Bound - Packet Relay Agent

この文書はBoundの説明書です。

目次

1.概要

Bound はネットワークパケットを中継するソフトウェアです。ネットワーク上に複数の Bound を配置し、目的地までパケットを運ぶことができます。

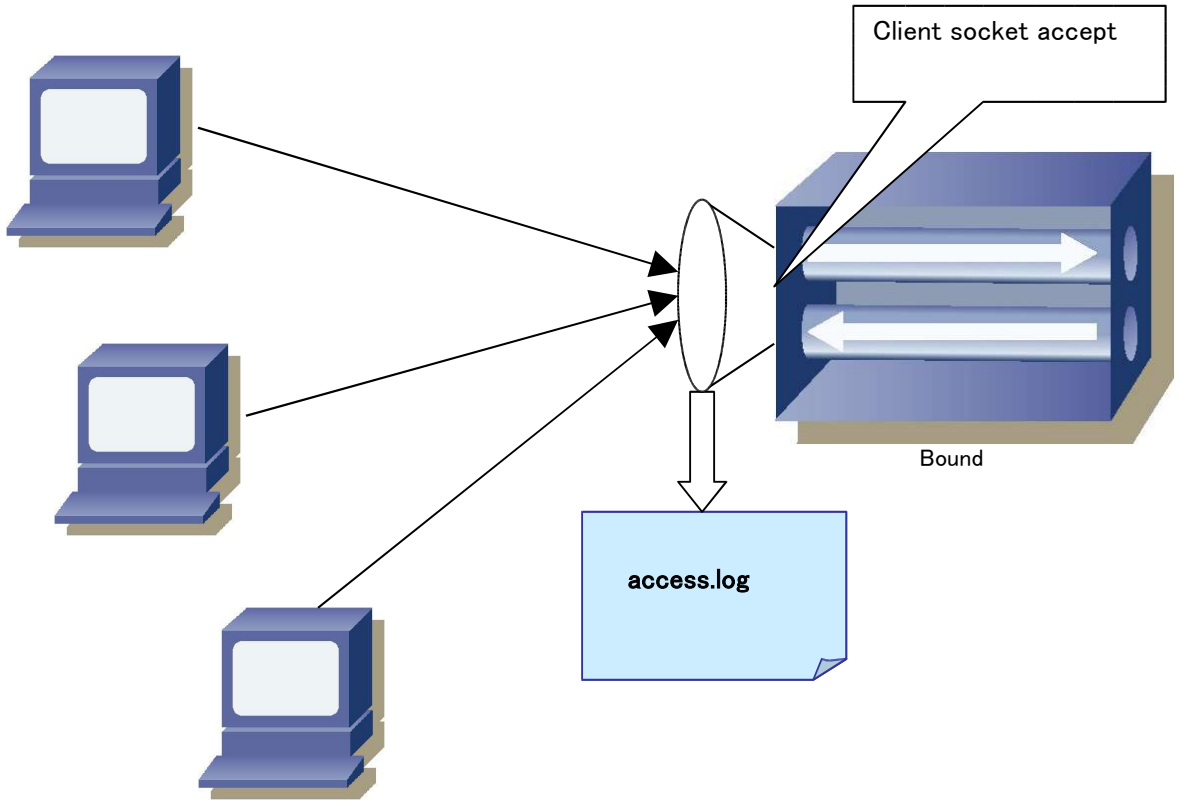
2. 機能

Bound には基本となるパケット中継機能に加えて、以下の機能があります。

- アクセスログの取得
- 送受信パケットのダンプ
- 任意の HTTP ヘッダの追加と削除
- プロキシ経由転送
- GZIP による圧縮転送
- SSL 通信化

2.1 アクセスログの取得

Bound を起動すると、サーバソケットによる待ち受けのスレッドが起動します。この待ち受けスレッドがクライアントからのソケット接続を accept するたびにアクセスログを取ります。アクセスログのデフォルトのファイル名は access.log です。

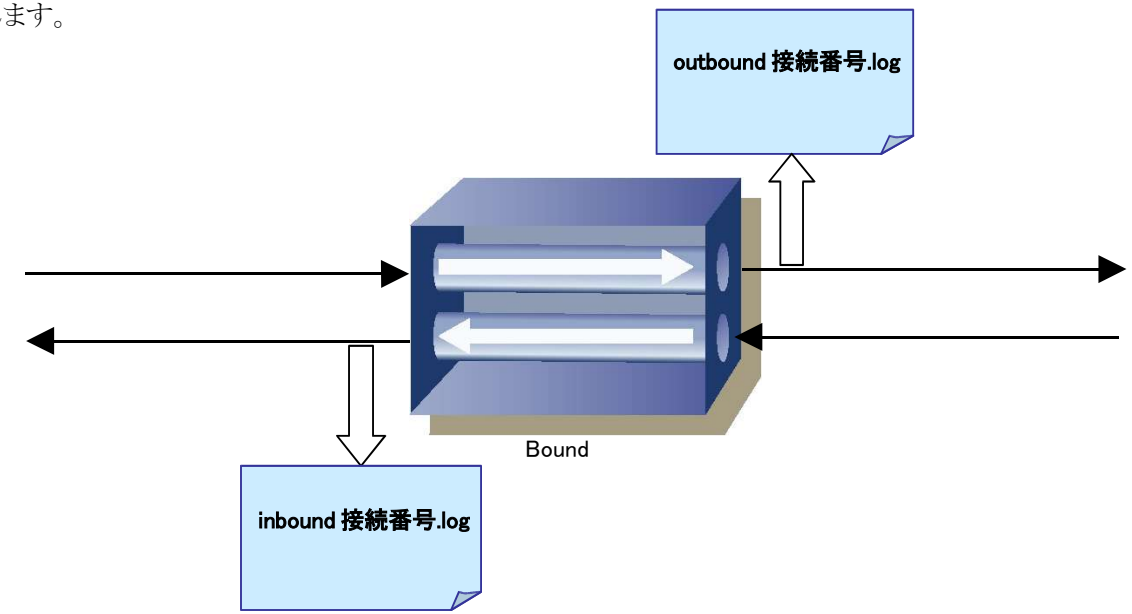


2.2 パケットダンプ

パケットの内容を送信時と受信時別々にファイルへ取得できます。ダンプの整形形式は次の3つから選択可能です。

- **Byte 形式** ― パケットの内容をそのまま書き出します。
- **HEX 形式** ― HEX ダンプ形式に変換して書き出します。
- **Base64 形式** ― Base64 形式に変換して書き出します。

