HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ
Cosminexus HTTP Server

手引書

3021-3-J18-80

前書き

■ 対象製品

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の前書きの対象製品の説明を参照してください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

https://www.apache.org/licenses/LICENSE-2.0
This product includes the nghttp2 library licensed under MIT License.

■ マイクロソフト製品のスクリーンショットの使用について

マイクロソフトの許可を得て使用しています。

■ 発行

2026 年 1 月 3021-3-J18-80

■ 著作権

変更内容

変更内容(3021-3-J18-80) uCosminexus Application Server 11-70，uCosminexus Client 11-70，uCosminexus Developer 11-70，uCosminexus Service Architect 11-70，uCosminexus Service Platform 11-70

追加・変更内容	変更個所
マニュアル訂正の内容を反映した。	-

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルをお読みになる際の前提情報については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」のはじめにの説明を参照してください。

目次

1 HTTP Server とは

この章では，HTTP Server の概要について説明します。

1.1 HTTP Server の概要

トランザクション処理を含む業務システムや，基幹系システムなどの，ミッションクリティカルな環境で利用できる基幹業務システム向け Web サーバである HTTP Server は，きめ細かな保守サービス，テクニカルサービスによって，信頼性の高いシステムをサポートしています。

1.2　HTTP Server の特長

HTTP Server は，全世界で高いシェアを持つ Apache HTTP Server をベースに開発しています。HTTP Server でのサポート範囲は，このマニュアルの記述範囲です。

HTTP Server で使用できる主な機能には次のものがあります。

- ユーザ認証とアクセス保護
- バーチャルホスト
- リバースプロキシ
- 流量制限機能
- 有効期限設定機能
- ヘッダカスタマイズ機能
- CGI プログラムの実行
- ディレクトリインデクス表示
- イメージマップ

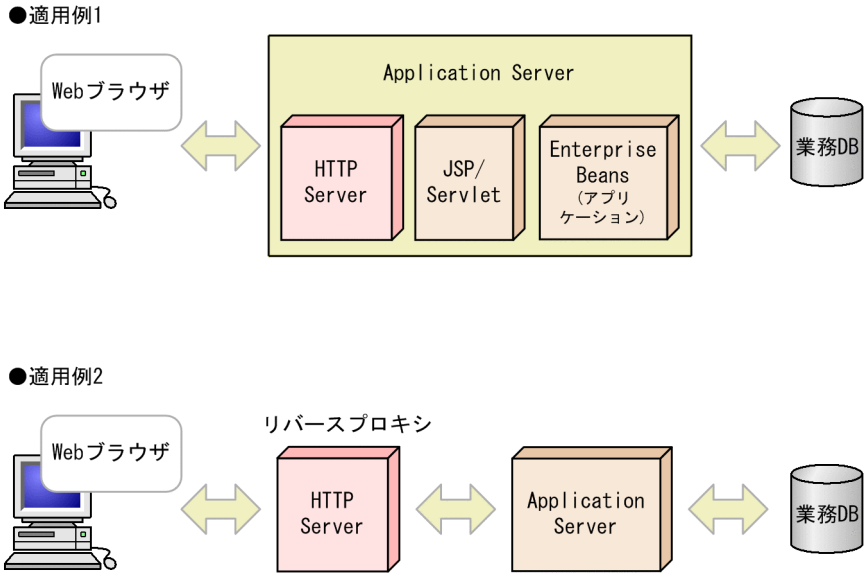
また，この製品は，SSL/TLS ライブラリとして高いシェアをもつ OpenSSL を導入して，SSL（Secure Sockets Layer）を実装しています。これによって，データの改ざん，なりすまし（クライアントから見たサーバのなりすましおよびサーバから見たクライアントのなりすまし）および盗聴を防止し，情報の安全性を確保できます。

適用例

HTTP Server は，Application Server を構成する製品の一つです。

適用例を次に示します。

図 1-1　適用例



2 運用の準備と起動，停止（UNIX 版）

この章では，HTTP Server を運用する前に，知っておいていただきたいことおよび起動と停止について説明します。

2.1 HTTP Server を運用するためのシステム構成

HTTP Server を運用するために必要なシステム構成（ハードウェア構成）について説明します。

サーバ

HTTP Server の適用機種，使用するメモリ所要量，およびディスク占有量については，リリースノートを参照してください。

クライアント

Web ブラウザが動作できる端末

ネットワーク関連

- イーサネットなどのネットワーク（必須）
- ドメインネームシステムサーバ（任意）
- ロードバランサ（任意）
- SSL アクセラレータ（任意）
- ファイアウォール（任意）

