myfile myhost.com myname

バッチファイルでは、以下に示す対話型コマンドのいずれかを使用できます。

メモ:セミコロンは、-B オプションを使って ｓｆｔｐ コマンドに提供されたスクリプト内のコメントには使用できません。これらのバッチファイル内のコメントに印を付けるには、番号記号 (#) を使用します。

-c *cipher*

優先順に指定した暗号のカンマ区切りリスト。既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour」です。接続が FIPS モードで行われるように設定されている場合、既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc」です。

プロトコルバージョン 1（廃止される可能性があるため、推奨されません）では、単一の暗号の指定が許可されています。対応する値は「3des」、「blowfish」、「des」です。

-C

すべての送信データの圧縮を有効にします。圧縮はモデム回線やほかの低速接続に適していますが、高速のネットワークでは応答速度の低下を招くだけです。

-d

ターゲットをディレクトリにします。

-F *config_file*

この接続に使用する代替構成ファイルを指定します。構成ファイルをコマンドラインで指定した場合、ほかの構成ファイルは無視されます。

-h

コマンドラインオプションに関する簡単な説明を表示します。

-H *scheme*

この接続に使用する SSH 構成セクションを指定します。

-i *key_file*

鍵認証に使用する秘密鍵を指定します。鍵ファイルは、ホスト単位で構成ファイルに指定することもできます。複数の -i オプション（および構成ファイルに指定した複数の鍵）を指定できます。ファイルまたはパスが空白を含む場合、引用符を使用します。

-k *directory*

config、ホスト鍵、ユーザ鍵ファイルの代替位置を指定します。注意: -k が使用されていると、既知のホストファイルが指定した場所にすでに存在していない場合のみ、その場所からホスト鍵の読み取りと書き込みが行われます。既知のホストファイルが見つからない場合、既定の場所で既知のホストファイルにホスト鍵の読み取りと書き込みが行われます。

-m *mac_spec*

この接続に使用する 1 つまたは複数のカンマ区切り MAC（メッセージ認証コード）アルゴリズムを指定します。アルゴリズムを優先順に指定します。既定値は「hmac-sha1,hmac-sha256,hmac-sha512,hmac-md5,hmac-ripemd160,hmac-sha1-96,hmac-md5-96」です。接続が FIPS モードで実行するように設定されている場合、既定値は「hmac-sha1,hmac-sha256,hmac-sha512」です。

-o *option*

構成ファイルで対応するオプションを指定します。次に例を示します。

ssh ″-o FIPSMode=yes″ myuser@myhost

-p

タイムスタンプとファイル属性を保持します。

-P *port*

リモートホストに接続するポート。

-q

クワイエットモードを有効にします。 このモードでは、バナーを含むすべての警告および診断メッセージが表示されません。

-Q

進行状況のインジケータの表示をオフにします。

-R *maximum_requests*

同時要求の最大数を指定します。この数を増やすと、ファイル転送速度が多少向上しますが、メモリ使用量が増えます。既定値は未処理要求数 16 個です。

-s *subsystem*

ssh サブシステムを指定します。

-S *program*

暗号化された接続に使用するプログラム。

-u

コピー後、ソースファイルを削除します。

-v

デバッグレベルを冗長モードに設定します。 これは、デバッグレベルを 2 に設定することと同じです。

-V

製品名およびバージョン情報を表示して終了します。コマンドラインで他のオプションが指定された場合、それらは無視されます。

-4

IPv4 アドレスのみを使用して接続させます。

-6

IPv6 アドレスのみを使用して接続させます。

インタラクティブモード:

auto

転送モードを自動に設定します。

binary

転送の種類をバイナリに設定します。

bye

sftp を終了します。

cd *path*

リモートディレクトリを path に変更します。

chmod *path*

path に関連付けられている許可を変更します。mode を使用して、3 桁の数値による許可を指定します。

lcd *path*

ローカルディレクトリを path に変更します。

exit

sftp を終了します。

get *remote-path [local-path]*

remote-path を取得し、ローカルマシンに保存します。ローカルパス名が指定されていない場合、リモートマシンと同じ名前が指定されます。

getext *[extension、extension...]*

ASCII 転送を使用するファイル拡張子を表示します。setext を使用して、この一覧を変更します。

help

ヘルプテキストを表示します。

lls *[ls-options [path]]*

path が指定されていない場合に、path または現在のディレクトリのローカルディレクトリ一覧を表示します。

lmkdir *path*

path で指定されたローカルディレクトリを作成します。

lpwd

ローカル作業ディレクトリを印刷します。

ls *[path]*

path が指定されていない場合に、path または現在のディレクトリのリモートディレクトリ一覧を表示します。

mkdir *path*

path で指定されたリモートディレクトリを作成します。

progress

進捗メーターの表示を切り替えます。

put *local-path [remote-path]*

local-path をリモートマシンに転送します。リモートパス名が指定されていない場合、ローカルマシンと同じ名前が指定されます。

pwd

リモート作業ディレクトリを表示します。

quit

sftp を終了します。

reget *remote-file [local-file]*

指定した転送を再開します。これは get コマンドのように機能しますが、部分的に書き込まれたローカルファイルの存在を確認し、見つかった場合は最後に試行が中止された場所から転送を開始します。

rename *oldpath newpath*

リモートファイルの名前を oldpath から newpath に変更します。

rmdir *path*

path で指定されたリモートディレクトリを削除します。

rm *path*

path で指定されたリモートファイルを削除します。

setext *[extension、extension...]*

ASCII 転送を使用するファイル拡張子を設定します。ワイルドカード文字を使用できます。引数が指定されていない場合、ファイル拡張子は ASCII 転送を使用しません。

version

sftp バージョンを表示します。

?

help と同じです。

5.2.3 scp コマンドラインユーティリティ

構文： scp [options] [user@host:]file1 [user@host:]file2

オプション：

-a

ASCII モードでファイルを転送します。

-b *buffersize*

1 つの要求の最大バッファサイズを設定します。

-B

バッチモードを設定すると、パスワードまたはパスフレーズの入力を求められません。パスフレーズのないユーザ鍵を使用して認証します。

-c *cipher*

優先順に指定した暗号のカンマ区切りリスト。既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour」です。接続が FIPS モードで行われるように設定されている場合、既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc」です。
プロトコルバージョン 1（廃止される可能性があるため、推奨されません）では、単一の暗号の指定が許可されています。対応する値は「3des」、「blowfish」、「des」です。