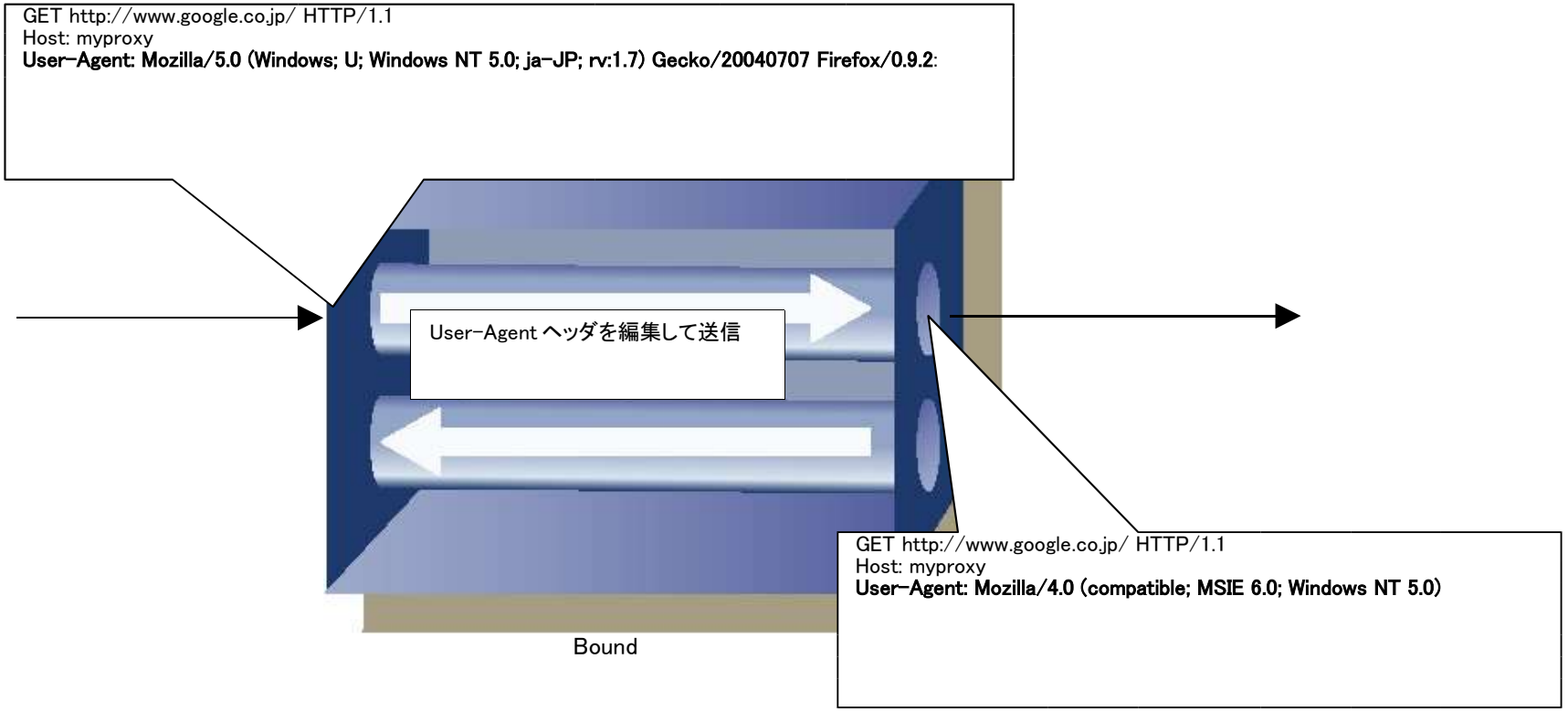
保存するファイル名は、送信パケットが **outbound 接続番号**.log、受信パケットが **inbound 接続番号**.log となります。接続番号はソケット接続があるたびにインクリメントされます。



2.3 任意の HTTP ヘッダの追加と削除

HTTP パケットをリレーする際、任意の HTTP ヘッダを追加、削除することができます。

下記の図は、HTTP リクエストの既存の User-Agent ヘッダを一旦削除し、新しい User-Agent ヘッダを付加している例です。同じようにレスポンス送信時にも HTTP ヘッダを追加、削除することができます。