2.2 インストールとアンインストール

HTTP Server は，Application Server のインストールによって使用できます。

Application Server のインストールとアンインストールの方法は次のとおりです。詳細手順については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

- インストール
  インストーラを使用します。HTTP Server は次のディレクトリにインストールされます。
  /opt/hitachi/httpsd
- アンインストール
  PP インストーラを使用します。

2.3　運用環境を定義する

HTTP Server の動作を定義するファイルについて説明します。

2.3.1　環境の定義方法

(1)　ディレクトリ構成

HTTP Server をインストールしたときの，ディレクトリ構成を次に示します。この構成は変更しないでください。

図 2-1　ディレクトリ構成

```
/
 └opt
   └hitachi
     └httpsd            ルートディレクトリ
       ├admin
       │ └bin           複数サーバ環境生成コマンド格納ディレクトリ
       │                 (httpsd.conf.org, httpsd.conf.org.prefork,
       │                 hwsserveredit, hwsconfigedit)
       ├bin              実行ファイル格納ディレクトリ(htpasswd)
       ├build            モジュール作成用ファイル格納ディレクトリ
       ├cgi-bin          CGIプログラム格納ディレクトリ
       ├conf             設定ファイル格納ディレクトリ(httpsd.conf, mime.types)
       │ └ssl            SSL用ディレクトリ
       │   ├cacert       CA証明書格納ディレクトリ
       │   ├cacerts      CA証明書のハッシュリンク用ディレクトリ
       │   ├crl          CRL格納ディレクトリ
       │   └server       サーバ秘密鍵，サーバ証明書用ディレクトリ
       ├htdocs           デフォルトドキュメントルートディレクトリ(index.html)
       ├icons            アイコン画像格納ディレクトリ
       ├include          ヘッダファイル格納ディレクトリ
       ├libexec          共有ライブラリ，モジュール格納ディレクトリ(
       │                 mod_expires.so, mod_headers.so, mod_http2.so, mod_hws_qos.so,
       │                 mod_mpm_prefork.so, mod_mpm_worker.so, mod_proxy.so,
       │                 mod_proxy_http.so, mod_proxy_http2.so, mod_proxy_wstunnel.so,
       │                 mod_reqtimeout.so)
       ├logs             ログ，プロセスIDファイル格納ディレクトリ
       ├maintenance      保守情報収集機能用ディレクトリ
       ├sbin             コマンド格納ディレクトリ(httpsd, httpsdctl, logresolve,
       │                 rotatelogs, rotatelogs2, hwstraceinfo, openssl.sh)
       ├servers          複数サーバ環境生成ディレクトリ
       └ssl              OpenSSLディレクトリ
```

(2)　コンフィグファイル

HTTP Server の動作環境を定義するファイルをコンフィグファイルといいます。なお，コンフィグファイルのコメント行以外に，マルチバイト文字および Unicode の補助文字は指定できません。また，コンフィグファイルの 1 行に指定可能な文字数の上限は，8191 文字です。

各ファイルの用途を次に示します。

表 2-1　コンフィグファイルの用途

ファイル名	用途	標準提供
httpsd.conf	HTTP Server の動作環境を各種ディレクティブで定義します。システム管理者が管理します。	○
mime.types	コンテンツのファイル拡張子とコンテントタイプ（MIME タイプ）の関連づけを定義します。システム管理者が管理します。 指定形式は，TypesConfig ディレクティブの説明を参照してください。	○
.htaccess	アクセス制御を定義するアクセスコントロールファイル。必要に応じてエンドユーザがアクセス制御するディレクトリ下に作成します（デフォルトファイル名は.htaccess）。	×
Include ディレクティブで指定したファイル	HTTP Server の動作環境を各種ディレクティブで定義します。システム管理者が管理します。	×