-C

圧縮を使用します。

-d

ターゲットをディレクトリにします。

-D *level*

デバッグレベルを設定します。指定可能な値は、1、2、および 3 です。

-F *configfile*

ユーザ単位の代替構成ファイルを指定します。構成ファイルをコマンドラインで指定した場合、システム規模の構成ファイルは無視されます。

-h

コマンドラインオプションに関する簡単な説明を表示します。

-H *scheme*

この接続に使用する SSH 構成セクションを指定します。

-i *keyfile*

RSA 認証または DSA 認証の識別情報（秘密鍵）を読み込むファイルを選択します。識別情報ファイルは、ホスト単位で構成ファイルに指定することもできます。複数の -i オプション（および構成ファイルに指定した複数の識別情報）を指定できます。空白を含むパス名は、二重引用符で囲む必要があります。

-k *directory*

config、ホスト鍵、ユーザ鍵ファイルの代替位置を指定します。注意: -k が使用されていると、既知のホストファイルが指定した場所にすでに存在していない場合のみ、その場所からホスト鍵の読み取りと書き込みが行われます。既知のホストファイルが見つからない場合、既定の場所で既知のホストファイルにホスト鍵の読み取りと書き込みが行われます。

-o *option*

構成ファイルで使用される形式でオプションを指定するために使用できます。これは、別個のコマンドラインフラグがないオプションを指定する場合に便利です。対応するオプションの一覧については、「構成ファイルのキーワード参照」を参照してください。

--overwrite

既存の転送先ファイルを上書きするかどうかを指定します。指定可能な値は「yes」および「no」です。既定値は「yes」です。

-p

タイムスタンプとファイル属性を保持します。

-P *port*

リモートホストに接続するポート。

-q

低度モード.バナーを含むすべての警告メッセージと診断メッセージを表示しないようにします。

-Q

進行状況のインジケータの表示をオフにします。

-r

すべてのサブディレクトリを含むディレクトリを再帰的にコピーします。

-u

コピー後、ソースファイルを削除します。

-v

高度モード。ssh により進捗状況に関するデバッグメッセージが表示されます。これは、接続、認証、構成に関する問題のデバッグに役立ちます。複数の -v オプションを使用すると、詳細度が高くなります。最大は 3（-vvv）です。

-V

バージョン番号とアプリケーション情報を表示します。

-z

既定では、すべてのダウンロードについて、ファイル名の照合では大文字小文字が区別されます。このオプションを使用すると、ダウンロードで、サーバのファイル名指定にワイルドカードを使用した場合、大文字小文字は区別されません。

-1

プロトコルバージョン 1 のみにします。このオプションは、ssh トンネルを介して rcp を使用する OpenSSH サーバにもファイルを転送します。

-2

プロトコルバージョン 2 のみにします。

-4

IPv4 アドレスのみを使用します。

-6

IPv6 アドレスのみを使用します。

5.2.4 ssh-keygen コマンドラインユーティリティ

構文： ssh-keygen [-b bits] -t type [-N [passphrase]] [-C comment] [-f output_keyfile]

ssh-keygen -B [-f input_keyfile]

ssh-keygen -c [-P passphrase] [-C comment] [-f keyfile]

ssh-keygen -e [-f input_keyfile]

ssh-keygen -p [-P old_passphrase] [-N new_passphrase] [-f keyfile]

ssh-keygen -i [-f input_keyfile]

ssh-keygen -y [-f input_keyfile]

ssh-keygen -l [-f input_keyfile]

オプション：

-b *bits*

鍵のサイズを指定します。.鍵のサイズを大きくすると、ある程度までセキュリティは向上します。鍵のサイズを大きくすると最初の接続が遅くなりますが、正常に接続した後は、鍵のサイズはデータストリームの暗号化や解読の速度に影響しません。使用する鍵の長さは、多くの要素に依存します。 その要素には、鍵の種類、鍵の有効期間、保護するデータの値、潜在的な攻撃者にとって利用可能なリソース、この非対称鍵とともに使用する対称鍵のサイズなどがあります。ニーズに合った最適な選択をするには、セキュリティ管理者にお問い合わせください。鍵のサイズは、64 ビットで均等に割り切れる次の値に切り上げられます。DSA 鍵の既定値は 1024 ビットで、RSA 鍵の既定値は 2048 ビットです。

-B

指定された鍵の指紋を、SHA-1 Bubble Babble 形式で表示します。-f を使用して、鍵ファイルを指定できます。 ファイルを指定していない場合、ファイル名の入力を求められます。秘密鍵または公開鍵の名前を指定できますが、どちらの場合でも、公開鍵は使用可能でなくてはなりません。

-c

秘密鍵ファイルと公開鍵ファイルのコメントの変更を要求します。この操作は、RSA1 鍵にのみ対応します。秘密鍵を含むファイル、鍵にパスフレーズがある場合はパスフレーズ、新しいコメントの入力をプログラムにより求められます。

-C *comment*

鍵ファイル内のコメントフィールドに情報を指定します。文字列に空白が含まれている場合、引用符を使用します。鍵の作成時にコメントを指定していない場合、鍵の種類、作成者、日付、時刻を含む既定のコメントが作成されます。

-e

指定した OpenSSH 公開鍵または秘密鍵を使用して、Reflection 形式の公開鍵を生成します。-f を使用して、鍵ファイルを指定できます。ファイルを指定していない場合、ファイル名の入力を求められます。

-f *ファイル名*

生成される秘密鍵のファイル名を指定します(公開鍵も作成され、常に秘密鍵と同じ名前となり、.pub というファイル拡張子が付きます)。このオプションは、入力ファイル名を指定するため -e、-i、-l、-p、-y、および -B と組み合わせて使用することもできます。

-i

指定された Reflection 公開鍵を OpenSSH 形式に変換します。-f を使用して、鍵ファイルを指定できます。ファイルを指定していない場合、ファイル名の入力を求められます。