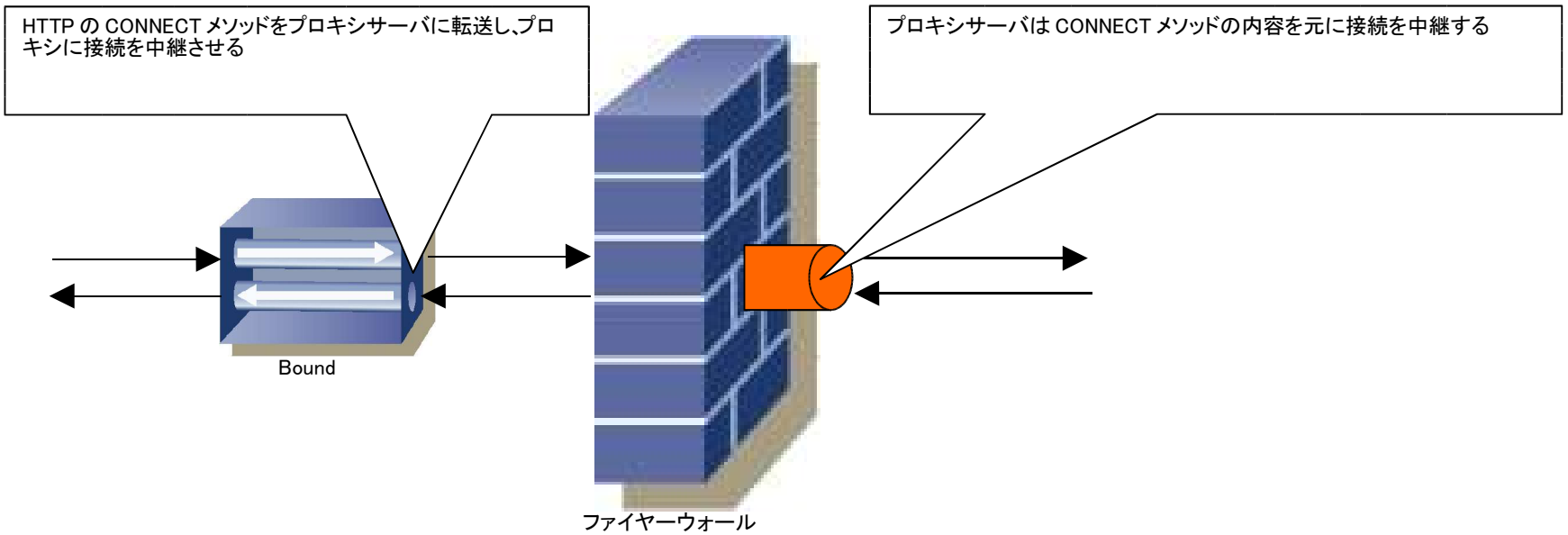


なお、追加する HTTP のヘッダには、以下のキーワードを使用することができます。

%t ー タイムスタンプ(ms)
%d ー 時刻(GMT)
%s ー ソケット情報
%h ー 接続先ホストアドレス
%p ー 接続先ポート
%H ー ローカルホストアドレス
%P ー ローカルポート

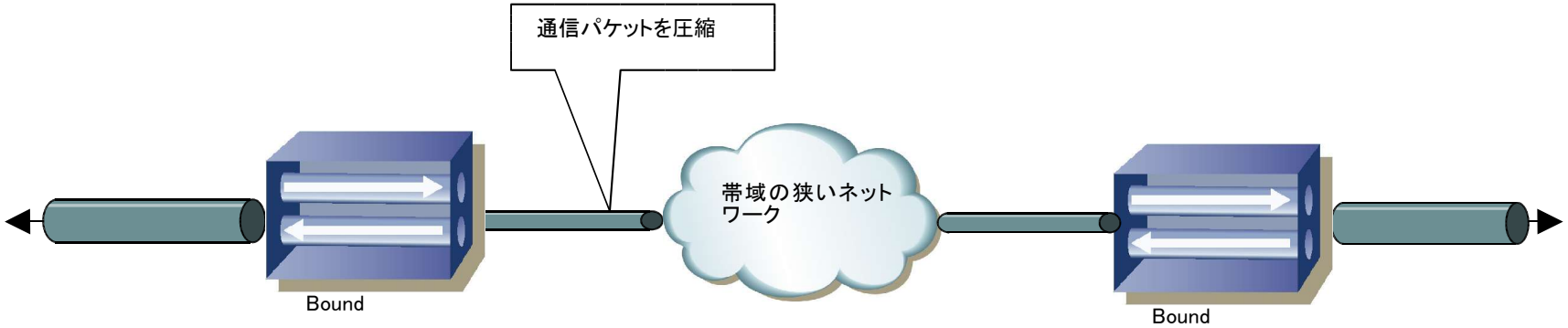
2.4 プロキシ経由転送

プロキシサーバが HTTP の CONNECT メソッドを受け付ける場合、接続先の中継役としてプロキシサーバを利用することができます。これにより、ファイヤーウォールに新たな穴を開けることなく任意のプロトコルを通すことができます。なお、プロキシサーバの基本認証にも対応します。