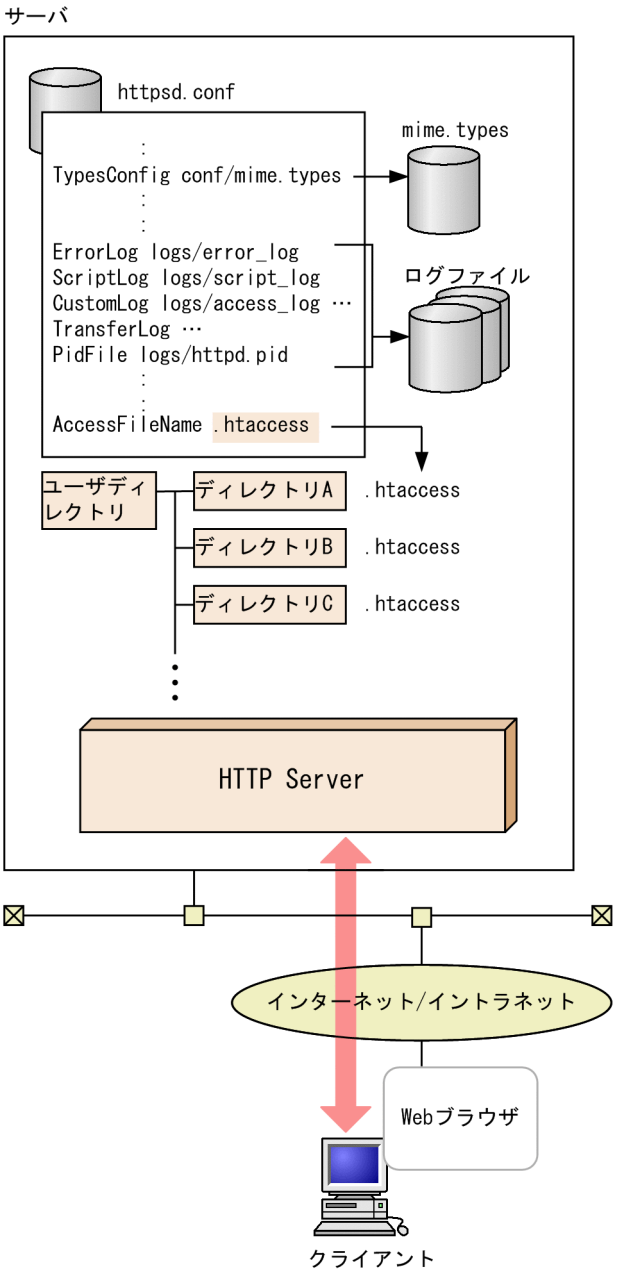
(凡例)

○：標準提供する。

×：標準提供しない。

コンフィグファイルの関連を次に示します。

図 2-2　コンフィグファイルの関連



2.4　起動と停止

HTTP Server の起動および停止方法について説明します。

2.4.1　HTTP Server を起動，停止する（Management Server の使用）

Management Server を使用して HTTP Server を起動，停止できます。詳細は，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「4.1.2　システムの起動方法」または「4.1.4　システムの停止方法」を参照してください。

2.4.2　HTTP Server を起動，停止する（httpsdctl コマンド）

HTTP Server の起動および停止をする httpsdctl コマンドについて説明します。

(1)　形式

```
/opt/hitachi/httpsd/sbin/httpsdctl　{start | stop | restart | graceful | gracefulstop | configtes
t | help}
```

(2)　オプション

- start

  HTTP Server を起動します。PRFSPOOL 環境変数を設定したあとで httpsdctl コマンドを実行して起動してください。

- stop

  HTTP Server を停止します。

- restart

  HTTP Server を再起動します。実行中のサーバプロセスは，直ちに停止します。すべてのサーバプロセス終了後に再起動します。prefork MPM を使用している場合，再起動時に MaxClients ディレクティブ指定値の変更は反映されないで，前回の値が引き継がれます。Listen ディレクティブ指定値および SSL 通信で使用する秘密鍵の設定（SSLCertificateKeyFile ディレクティブ）を変更した場合は，いったん HTTP Server を停止してから，起動し直してください。