-h

コマンドラインオプションに関する簡単な説明を表示します。

-l

MD5 ハッシュを使用し、指定された公開鍵ファイルの指紋を表示します。-f を使用して、鍵ファイルを指定できます。ファイルを指定していない場合、ファイル名の入力を求められます。秘密鍵を指定した場合、ssh-keygen では一致する公開鍵ファイルの検索を試行し、その指紋を印刷します。

-N *パスフレーズ*

パスフレーズを設定します。例えば、新しい鍵のパスフレーズを指定するには、次のようにします。

ssh-keygen -N mypassphrase -f keyfile

パスフレーズ保護されていない新しい鍵を作成するには、次のようにします。

ssh-keygen -N -f keyfile

-N を、-p および -P と組み合わせて使用すると、既存の鍵のパスフレーズを変更できます。

-p

このオプションを使用して、既存の秘密鍵のパスフレーズを変更します。このオプションのみを使用する場合、秘密鍵を含むファイル、古いパスフレーズ、新しいパスフレーズ (2 回) の入力をプログラムにより求められます。-f、-P、および -N と組み合わせて使用し、パスフレーズを非対話形式で変更できます。次に例を示します。

ssh-keygen -p -f keyfile -P oldpassphrase -N newpassphrase

-P *パスフレーズ*

(古い) パスフレーズを指定します。

-q

ssh-keygen を終了します。

-t *type*

鍵の生成に使用する鍵のアルゴリズムを指定します。プロトコルバージョン 2 に指定可能な値は「rsa」または「dsa」です。

-y

指定した秘密鍵を使用して、公開鍵の新しいコピーを取得します。-f を使用して、鍵ファイルを指定できます。ファイルを指定していない場合、ファイル名の入力を求められます。

返される値

コマンドが正常に完了した場合、ssh-keygen は 0 (ゼロ) を返します。ゼロ以外の値は失敗を示します。

5.3 構成ファイルのキーワード参照

5.3.1 Secure Shell 設定

AddAuthKeyToAgent

この設定は、ForwardAgent が「yes」に設定されている場合に、クライアントの公開鍵認証の処理方法に影響します。サーバの公開鍵認証が成功し、ForwardAgent と AddAuthKeyToAgent の両方が「yes」に設定されている場合、認証に使用された鍵や証明書は 鍵エージェントに自動的に追加されます。この鍵は鍵エージェントには保存されませんが、鍵エージェントが実行されている間は使用できます。AddAuthKeyToAgent が「no」(既定値) に設定されている場合、鍵と証明書が鍵エージェントに自動的に追加されることはありません。鍵エージェントは、手動でインポートされた鍵のみを使用します。

AuthUseAllKeys

この設定は、クライアントの公開鍵認証の処理方法に影響します。この設定が「no」(既定値) の場合、クライアントは IdentityFile というキーワードを使用して指定した 1 つまたは複数の鍵のみを使用して認証しようとします。この設定が「yes」の場合、クライアントは使用できる公開鍵全部を使用して認証しようとします。

BatchMode

スクリプトおよびバッチジョブに使用可能なパスワードメッセージおよびパスフレーズメッセージを含む、ユーザ入力要求をすべて無効にするかどうかを指定します。指定可能な値は「yes」および「no」です。既定値は「no」です。

メモ:このキーワードは、キーボード対話型の認証構成時のユーザ入力要求を無効にしませんが、BatchMode 有効時にキーボード対話型を使用する接続は失敗します。

BindAddress

複数インタフェースまたは別名アドレスを使用して、コンピュータから送信するインタフェースを指定します。

ChallengeResponseAuthentication

試行/応答認証を使用するかどうかを指定します。引数は「yes」または「no」にする必要があります。この認証方式は、SecurID、PAM 認証、またはサーバからのメッセージとユーザからの応答が必要なその他の外部認証方式を使用している場合に推奨されます。既定値は「yes」です。これは、対応している SSH プロトコル 1 のみに適用されますが、推奨されません。SSH プロトコルバージョン 2 には KbdInteractiveAuthentication を使用します。

CheckHostIP

このフラグを「yes」に設定した場合、Secure Shell クライアントは、ホストの公開鍵のほか、known_hosts ファイルのホスト IP アドレスを確認します。既知のホスト一覧のホスト IP が、接続に使用している IP アドレスに一致する場合にのみ、接続が許可されます。既定値は「no」です。注意: StrictHostKeyChecking が「no」の場合、この設定は適用されません。

CheckHostPort

このフラグを「yes」に設定した場合、Secure Shell クライアントは、ホストの公開鍵のほか、known_hosts ファイルのホストポートを確認します。既知のホスト一覧のホストポートが、接続に使用しているポートに一致する場合にのみ、接続が許可されます。既定値は「no」です。注意: StrictHostKeyChecking が「no」の場合、この設定は適用されません。

Cipher

プロトコルバージョン 1 のセッションの暗号化に使用する暗号を指定します。現在、「blowfish」、「3des」、「des」に対応しています。des は、3des 暗号に対応していない従来のプロトコル 1 実装との相互運用性のため Secure Shell クライアントのみが対応しています。暗号上の弱点があるため、使用することは推奨されません。既定値は「3des」です。

Ciphers

プロトコルバージョン 2 に使用可能な暗号を優先順に指定します。複数の暗号は、カンマで区切る必要があります。既定値は、「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour」です。接続が FIPS モードで行われるように設定されている場合、既定値は「aes128-ctr,aes128-cbc,aes192-ctr,aes192-cbc,aes256-ctr,aes256-cbc,3des-cbc」です。

ClearAllForwardings

ローカル転送、リモート転送、または動的転送されたポートのうち、既に処理されたすべてのポートを構成ファイルまたはコマンドラインからクリアします。注意: scp および sftp は、この設定に関係なく、すべての転送ポートを自動的にクリアします。指定可能な値は「yes」および「no」です。既定値は「no」です。

Compression

圧縮を有効にするかどうかを指定します。圧縮は、モデム回線などの低速接続には向いていますが、高速ネットワークでは応答速度を低下させます。また、圧縮はパケットをより不規則にするため、悪意のある人物がパケットを解読することが難しくなります。指定可能な値は「yes」および「no」です。既定値は「no」です。