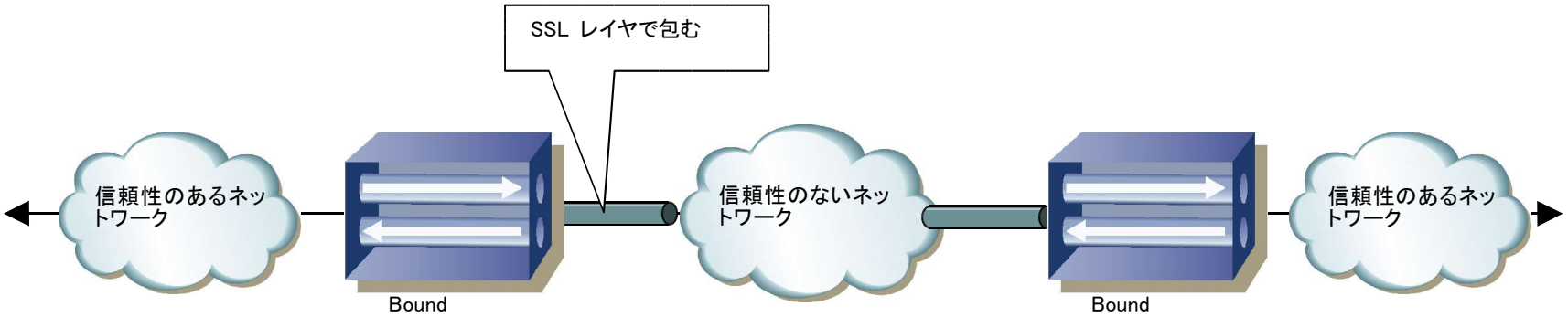
2.5 GZIP による圧縮転送

帯域の狭いネットワークの両側に Bound を配置し、Bound 同士の通信パケットを GZIP 圧縮することができます。



2.6 SSL 通信化

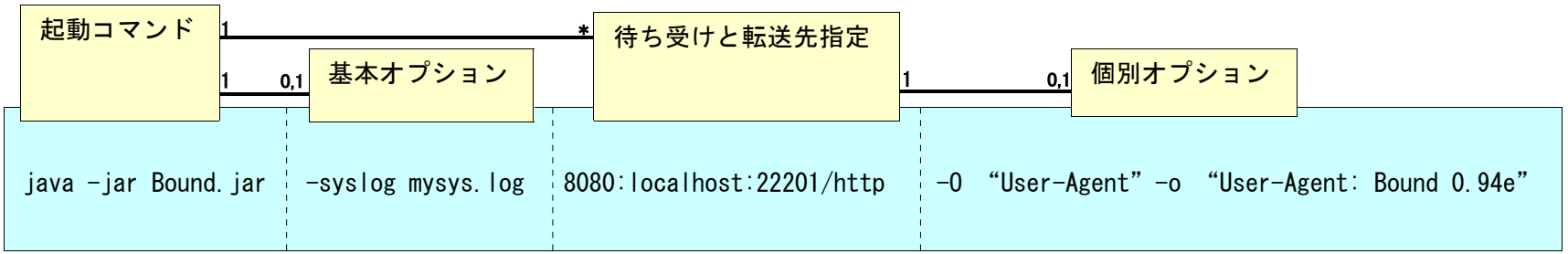
インターネットなどの信頼性のないネットワークの両側に Bound を配置し、Bound 同士の通信を SSL 化することにより、通信の機密性を上げることができます。



3 使い方

3.1 コマンドラインの指定方法

Bound を起動するコマンドラインの指定は、[基本オプション]、[待ち受けと転送先指定]、[個別オプション]に分けられます。このうち、必ず指定しなければならないのは、[待ち受けと転送先指定]です。



・基本オプション

Bound の動作全体に影響するオプションです。[待ち受けと転送先指定] の前に指定する必要があります。

指定できるオプションについては、5.オプション一覧を参照してください。

・待ち受けと転送先指定

Bound が待ち受けるポートとパケットの転送先を指定します。書式は以下のとおりです。

待ち受けポート/プロトコル:**転送先ホスト**:**転送先ポート**/プロトコル　　(/プロトコル **は省略可**)

[待ち受けと転送先指定] は複数指定することができます。サポートするプロトコルについては 3.2 サポートするプロトコルを参照してください。

・個別オプション

[待ち受けと転送先指定] に対して効力を持つオプションです。[待ち受けと転送先指定] の後に指定する必要があります。

指定できるオプションについては、5.オプション一覧を参照してください。

3.2 サポートするプロトコル

プロトコルは、[待ち受けと転送先指定] の中の 2 ヶ所で指定することができますが、それぞれを以下の例のように、[受信方向]、[送信方向] とします。

8080/受信方向:localhost:22201/送信方向

サポートするプロトコルは以下の表のとおりです。

名前	指定場所	機能
tcp	**送信方向**	3.1 の[待ち受けと転送先指定]が省略された場合のデフォルトのプロトコルです。任意の TCP パケットを中継します。
	受信方向	同上。
http	**送信方向**	HTTP リクエストに任意の HTTP ヘッダの追加と削除を行う。
	受信方向	HTTP レスポンスに任意の HTTP ヘッダの追加と削除を行う。
dump	**送信方向**	送信パケットをファイルにダンプします。
	受信方向	受信パケットをファイルにダンプします。

ssl	**送信方向**	送信先とSSL 通信を行う。
	受信方向	受信元とSSL 通信を行う。
gzip	**送信方向**	送信パケットをGZIP 圧縮する。
	受信方向	受信パケットをGZIP 展開する。

3.3 コマンド例

Bound を起動するコマンド例をいくつか簡単に紹介します。

<基本>

ポート 2020 で待ち受け、hostA:3030 へ転送。

```
java -jar Bound.jar 2020:hostA:3030
```

<プロキシ経由で転送>

ポート 2020 で待ち受け、プロキシ proxy:8080 を経由して hostA:3030 へ転送。プロキシの基本認証は myid:mypasswd。

```
java -jar Bound.jar 2020:hostA:3030 -P proxy:8080 -A myid:mypasswd
```

<SSL トンネル>

client 側:ポート 23 で待ち受け、server:3030 へ SSL 転送。

```
java -jar Bound.jar 23:server:3030/ssl
```

server 側:ポート 3030 で SSL で待ち受け、localhost:23 へ 転送。

```
java -jar Bound.jar 3030/ssl:localhost:23
```

<複数ポート待ち受け>

ポート 2020 で待ち受け、hostA:3030 へ転送。且つ、ポート 4040 で待ち受け、 hostB:5050 へ転送。

```
java -jar Bound.jar 2020:hostA:3030 4040:hostB:5050
```