- graceful

  HTTP Server を再起動します。実行中のサーバプロセスは，実行終了後に停止します。サーバプロセスは，随時，新しいコンフィグファイルに基づいて起動します。prefork MPM を使用している場合，再起動時に MaxClients ディレクティブ指定値の変更は反映されないで，前回の値が引き継がれます。Listen ディレクティブ指定値および SSL 通信で使用する秘密鍵の設定（SSLCertificateKeyFile ディレクティブ）を変更した場合は，いったん HTTP Server を停止してから，起動し直してください。

- gracefulstop
  HTTP Server を計画停止します。実行中のサーバプロセスは，実行終了後に停止します。KeepAlive 接続中または HTTP/2 通信中の場合，KeepAlive 接続の解除または HTTP/2 通信の終了までサーバプロセスは実行中の状態となります。実行が終了しない場合は，HWSGracefulStopTimeout ディレクティブに指定した待ち時間が経過すると終了します。
- configtest
  コンフィグファイルの文法チェックをします。文法エラーがあると，画面にエラーメッセージを表示します。このオプションを指定した場合は，HTTP Server は起動しません。
- help
  httpsdctl のヘルプを表示させます。

(3) 起動確認方法

HTTP Server の起動を確認するには，制御プロセスを確認してください。詳細は，「4.1.4 稼働管理について」の「(3) 制御プロセスの監視」を参照してください。

また，HTTP Server を起動したあと，エラーログに以下のメッセージが出力されていることを確認してください。

```
Cosminexus HTTP Server バージョン (Unix) configured -- resuming normal operations
```

(4) 使用例

HTTP Server を起動します。

```
export PRFSPOOL=/opt/Cosminexus/PRF/spool
/opt/hitachi/httpsd/sbin/httpsdctl start
```

(5) 注意事項

- httpsdctl stop および gracefulstop による Web サーバ停止操作実行時に，HTTP Server のコンフィグファイルの定義が不正な場合，httpsdctl の実行はエラーとなり Web サーバは停止しません。
- httpsdctl restart および graceful による Web サーバ再起動実行時に，HTTP Server のコンフィグファイルの定義が不正な場合，httpsdctl の実行はエラーとなり Web サーバは停止しないで再起動しません。
- httpsdctl コマンドによる HTTP Server の起動，再起動および停止操作を実行した場合，起動完了および停止完了を示すメッセージは出力されません。
- KeepAlive 接続中または HTTP/2 通信中の状態で httpsdctl gracefulstop による Web サーバ停止操作を実行すると，KeepAlive 接続の解除または HTTP/2 通信の終了までサーバ停止が待たされます。
- コンフィグファイルは，/opt/hitachi/httpsd/conf/httpsd.conf を使用します。

- httpsdctl stop および gracefulstop による Web サーバ停止操作実行時に，PidFile ディレクティブに指定したファイルを削除した場合，または同ファイルの設定内容を変更した場合，httpsdctl の実行はエラーとなり Web サーバは停止しません。
- httpsdctl コマンド実行時に HTTP Server が起動しない場合があります。詳細は，「4.1.1(2)(a)　留意点」の「PidFile ディレクティブ」に関する記載，または，「4.1.2(2)(a)　留意点」の「PidFile ディレクティブ」に関する記載をご参照ください。

2.4.3　HTTP Server を起動する（httpsd コマンド）

サーバのルートディレクトリや httpsd.conf ファイルを指定して起動する場合に，この方法を使用して起動します。

PRFSPOOL 環境変数を設定したあとで httpsd コマンドを実行してください。

(1)　形式

```
/opt/hitachi/httpsd/sbin/httpsd〔〔-d ディレクトリ〕〔-f ファイル名〕〔-R ディレクトリ〕｜-v｜-t〕
```

(2)　オプション

- -d ディレクトリ

  ServerRoot ディレクティブがコンフィグファイルに指定されていない場合の，デフォルト値を指定できます。
- -f ファイル名

  コンフィグファイルのパスを指定できます。絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスで指定します。未設定時は，/opt/hitachi/httpsd/conf/httpsd.conf を使用します。
- -R ディレクトリ