CompressionLevel

圧縮を有効にした場合に使用する圧縮レベルを指定します。このオプションは、プロトコルバージョン 1 にのみ適用されます。引数は、1（高速）から 9（低速、最適）までの整数にする必要があります。既定レベルは 6 で、ほとんどのアプリケーションに有効です。値の意味は gzip と同じです。

ConnectionAttempts

終了前に試行する回数（1 秒間に 1 回）を指定します。引数は整数にする必要があります。これは、接続に失敗することがある場合にスクリプトで使用できます。デフォルトは 1 です。

ConnectionReuse

同一ホストへの複数のセッションで元の Secure Shell 接続を再利用するかどうかを指定します。このため、再認証が不要となります。引数は「yes」または「no」にする必要があります。「yes」に設定すると、ホスト名、ユーザ名、SSH 構成セクション（使用する場合）がすべて一致する場合、新しい接続は既存のトンネルを再利用します。「no」に設定すると、Reflection ではセッションごとに新しい接続を確立します。つまり、新しい接続ごとに認証処理を繰り返し、変更された接続固有の設定（転送および暗号など）も適用されます。Reflection ウィンドウを使用した接続で接続する場合、既定値は「yes」になります。接続に コマンドラインユーティリティを使用している場合は「no」になります。詳細については、Secure Shell セッションにおける接続の再利用を参照してください。

ConnectTimeout

サーバへの接続完了を試行している時にクライアントが待機する最大時間（秒単位）を指定します。タイマーは、接続の確立時（ログオン前）に開始して、設定、ホスト鍵交換、および認証中に稼働します。実際には、タイマーは基本的に認証処理の間に稼働します。デフォルト値は 120 です。

DisableCRL

ホスト証明書の検証時に CRL（Certificate Revocation List）の確認を行うかどうかを指定します。これを「yes」に設定すると、CRL の確認が無効になります。この設定の既定値は、CRL の確認に関する現在のシステム設定に従います。システム設定を表示して編集するには、Internet Explorer を起動して、[ツール] -［インターネットオプション］-［詳細設定］コマンドに進みます。[セキュリティ] の下の［サーバー証明書の取り消しを確認する］を探します。

DynamicForward

安全なチャネルを介して転送するローカルマシンの TCP/IP ポートを指定します。アプリケーションプロトコルは、リモートマシンと接続する場所を決定するために使用されます。引数はポート番号にする必要があります。現在、SOCKS4 プロトコルに対応し、Reflection Secure Shell は SOCKS4 サーバとして動作します。複数転送を指定できるため、コマンドラインで追加転送を指定できます。管理権限のあるユーザだけが権限ポートを転送できます。

EscapeChar

エスケープ文字を設定します（既定は '~'）。エスケープ文字は、コマンドラインでも設定できます。引数は単一の文字「^」で、後に文字を続ける必要があります。あるいは、エスケープ文字を無効にするには「none」にする必要があります（バイナリデータ接続を透過にするため）。

FipsMode

この設定を「yes」にすると、アメリカ合衆国政府の連邦情報処理規格（FIPS）140-2 に合致するセキュリティプロトコルおよびアルゴリズムを使用して、接続する必要があります。この規格に合致しないオプションは、[暗号化] タブで使用できません。

メモ:この設定はキーワード Host で指定される SSH 構成セクションに影響し、同じ SSH 構成セクション（またはホスト名）を使用するように構成されていないかぎり、以降の Secure Shell セッションに影響しません。

ForwardAgent

これを「yes」に設定すると、 鍵エージェント接続の転送が有効になります。エージェントの転送を有効にする場合は注意を必要とします。(エージェントの UNIX ドメインソケットに関して) リモートホスト上でファイルのアクセス許可をバイパスする能力を持つユーザは、転送された接続を介してローカルエージェントにアクセスできます。攻撃者は、エージェントから鍵類を取得できませんが、鍵に関して、エージェントに読み込まれた ID による認証を可能にする操作を実行できます。サーバでもエージェント転送が有効になっている必要があります。既定値は「no」です。

ForwardX11

安全なチャネルを介して X11 接続を自動的にリダイレクトし、DISPLAY を設定するかどうかを指定します。引数は「yes」または「no」にする必要があります。既定値は「no」です。(注意: Reflection X を使用して Secure Shell を構成する場合は、「ForwardX11ReflectionX」を参照してください)。

ForwardX11ReflectionX

Reflection X (14.1 以降) 用の Secure Shell 接続を構成している場合にのみ、この設定を使用してください。安全なチャネルを介して X11 接続を自動的にリダイレクトし、DISPLAY を設定するかどうかを指定します。引数は「yes」または「no」にする必要があります。既定値は「yes」です。

GatewayPorts

リモートホストが、ローカル転送ポートへの接続を許可されるかどうかを指定します。既定では、Reflection Secure Shell はローカルポート転送をループバックアドレスに結合します。これによって、ほかのリモートホストが、転送ポートに接続できないようにしています。GatewayPorts を使用して、Reflection Secure Shell がローカルポート転送をワイルドカードアドレスに結合するように指定できます。これによって、リモートホストは転送ポートに接続できます。この設定を有効にする場合は注意を必要とします。この設定を使用すると、システム上の転送ポートをリモートホストが認証なしで使用できるようになるため、ネットワークおよび接続のセキュリティレベルが下がるおそれがあります。引数は「yes」または「no」にする必要があります。既定値は「no」です。

GlobalKnownHostsFile

Windows の共通アプリケーションデータフォルダーの ssh_known_hosts という既定ファイルの代わりに使用する、グローバルホスト鍵データベース用のファイルを指定します。

メモ:パスまたはファイル名の一部に空白が含まれている場合、ファイル名を引用符で囲みます。

GssapiAuthentication

GSSAPI 認証を使用して、Kerberos KDC を認証するかどうかを指定します。この設定は、使用中のプロトコルがプロトコルバージョン 2 である場合にのみ適用できます(プロトコルバージョン 1 で同等の設定は KerberosAuthentication になります)。指定可能な値は「yes」および「no」です。既定値は「no」です。

GssapiDelegateCredentials

GSSAPI を使用して、ホストに発券許可チケット (krbtgt) を転送するかどうかを指定します。この設定は、使用中のプロトコルがプロトコルバージョン 2 である場合にのみ適用できます(プロトコルバージョン 1 で同等の設定は KerberosTgtPassing になります)。指定可能な値は「yes」および「no」です。既定値は「yes」です。