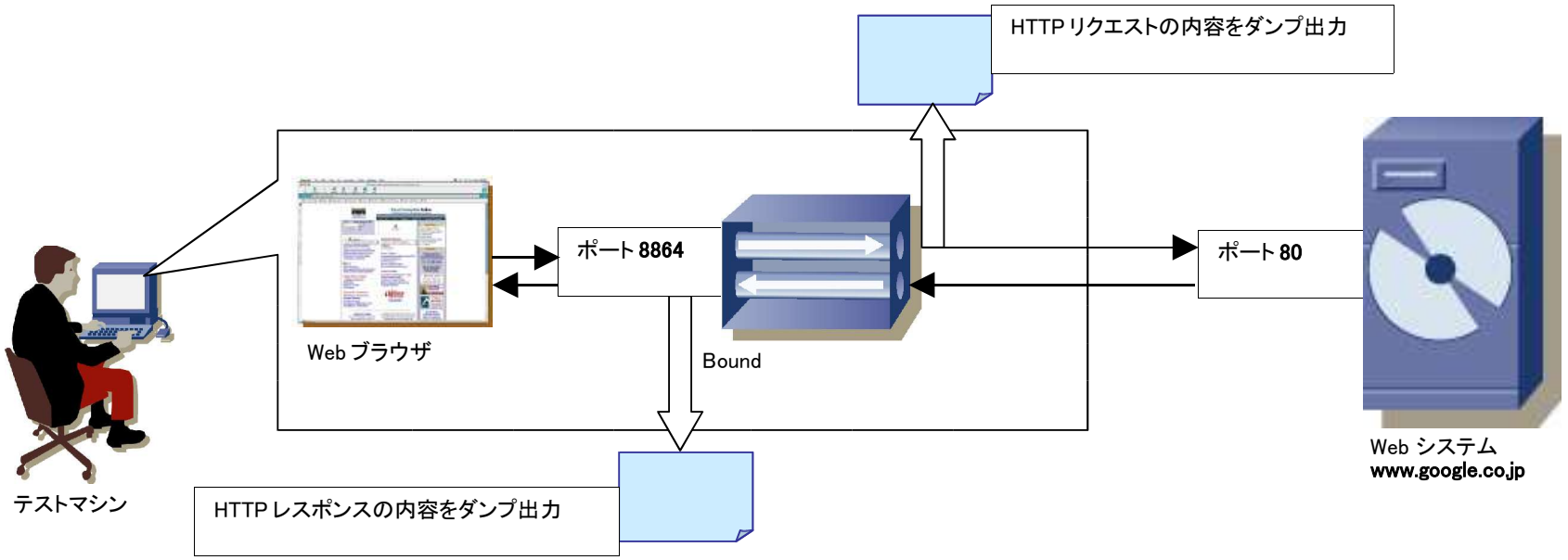
4.使用例

以下の表のユースケースを例にして使い方を説明します。

#	使用する機能	ユースケース
Example 1	パケットダンプ	Web システムのテストで、Web ブラウザと Web システム間の HTTP リクエストと HTTP レスポンスをダンプする。
Example 2	HTTP ヘッダの追加	プロキシ認証に対応していない HTTP クライアントで、プロキシサーバを介して外部のサーバにアクセスする。
Example 3	プロキシ経由転送 SSL 通信化	telnet を遮断するファイヤーウォールの内側から外部のサーバに telnet ログインし、サーバのメンテナンスを実施する。

4.1 Example 1 – パケットダンプ

この例では、Bound を Web ブラウザと Web システム間のプロキシとして配置します。Bound は HTTP リクエストと HTTP レスポンスを中継する際、メッセージをダンプします。



Bound 起動コマンド例:

```
java －jar Bound.jar 8864/dump:www.google.co.jp:80/dump －dump BYTE
```

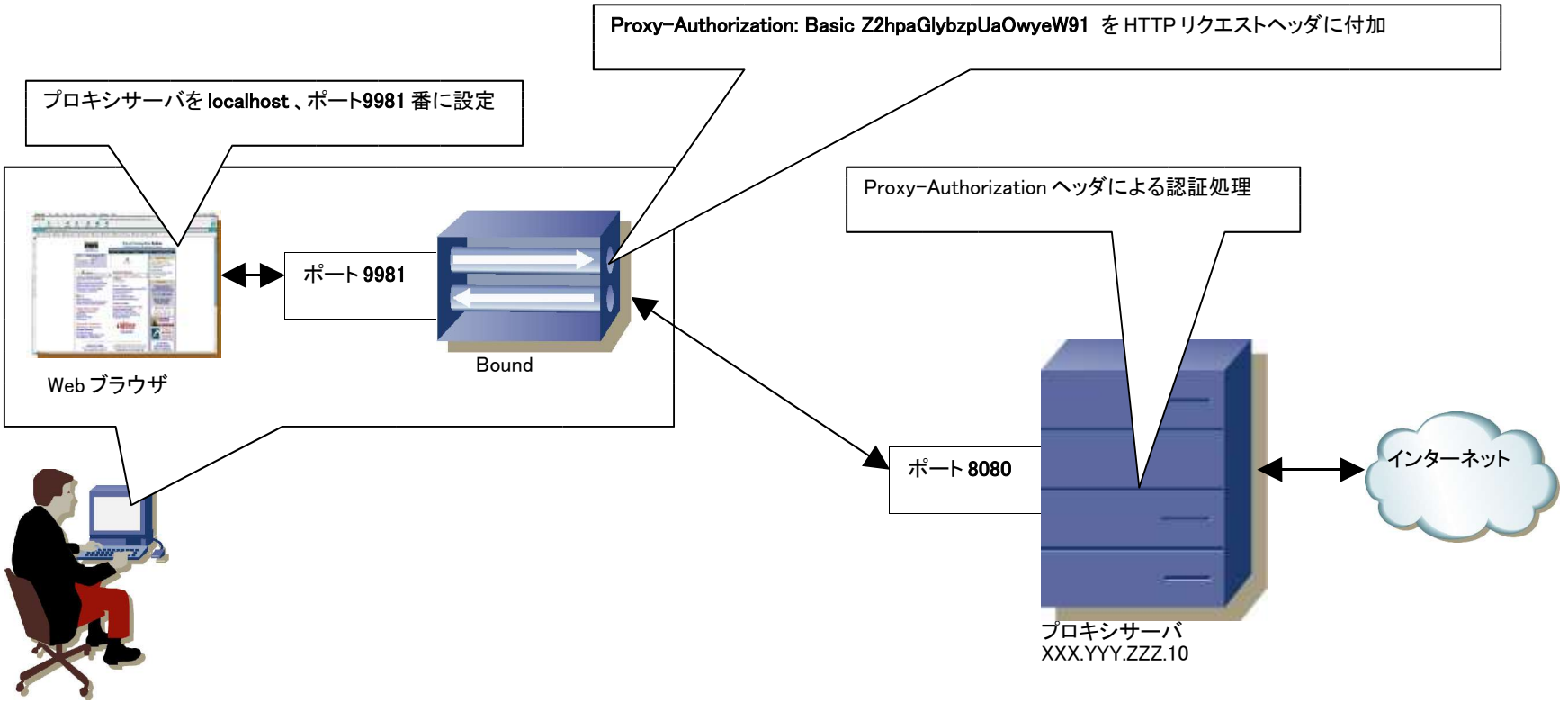
テストマシン上で上記のコマンドで Bound を起動すると、ポート 8864 番で待ち受けし、転送先が www.google.co.jp に設定されます。次にブラウザを立ち上げ、HTTP プロキシの設定(注)を localhost、ポート 8864 に設定します。この状態でブラウザから Google の任意の URL(例えば http://www.google.co.jp/index.html)にアクセスすると、Bound はブラウザと www.google.co.jp のポート 80 番との HTTP リクエストとレスポンスを中継します。ポート指定の後の / (スラッシュ) に続く指定は プロトコル指定を表します。ここで指定している dump プロトコルは、パケット中継しながらデータの内容をファイルダンプしていくというプロトコルです。受信方向と送信方向の両方に dump プロトコルを指定することで、リクエストとレスポンスの両方をダンプする指定となっています。