GssapiUseSSPI

Microsoft の Security Support Provider Interface (SSPI) を GSSAPI 認証に使用するかどうかを指定します。この設定は、Kerberos/GSSAPI 認証が有効な場合にのみ使用できます (プロトコルバージョン 2 では GssapiAuthentication を使用、プロトコルバージョン 1 では KerberosAuthentication を使用)。このキーワードの引数は「yes」または「no」にする必要があります。「no」に設定すると、 Secure Shell クライアントは GSSAPI 認証に Reflection Kerberos クライアントを使用します。「yes」に設定すると、Secure Shell クライアントは Secure Shell サーバに対する認証に Windows ドメインのログイン資格情報 (SSPI) を使用します。SSPI はプロトコルバージョン 2 接続のみに対応しており、サーバが GSSAPI-with-mic 認証方式に対応する必要があります。既定値は「yes」です。

GssServicePrincipal

クライアントが Kerberos Key Distribution Center (KDC) へサービスチケット要求を送信する時に、使用する既定以外のサービスプリンシパル名を指定します。GSSAPI プロバイダとして SSPI を選択している場合、この設定を使用して Windows ドメインとは異なるレルムにサービスプリンシパルを指定できます。完全なホスト名の後に @ とレルム名を続けます。例えば、myhost.myrealm.com@MYREALM.COM のようになります(既定で、ホスト名の値は接続する Secure Shell サーバの名前になり、レルムは GssapiUseSSPI の値によって異なります。GssapiUseSSPI が「no」の場合、レルム名は既定のプリンシパルプロファイルで指定されます。GssapiUseSSPI が「yes」の場合、レルムは Windows ドメイン名になります)。

Host

指定した SSH 構成セクションに属する (次の Host キーワードまで) 続く宣言を指定します。文字「*」および「?」は、ワイルドカードとして使用できます。パターンに単一の「*」を使用すると、すべてのホストにグローバルな既定設定を指定できます。Reflection 接続では、最初に一致する Host 文字列 (ワイルドカード文字を含む) を使用します。以降の一致は無視されます。

メモ:[Reflection Secure Shell の設定] ダイアログボックスを閉じる時、既定の設定値は構成ファイルに保存されません。既定値を手動でファイルに追加している場合、ダイアログボックスを閉じる時にその既定値は削除されます。ワイルドカードホストのスタンザを、特定のホスト名を使用するスタンザと組み合わせて使用する場合、これにより、設計制約が課せられます。ワイルドカードのスタンザで構成された値を上書きするよう設定された特定のホストのスタンザに既定値を手動で構成した場合、ホスト固有の SSH 構成セクションの設定を表示するために [Reflection Secure Shell の設定] ダイアログボックスを開くと、既定値は削除されます。この状況は、グローバル構成ファイルを使用すると適切に処理することができます。グローバル構成ファイルは、ユーザが [Reflection Secure Shell の設定] ダイアログボックスを開いたり閉じたりしても、更新されません。

HostKeyAlgorithms

クライアントが使用するホスト鍵アルゴリズムを優先順に指定します。このオプションの既定は、「x509v3-rsa2048-sha256, x509v3-sign-rsa, x509v3-sign-dss, ssh-rsa-sha2-256@attachmate.com, ssh-rsa,ssh-dss」です。この設定は、証明書と標準ホスト鍵認証の両方をサーバに構成する場合に便利です。既定値では、通常の SSH 鍵アルゴリズムの前に x509 アルゴリズムが指定されています。SSH プロトコルでは、ホストの認証試行を 1 回しか許可しません。(これは、複数の認証方式と試みがサポートされているユーザ認証とは異なります。)ホストが証明書を提示し、クライアントが証明書を使ってホスト認証することを設定しない場合、x509 アルゴリズムを優先すると接続は失敗します。この状況では、優先順位を「ssh-rsa-sha2-256@attachmate.com、ssh-rsa,ssh-dss、x509v3-rsa2048-sha256、x509v3-sign-rsa、x509v3-sign-dss」に変更することで、クライアントを構成して証明書上で SSH キーを優先させることができます。

HostKeyAlias

ホスト鍵データベースファイルでホスト鍵の検索または保存のため、実際のホスト名の代わりに使用する別名を指定します。このオプションは、ssh 接続のトンネリングまたは単一のホストで複数のサーバを実行している場合に使用できます。

IdentityFile

鍵認証に使用する秘密鍵を指定します。ファイルはユーザーの .ssh フォルダにあります。(¥Users¥username¥Documents¥Micro Focus¥reflection¥.ssh¥).IdentityFile 項目は、[Reflection Secure Shell の設定] ダイアログボックスの [ユーザ鍵] タブの一覧から鍵または証明書を選択すると追加されます。構成ファイルに複数の識別情報ファイルを指定でき、これらのすべての識別情報が順番に試行されます。

メモ:空白が含まれている場合、完全なパス名を引用符で囲みます。

KbdInteractiveAuthentication

キーボード対話型認証を使用するかどうかを指定します。指定可能な値は「yes」および「no」です。既定値は「yes」です。この認証方式は、SecurID、PAM 認証、またはサーバからのメッセージとユーザからの応答が必要なその他の外部認証方式を使用している場合に推奨されます。パスワード有効期限または最初のログインパスワードの変更が有効になっているホストのパスワード認証で、PasswordAuthentication 方式よりも有効に機能する場合があります。認証に成功するために有効期限が切れたパスワードをリセットする必要がある場合、パスワード認証に必要な場合もあります。これは、SSH プロトコル 2 にのみ適用されます。SSH プロトコルバージョン 1 には ChallengeResponseAuthentication を使用します。

KeepAlive

システムが TCP キープアライブメッセージを相手に送信する必要があるかどうかを指定します。送信されると、接続の切断またはいずれかのマシンのクラッシュが検出されます。ネットワークのダウンまたはリモートホストの停止をクライアントが検出するため、既定値は「yes」(キープアライブを送信する) になっています。これはスクリプトにとって重要であり、ユーザにとっても役に立ちます。ただし、ルートが一時的にダウンした場合に接続が切断されることになるので、一部のユーザにとっては迷惑かもしれません。キープアライブを無効にするには、値を「no」に設定します。このキーワードにより、Windows TCP キープアライブ設定が有効になり、既定では 2 時間ごとにキープアライブメッセージが送信されます。TCP/IP キープアライブは、Windows レジストリには通常存在しない 2 つのオプションパラメータ (KeepAliveTime および KeepAliveInterval) を使用して構成できます。これらは、以下の場所の HKEY_LOCAL_MACHINE レジストリサブツリーに構成されます。