なお、ダンプの出力形式は -dump オプションで指定できます。ここでは BYTE 形式を指定しているので、リクエストとレスポンスの内容は整形されずにそのままダンプされます。

(注) IE の場合、ツール->インターネットオプション->「接続」タブ->「LAN の設定」ボタンのダイアログで設定する。

4.2 Example 2 - HTTP ヘッダの追加

内部ネットワークとインターネットとの間にプロキシサーバを配置していて、HTTP でインターネットにアクセスするにはプロキシサーバの認証が必要な状況を想定します。この例では、プロキシ認証をサポートしていない HTTP クライアントでインターネット接続を実現します。HTTP クライアントとして Web ブラウザを例にとって説明します。



Bound 起動コマンド例:

```
java －jar Bound.jar 9981:XXX.YYYY.ZZZZ.10:8080/http －o “Proxy-Authorization: Basic Z2hpaGlybzpUaOwyeW91”
```

上記のコマンドで Bound を立ち上げ、プロキシサーバ(IP アドレス XXX.YYY.ZZZ.10、ポート番号 8080)に転送先を設定します。-o オプション は送信方向のプロトコルが http の場合に有効となるオプションです。ここでは “Proxy-Authorization: Basic Z2hpaGlybzpUaOwyeW91” をブラウザからの HTTP リクエストヘッダに付加してプロキシサーバに送信します。Proxy-Authorization: Basic の後の文字列は、ユーザ名とパスワードをコロンで繋げたもの(username:password)を Base64 形式でエンコードした文字列です。プロキシは Proxy-Authorization ヘッダの内容で認証を実施し、認証 OK ならばインターネットへの接続を許可します。

次にブラウザの設定で、プロキシサーバの HTTP の設定を、ホスト名 localhost、ポート番号 9981 に設定した後、任意のインターネットの Web ページにアクセスします。プロキシの基本認証ダイアログが出ないでアクセスできるはずです。

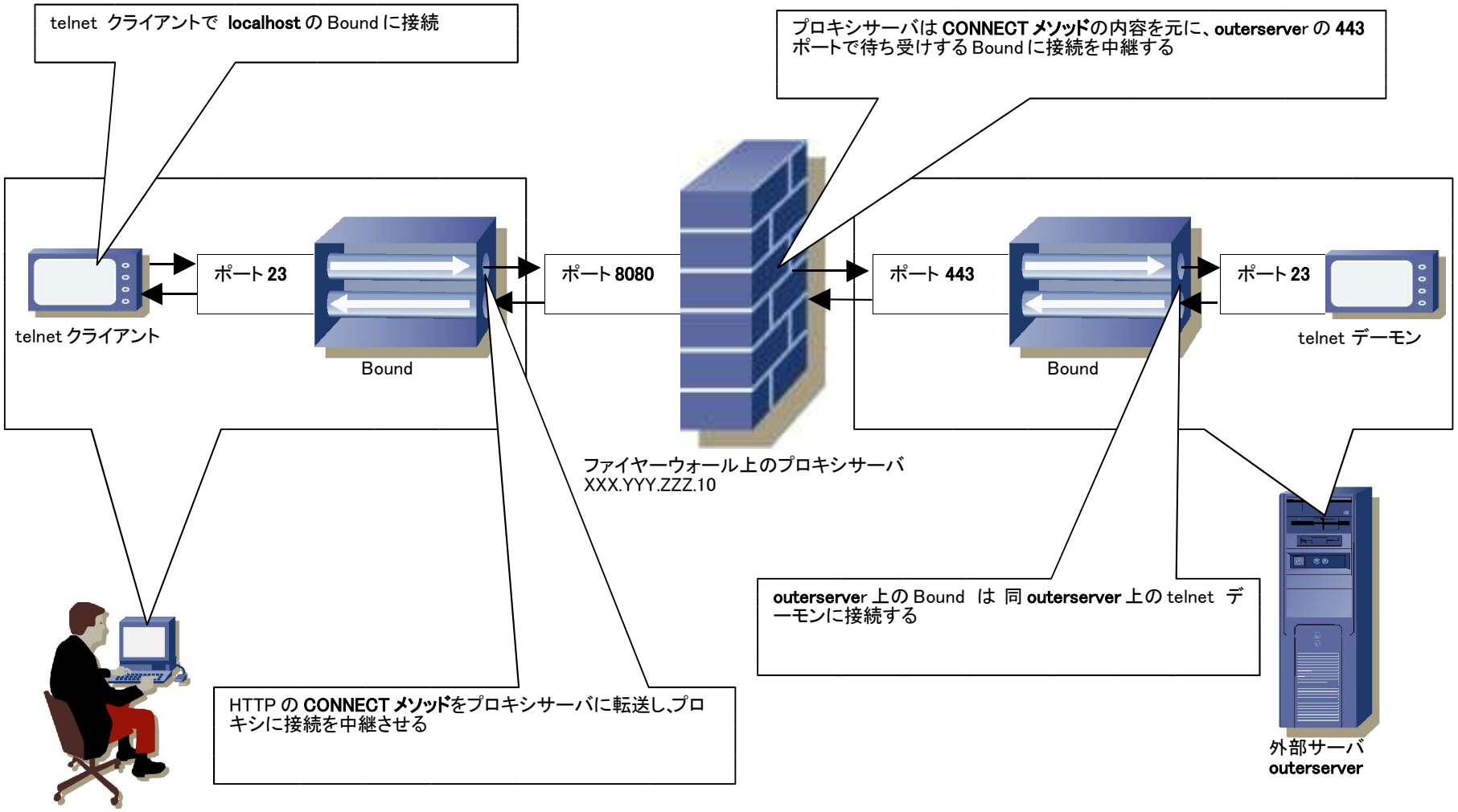
なお、Bound には文字列を Base64 形式に変換するための簡易ツールがあります。以下のコマンドの例では文字列 username:password を Base64 形式に変換しています。

```
java －cp Bound.jar cm.utils.EncodeOutputStream BASE64 username:password
```