SYSTEM¥CurrentControlSet¥Services¥Tcpip¥Parameters

これらのパラメータの設定については、Microsoft Knowledge Base Article 120642 を参照してください。

KerberosAuthentication

Kerberos 認証をプロトコルバージョン 1 接続に使用するかどうかを指定します(プロトコルバージョン 2 で同等の設定は GssapiAuthentication になります)。このキーワードの引数は「yes」または「no」にする必要があります。

KerberosTgtPassing

Kerberos TGT をサーバに転送するかどうかを指定します。これは、Kerberos サーバが実際に AFS kaserver の場合にのみ機能します。この設定は、プロトコルバージョン 1 にのみ適用されます(プロトコルバージョン 2 で同等の設定は GssapiDelegateCredentials になります)。このキーワードの引数は「yes」または「no」にする必要があります。

KexAlgorithms

クライアントが対応する鍵交換アルゴリズムと優先順位を指定します。対応する値は、「diffie-hellman-group1-sha1」、「diffie-hellman-group-exchange-sha1」、および「diffie-hellman-group14-sha1」です。

既定値は、「diffie-hellman-group1-sha1,diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1」です。場合によっては、「diffie-hellman-group14-sha1」を他の 2 つよりも前に配置するために、鍵交換アルゴリズムの順番を変更する必要があります。hmac-sha512 MAC を使用する場合、または鍵交換中に次のエラーが表示される場合は、この操作が必要です。"fatal: dh_gen_key: group too small: 1024 (2*need 1024)"

メモ:Reflection Kerberos クライアントを使用する GSSAPI 認証が有効の場合、追加の鍵交換アルゴリズム (gss-group1-sha1 および gss-gex-sha1) が一覧に自動的に追加されます。

LocalForward

安全なチャネルを介して、リモートマシンの指定したホストとポートに転送するローカルマシンの TCP/IP ポートを指定します。複数の転送を指定できます。管理権限のあるユーザだけが権限ポートを転送できます。FTP の転送、リモートデスクトップの構成、接続後の実行ファイル (*.exe) の自動起動にオプションの引数を構成することもできます。このキーワードの構文は次のとおりです。

LocalForward localport host: hostport [FTP=0|1] [RDP=0|1] [" ExecutableFile " [args]]

オプションは次のとおりです。

localport

ローカルポート番号。

host: hostport

リモートホストおよびそのホストのポート(localhost を指定して、すでに Secure Shell 接続を確立した同じリモートホストの異なるポートにデータを転送できます)。IPv6 アドレスは、別の構文 host/port を使用して指定できます。

FTP

FTP ファイル転送をトンネル接続する場合、1 に設定します。

RDP

リモートデスクトップセッションをトンネル接続する場合、1 に設定します。

"ExecutableFile"

Secure Shell 接続確立後すぐに Reflection がアプリケーションを起動するように、実行ファイル (必要に応じて完全なパス情報を含む) を指定します。安全なトンネル経由でデータを転送するには、このアプリケーションは ローカルホストとの接続を行うように構成する必要があります (またはループバック IP アドレス、127.0.0.1)。このとき、指定した localport を使用します。

Logfile

デバッグに使用するログファイルを指定します。すべてのセッションの入力および出力は、このファイルに書き込まれます。次に示すように、このキーワードと -o コマンドラインユーティリティオプションを使用します。

-o Logfile=¥path¥logfile_name

メモ:パスまたはファイル名の一部に空白が含まれている場合、パスとファイル名を引用符で囲みます。

LogLevel

Secure Shell クライアントからのメッセージの記録時に使用する詳細レベルを指定します。指定可能な値は QUIET、FATAL、ERROR、INFO、VERBOSE、DEBUG、DEBUG1、DEBUG2 および DEBUG3 です。既定値は「INFO」です。DEBUG と DEBUG1 は同じです。DEBUG2 と DEBUG3 はそれぞれ、より高いレベルの詳細出力を指定します。

Macs

MAC（メッセージ認証コード）アルゴリズムを優先順に指定します。MAC アルゴリズムは、データ整合性保護のためにプロトコルバージョン 2 で使用されます。複数のアルゴリズムは、カンマで区切る必要があります。既定は、「hmac-sha256、hmac-sha2-256、hmac-sha1、hmac-md5、hmac-ripemd160、hmac-sha1-96、hmac-md5-96、hmac-sha512、hmac-sha2-512」です。接続が FIPS モードで実行するように設定されている場合、既定は「HMAC-SHA256、HMAC-SHA2-256、HMAC-SHA1、HMAC-SHA512、HMAC-sha2-512」です。

MatchHostName

ホスト証明書の確認時にホスト名の一致が必要かどうかを指定します。この設定が ［yes］ の場合 （既定値)、接続のために構成したホスト名が、証明書の［CommonName］フィールドまたは［SubjectAltName］フィールドに入力されているホスト名に一致していなければなりません。

Multihop

一連の SSH サーバによって安全な接続を確立するために使用可能なマルチホップ接続を構成します。これは、直接リモートサーバにアクセスすることはできないが、中間サーバを介してアクセスすることができるネットワーク構成で役に立ちます。

このキーワードの構文は次のとおりです。

Multihop localport host: hostport [″ SSH 構成セクション ″]

一連の各サーバに Multihop 行を新しく追加します。一覧にある各接続は、その上に表示されている接続によって設定されたトンネルを経由して送信されます。

下記の例では、サーバ C に構成された SSH 接続が最初に サーバ A に接続し、それから サーバ B、最後に サーバ C に接続します。

Host ServerC

Multihop 2022 ServerA:22

Multihop 3022 ServerB:22

オプションで SSH 構成セクションを指定して、チェーン内の任意のホストの Secure Shell 設定を構成できます。次に例を示します。

Multihop 4022 joe@ServerA:22 ″Multihop SchemeA″

Nodelay

この設定は、既定で、Windows の TCP ソケットで Nagle アルゴリズムを有効にする Microsoft によって行われる変更に対処し、Secure Shell 接続でのパフォーマンスに悪影響を与える可能性があります。Nodelay を yes（既定）に設定すると、このアルゴリズムが無効になり、ほとんどのシステムでパフォーマンスが向上します。