4.3 Example 3 – プロキシ経由転送と SSL 通信化

内部ネットワークから外部ネットワークにアクセスする場合に、ファイヤーウォール上のプロキシサーバで HTTP による接続のみを許している状況を想定します。この例では内部ネットワーク側の Bound が HTTP の CONNECT メソッドをプロキシサーバに向けて送信し、外部のサーバとの接続をプロキシサーバに中継させることにより、ファイヤーウォールをトンネリングして外部ネットワーク側の Bound に接続します。つまり、内部ネットワーク側と外部ネットワーク側それぞれに TELNET メッセージをやり取りする Bound が存在し、その間の中継役としてプロキシサーバを利用します。

なお、CONNECT メソッドを完全に拒否する設定のプロキシサーバの場合はこのシナリオは成り立ちません。


telnet クライアントで localhost の Bound に接続
プロキシサーバは CONNECT メソッドの内容を元に、outerserver の 443 ポートで待ち受けする Bound に接続を中継する
telnet クライアント
ポート 23
Bound
ポート 8080
ファイヤーウォール上のプロキシサーバ
XXX.YYY.ZZZ.10
ポート 443
Bound
ポート 23
telnet デーモン
outerserver 上の Bound は 同 outerserver 上の telnet デーモンに接続する
外部サーバ
outerserver
HTTP の CONNECT メソッドをプロキシサーバに転送し、プロキシに接続を中継させる

内部ネットワーク側 Bound 起動コマンド例:

```
java －jar Bound.jar 23:outerserver:443/ssl －P XXX.YYY.ZZZ.10:8080 －A userid:password -cs client.jks -cspass cspasswd
```

上記のコマンドで Bound は telnet の標準ポート 23 で待ち受けし、-P オプションで指定したプロキシサーバに対して outerserver の 443 ポートに接続するように要求します。また、送信方向のプロトコルを指定を ssl とし、SSL 転送としています。また、プロキシ認証を通すため、-A オプションにより、プロキシ認証のユーザ ID とパスワードを :(コロン) で繋げた文字列 userid:password を指定しています(プロキシが認証を要求しないならば -A オプションは必要ありません)。

-P オプションにより実際にプロキシサーバに送信されるメッセージは、HTTP の CONNECT メソッド ”CONNECT outerserver:443 HTTP/1.0” となります。プロキシサーバは、この情報を元に外部ネットワーク側の outerserver の 443 ポートで待ち受ける Bound に接続します。なお、443 ポートは通常 SSL に使用されるポートで、多くのプロキシは 443 ポートへの CONNECT メソッドを許可します。

外部ネットワーク側 Bound 起動コマンド例:

```
java －jar Bound.jar 443/ssl:outerserver:23 -ss server.jks -sspass sspasswd
```

上記のコマンドでは Bound は 443 ポートで 受信方向のプロトコル指定を ssl とし、SSL 待ち受けとなります。プロキシサーバからの接続が accept されると、転送されて来た TELNET メッセージを telnet デーモン(outerserver、ポート 23)に転送します。

(注) TELNET プロトコルは通信データをクリアテキストのまま送受信するので、この例では Bound 間の通信データを SSL により暗号化していますが、そのような必要がなければ省略できます。なお、-cs、-cspass、-ss、-sspass オプションは SSL プロトコル用のオプションです。詳しくは 5.2.4 ssl プロトコルオプションを参照ください。

5. オプション一覧

5.1 基本オプション

以下の表に Bound の基本動作に影響するオプション一覧を示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-aclog [--access-log]	有	No	アクセスログを書き込むファイルを指定する。	access.log
-syslog [--system-log]	有	No	システムログを書き込むファイルを指定する。	system.log
-verbose [--verbose-mode]	無	No	過度な情報を表示する。	-
-h [--help]	無	No	オプションに関するヘルプを表示する。	-
-e [--example]	無	No	コマンド例を表示する。	-
-v [--version]	無	No	バージョンを表示する。	-
-aport [--admin-port]	有	No	管理コマンドを受けつけるポートを指定する。	-

5.2 個別オプション

個別オプションで有効なオプション一覧を示します。

5.2.1 プロトコルに依存しない個別オプション

以下の表にプロトコルに依存しない個別オプションを示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-P [--proxy]	有	No	プロキシサーバを指定する。	-
-A [--proxy-authorization]	有	No	プロキシ基本認証に使用するユーザ ID とパスワードを:(コロン)で繋げた文字列を指定する。	-
-Q [--max-queue-length]	有	No	接続リクエストを格納する待ち行列の長さを指定する。待ち行列に入ったリクエストは、リクエストを処理するワーカースレッドの内の1つが取り出して処理する。特に-T オプションで、ワーカースレッド数の上限を指定している場合は、処理待ちリクエスト数の許容量として、待ち行列の長さを調整する必要がある。	50
-T [--max-threads]	有	No	接続リクエストを処理するワーカースレッドのプールサイズの最大値を指定する。ワーカースレッドのプールはインバウンド側とアウトバウンド側で2つ用意され、それぞれのプールから選ばれた1対のスレッドが、1つのリクエストを処理する。-T オプションの指定は最大同時接続数を指定することと同等である。	-1(無制限)

-iT [--initial-threads]	有	No	接続リクエストを処理するワーカースレッドのプールサイズの初期値を指定する。	10
-gT [--gain-threads]	有	No	接続リクエストを処理するワーカースレッドの増加量を指定する。Bound 動作中のワーカースレッドは同時接続リクエストの増減に応じて以下のように自動調整される。 a) 同時接続リクエストが増加し、現在のプールサイズの 90%以上のスレッドが処理中になった場合は、-gT オプションで指定した数のスレッドを新た生成して、プールサイズを増やす。 b) a)により、プールサイズが初期値より一時増加したが、その後同時接続リクエストが減少し、処理中のスレッド数が現在のスレッド数の 30%以下になった場合は、プールサイズを-gT オプションで指定した数だけ減少させる。	-iT オプションの指定値の半分(端数切り捨て)
-a [--allow-addresses]	有	No	接続を許すアドレスのリストを指定する。なお、ワイルドカード(*)を使用することができる。 例 1: -a 192.168.1.* 例 2: -a 192.168.2.10 192.168.2.11 192.168.2.12 指定以外のアドレスからの接続は全て拒否されるようになる。	全ての接続を許す

5.2.2 http プロトコルオプション

以下の表に http プロトコルを指定した場合に有効となる個別オプションを示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-o [--http-outbound]	有	(注 1)	送信方向のデータへ添付する HTTP ヘッダ(ヘッダ名＋値)のリストを指定する。	-
-O [--delete-http-outbound]	有	(注 1)	送信方向のデータから削除する HTTP ヘッダ名のリストを指定する。	-
-i [--http-inbound]	有	(注 2)	受信方向のデータへ添付する HTTP ヘッダ(ヘッダ名＋値)のリストを指定する。	-
-I [--delete-http-inbound]	有	(注 2)	受信方向のデータから削除する HTTP ヘッダ名のリストを指定する。	-

(注 1) 送信方向のプロトコルに http を指定して場合に、-o または -O のどちらか一方、または両方を指定する必要がある。

(注 1) 受信方向のプロトコルに http を指定して場合に、-i または -I のどちらか一方、または両方を指定する必要がある。

5.2.3 dump プロトコルオプション

以下の表に dump プロトコルを指定した場合に有効となる個別オプションを示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-dump [--dump-format]	有	No	パケットをダンプする形式を指定する。指定可能なダンプ形式は次の3つがある。 BYTE － パケットの内容をそのまま書き出す。 HEX － HEX ダンプ形式に変換して書き出す。 BASE64 － Base64 形式に変換して書き出す。	BYTE
-dumplog [--dump-log-dir]	有	No	ダンプログの出力先のディレクトリ名	カレントディレクトリ

5.2.4 ssl プロトコルオプション

以下の表に 受信方向のプロトコル指定に に ssl プロトコルを指定した場合に有効となる個別オプションを示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-ss [--server-keystore]	有	No(注 1)	SSL サーバ ソケットを作成する際使用するキーストアファイルを指定する(注 2)	システムにハードコードされたキーストア
-sspass [--server-keystore-password]	有	必ず-ss オプションと併用する必要がある	-ss オプションで指定したキーストアのパスワードを指定する。	-
-skpass [--server-keys-password]	有	必ず-ss オプションと併用する必要がある	-ss オプションで指定したキーストア内のキーのパスワードを指定する。	-sspass で指定したパスワード
-needauth [--server-need-client-authentication]	無	No	クライアント認証を実施する。	-

(注 1) 実際の使用では、必ず自前でキーストアを用意して、このオプションを指定すること。

(注 2) キーストアに入れなければならないものは以下となります。

a) サーバ証明書と秘密鍵

b) サーバ証明書を署名した CA の証明書

さらに、-needauth オプションにより、クライアント側に証明書の提示を求めた場合は、以下も必要となります。

c) クライアント証明書を署名した CA の証明書

以下の表に 送信方向のプロトコル指定に に ssl プロトコルを指定した場合に有効となる個別オプションを示します。

オプション名	引数の有無	必須か	意味	デフォルト値
-cs [--client-keystore]	有	No	SSL ソケットを作成する際使用するキーストアファイルを指定する(注 1)	システムにハードコードされたキーストア
-cspass [--client-keystore-password]	有	必ず-cs オプションと併用する必要がある	-cs オプションで指定したキーストアのパスワードを指定する。	-
-ckpass [--client-keys-password]	有	No	-cs オプションで指定したキーストア内のキーのパスワードを指定する。	-cspass で指定したパスワード
-igauth [--client-ignore-server-authentication]	無	No	サーバ認証を実施しない。	-

(注 1) キーストアに入れるものは以下となります。

a) サーバ証明書を署名した CA の証明書(ただし、-igauth オプションを指定した場合は不要)

さらに、サーバ側が -needauth オプションにより、クライアント側に証明書の提示を求めている場合は、以下も必要となります。

b) クライアント証明書と秘密鍵

c) クライアント証明書を署名した CA の証明書