NoShell

NoShell を「Yes」に設定すると、クライアントは端末セッションを開かずにトンネルを作成します。このオプションは ConnectionReuse と組み合わせて使用し、その他の ssh 接続が再利用できるトンネルを作成できます。注意: このオプションは、接続がコマンドラインユーティリティで行われた場合に適用されます。このオプションは、ユーザインタフェースでの使用には対応していません。

NumberOfPasswordPrompts

失効するまでに試行できるパスワードメッセージの入力回数を指定します。このキーワードの引数は整数にする必要があります。デフォルトは 3 です。

PasswordAuthentication

パスワード認証を使用するかどうかを指定します。指定可能な値は「yes」および「no」です。既定値は「yes」です。

Port

リモートホストに接続するポート番号を指定します。デフォルト値は 22 です。

PreferredAuthentications

クライアントがプロトコル 2 認証方式を試行する順番を指定します。これは、[Reflection Secure Shell の設定] ダイアログボックスの [全般] タブの [ユーザ認証] 一覧に表示される方式の順番 (上から下) に対応します。この設定によって、クライアントはある方式 (キーボード対話型など) を別の方式 (パスワードなど) より優先させることができます。既定で、Reflection は「publickey,keyboardinteractive,password」の順番で認証を試行します。GSSAPI 認証が有効な場合、既定値は「gssapi-with-mic,external-keyex,gssapi,publickey,keyboard-interactive,password」に変更されます。

メモ:

config ファイルに PreferredAuthentications が含まれている場合、指定する一覧に試行したい認証方式をすべて含める必要があります。PreferredAuthentications が存在するにもかかわらず特定の認証方式を指定しない場合、その認証方式を有効にするキーワードが正しく構成されていても、Reflection ではその認証方式を使用しません。

PreferredAuthentications 一覧に認証方式を含めることによって、その方式を使用する認証が有効になるわけではありません。既定で使用されない認証を有効にするには、その認証方式のキーワードも正しく構成する必要があります (例えば、GSSAPI 認証を有効にするには、GssapiAuthentication を yes に設定する必要があります)。

PreserveTimestamps

サーバ間でファイルを転送する時に、属性とタイムスタンプを変更するかどうかを指定します。このキーワードが「no」(既定値) の場合、タイムスタンプと属性が変更されます。「yes」の場合、ファイルの元のタイムスタンプと属性が保持されます。

Protocol

Secure Shell クライアントが対応するプロトコルバージョンを優先順に指定します。指定可能な値は「1」および「2」です。複数の値は、カンマで区切る必要があります。既定値は「2,1」で、Reflection はバージョン 2 を試行し、バージョン 2 が使用できない場合にバージョン 1 にフォールバックします。

Proxy

Secure Shell 接続に使用するプロキシの種類を指定します。対応する値は、「SOCKS」と「HTTP」です。
メモ:この設定を使用する各 Host セクションで、プロキシの使用が有効になります。プロキシサーバアドレスは、ユーザごとに Windows レジストリに保存されます。

PubkeyAlgorithms

クライアントがサーバーに提案するキーアルゴリズムを優先順位順に指定します。サーバーが一つのアルゴリズムにのみ設定されている場合は、そのオプションが提案するキーワードにのみ設定することができます。
利用可能な値：″x509v3-rsa2048-sha256, x509v3-sign-rsa, x509v3-sign-dss,
ssh-rsa-sha2-256@attachmate.com, ssh-rsa,ssh-dss″。

PubkeyAuthentication

公開鍵認証を試行するかどうかを指定します。このオプションは、プロトコルバージョン 2 にのみ適用されます。指定可能な値は「yes」および「no」です。既定値は「yes」です。

RemoteCommand

リモートサーバで実行する 1 つまたは複数のコマンドを指定します。UNIX サーバへの接続時には、セミコロン（;）を使用して複数のコマンドを区切ります。Windows サーバへの接続時には、アンパサンド記号(&) を使用してコマンドを区切ります。接続の確立後、サーバは指定したコマンドを実行（または実行を試行）し、セッションが終了します。サーバは、クライアントから受信したコマンドの実行を許可するよう構成されている必要があります。
コマンドは、使用するサーバに合った構文を使用して指定する必要があります。例えば、以下が同じになります。
UNIX の場合: ls ; ls -l
Windows の場合: dir/w & dir

RemoteForward

安全なチャネルを介して、ローカルマシンの指定したホストとポートに転送するリモートマシンの TCP/IP ポートを指定します。最初の引数はポート番号にする必要があり、2 つ目は次のようにします Host:port を使用します。IPv6 アドレスは、別の構文 Host/port を使用します。複数転送を指定できます。管理権限のあるユーザだけが権限ポートを転送できます。

RSAAuthentication

RSA 認証を試行するかどうかを指定します。このオプションは、プロトコルバージョン 1 にのみ適用されます。RSA 認証は、識別情報ファイルが存在する場合のみ試行されます。指定可能な値は「yes」および「no」です。既定値は「yes」です。

SendEnv

シェルまたはコマンドの実行前に、サーバに設定する環境変数を指定します。値の形式は VAR val にする必要があります。サーバは指定した変数に対応し、これらの環境変数を受け付けるように構成する必要があります。

ServerAlive

ServerAliveInterval で指定した間隔で、SSH サーバにサーバアライブメッセージを送信するかどうかを指定します。Secure Shell の ServerAlive 設定では、SSH プロトコルメッセージを指定した間隔でサーバに送信し、サーバが機能していることを確認します。この設定が有効になっていないと、サーバが停止するかネットワーク接続が失われた場合に SSH 接続が終了されません。この設定は、TCP セッションのみを転送する接続がサーバによって時間切れになるのを防ぐためにも使用できます。サーバは、SSH トラフィックが存在しないことの検出を理由にこれらの接続を時間切れにする場合があるからです。指定可能な値は「yes」および「no」です。既定値は「no」です。

メモ:Secure Shell の ServerAlive 設定は、すべての TCP/IP 接続がファイアウォールによって時間切れになるのを防ぐために Windows レジストリに設定可能な TCP キープアライブ設定 (KeepAlive) とは関係ありません。TCP/IP キープアライブの動作を変更するには、Windows レジストリを編集する必要があります。

ServerAliveInterval

ServerAlive = 'yes' の場合に使用する間隔 (秒単位) を指定します。1 以上の整数値を使用します。デフォルトは 30 です。

ServerKeyFormat

[ユーザーキー] タブ上のアップロード機能を使用してホストにキーをアップロードする際に使用するキー形式を指定します。ユーティリティは、使用するキー形式を自動的に決定します。その形式がご使用のサーバーに対して間違っている場合は、この設定を変更します。指定可能な値は「OpenSSH」および「SECSH」です。

ServerStyle

[ユーザーキー] タブ上のアップロード機能を使用してホストにキーをアップロードする際に使用するホスト公開キー構成設定を指定します。ユーティリティは、ホストスタイルを自動的に決定します。その形式がご使用のサーバーに対して間違っている場合は、この設定を変更します。指定可能な値は「UNIX」および「VMS」です。

SftpBufferLen

SFTP 転送時に、各パケットで要求されるバイト数を指定します。デフォルトは 32768 です。この値を調整すると、転送速度を上げることができます。最適な値は、使用しているネットワークおよびサーバ設定によって異なります。この値を変更すると、データの転送をキャンセルしてから実際に転送が停止するまでの時間にも影響を与えることがあります。

SftpMaxRequests

クライアントが SFTP 転送時に許可する、未処理データ要求の最大数を指定します。デフォルトは 10 です。この値を調整すると、転送速度を上げることができます。最適な値は、使用しているネットワークおよびサーバ設定によって異なります。この値を変更すると、データの転送をキャンセルしてから実際に転送が停止するまでの時間にも影響を与えることがあります。

SftpVersion

クライアントが SFTP 接続に使用するバージョンを指定します。有効な値は 3 と 4 です。この設定が 4 (既定値) の場合、接続では (サーバが対応していれば) SFTP バージョン 4 が使用されます。また、サーバがバージョン 4 に対応していない場合はバージョン 3 が使用されます。この設定が 3 の場合、クライアントは常に SFTP バージョン 3 を使用します。

StrictHostKeyChecking

引数は「yes」、「no」または「ask」にする必要があります。既定値は「ask」です。このオプションを「yes」に設定した場合、 Secure Shell クライアントはホスト鍵を known_hosts ファイル (ユーザの .ssh フォルダにある) に自動的に追加せず、ホスト鍵が変更されたホストへの接続を拒否します。このオプションを使用する場合、ユーザは新しいホストを手動で追加する必要があります。このフラグを「no」に設定した場合、Reflection は確認ダイアログボックスを表示せずにホストに接続し、ホスト鍵を信頼する鍵の一覧に追加しません。このフラグを「ask」に設定した場合、ユーザが該当する鍵であることを確認してから、新しいホスト鍵がユーザ既知のホストファイルに追加されます。すべての場合、既知のホストのホスト鍵は自動的に検証されます。

メモ:この設定は、ホストが x509 証明書を使用する認証を構成されている場合は影響しません。ホストがホスト認証のために証明書を提示し、トラストアンカとして構成された必要な CA 証明書がない場合、接続に失敗します。

TryEmptyPassword

このフラグを「yes」に設定した場合、クライアントは空のパスワードの入力を試行してパスワード認証を開始します。これは、ほとんどのシステムでログイン試行と見なされます。

User

ログインするユーザを指定します。これは、異なるマシンで異なるユーザ名を使用する場合に便利です。

UseOCSP

クライアントが OCSP (Online Certificate Status Protocol) を使用してホスト証明書を検証するかどうかを指定します。指定可能な値は「yes」および「no」です。既定値は「no」です。

UserKeyCertLast

クライアントが公開鍵認証中に証明書の署名を処理する方法を指定します。この設定が「yes」(既定) の場合、クライアントは最初に標準の ssh 鍵署名 (ssh-rsa または ssh-dss) を使用して証明書を送信します。それに失敗すると、クライアントは再び証明書署名 (x509-sign-rsa または x509-sign-dss) の使用を試みます。場合によっては、この 2 番目の試みが行われず、認証が失敗します。この設定が「no」の場合、クライアントは最初に証明書署名を試み、次に ssh 鍵署名を試みます。

UserKnownHostsFile

known_hosts ファイル (ユーザの .ssh フォルダにある) の代わりに使用する、ユーザホスト鍵データベース用のファイルを指定します。ファイルまたはパスが空白を含む場合、引用符を使用します。

x509dsasigtype

DSA 秘密鍵の所有を提供する過程でクライアントが使用するハッシュアルゴリズムを指定します。指定可能な値は「sha1raw」(既定値) と「sha1asn1」です。

x509rsasigtype

RSA 秘密鍵の所有を提供する過程でクライアントが使用するハッシュアルゴリズムを指定します。指定可能な値は「md5」、「sha1」(既定値)、および「sha256md5」です。

X11Display

X11 転送が有効な場合、X11 プロトコルが通信する PC のローカルループバックインタフェース上のポートを転送するかどうかを決定します。

メモ:Reflection X (バージョン 12.x、13.x、または 14.x) を使用している場合は、このキーワードを構成する必要はありません。Reflection X サーバと Secure Shell クライアントは自動的に同期して、X サーバ表示設定 ([設定] - [表示] - X 表示番号) に基づく適切なポートを使用します。この場合、X11Display キーワードは無視されます。別の PC X サーバを使用する場合は、このキーワードを使用して、PC X サーバに定義されている適切なリスニングポートを指定します。

デフォルト値は「0」です。これにより、ポート 6000 への転送が構成されます。このポートは、X11 プロトコル規約で定義された既定のリスニングポートです。指定する表示値は実際のリスニングポートを決定するために、6000 に追加されます。例えば、X11Display を 20 に設定した場合、Secure Shell クライアントにとって、PC-X サーバがポート 6020 で待ち受けしていることを意味します。

以上