  DSO 実行ライブラリが格納されているディレクトリを絶対パスで指定します。
- -v

  バージョン情報を表示させます。このオプションを指定した場合は，HTTP Server は起動しません。
- -t

  コンフィグファイルの文法チェックをします。文法エラーがあると，画面にエラーメッセージを表示します。このオプションを指定した場合は，HTTP Server は起動しません。

(3)　再起動方法

kill コマンドで HTTP Server を再起動できます。

```
kill　{-HUP｜-USR1}　`cat PidFileディレクティブ指定値`
```

- -HUP
  httpsdctl コマンドの restart に相当する再起動をします。
- -USR1
  httpsdctl コマンドの graceful に相当する再起動をします。
- PidFile ディレクティブ指定値
  PidFile ディレクティブで指定した値(ファイル名)を指定します。

(4) 終了方法

httpsd コマンドで HTTP Server を起動した場合，次に示すコマンドを実行してプロセスを終了し，HTTP Server を停止してください。

```
kill {-TERM | -USR2} `cat PidFileディレクティブ指定値`
```

- -TERM
  httpsdctl コマンドの stop に相当する停止をします。
- -USR2
  httpsdctl コマンドの gracefulstop に相当する停止をします。

(5) 起動確認方法

HTTP Server の起動を確認するには，制御プロセスを確認してください。詳細は，「4.1.4 稼働管理について」の「(3) 制御プロセスの監視」を参照してください。

また，HTTP Server を起動したあと，エラーログに以下のメッセージが出力されていることを確認してください。

```
Cosminexus HTTP Server バージョン (Unix) configured -- resuming normal operations
```

(6) 注意事項

httpsd コマンド実行時に HTTP Server が起動しない場合があります。詳細は，「4.1.1(2)(a) 留意点」の「PidFile ディレクティブ」に関する記載，または，「4.1.2(2)(a) 留意点」の「PidFile ディレクティブ」に関する記載をご参照ください。

2.4.4 一般ユーザアカウントによる運用

HTTP Server は，通常の運用方法として，スーパーユーザによる運用を想定しています。

インストールした状態では，スーパーユーザによる運用ができるように各種設定が施されています。

このことから，スーパーユーザ以外のユーザ（以下，一般ユーザと呼びます）で運用する場合，HTTP Server の設定ファイルや関連するディレクトリ・ファイルの各種設定内容の変更が必要になります。また，HTTP Server の一部の機能については，一般ユーザによる運用は制限事項になるものがあります。

ここでは，スーパーユーザと一般ユーザの違い，一般ユーザによる HTTP Server を運用するための環境構築方法，制限事項について説明します。

(1) 各プロセスの権限

スーパーユーザまたは一般ユーザで運用した場合，HTTP Server の各プロセスの権限を次に示します。

表 2-2 各プロセスの権限

項番	プロセス	スーパーユーザによる運用	一般ユーザによる運用
1	制御プロセス	スーパーユーザ	一般ユーザ
2	rotatelogs，rotatelogs2 プロセス		
3	サーバプロセス	User，Group ディレクティブで指定したユーザ，グループ	
4	CGI プロセス		

(2) UNIX におけるスーパーユーザと一般ユーザの違い

UNIX において，スーパーユーザは一般ユーザと異なり，システムの管理者権限を持つユーザになります。UNIX におけるスーパーユーザと一般ユーザの権限の差異（一例）を次に示します。

表 2-3 UNIX におけるスーパーユーザと一般ユーザの権限の差異（一例）

項番	項目	スーパーユーザ	一般ユーザ
1	別のユーザが起動したプロセスの停止	可	不可
2	well-known ポート（1023 番以下のポート）を開く	可	不可
3	明示的に読み取り/書き込み権限が与えられていないファイルへのアクセス	可	不可

一般ユーザで HTTP Server を運用する場合，HTTP Server の制御プロセスの権限が一般ユーザ権限で動作するため，このときの挙動はスーパーユーザで HTTP Server を運用した場合と異なる場合があります。したがって，一般ユーザで HTTP Server を運用する場合は，スーパーユーザとの権限の差異を意識しながら環境を構築する必要があります。

(3) リソースの所有者・グループの変更

HTTP Server のコンテンツ，設定ファイル類，および HTTP Server が動作する際にアクセスする各種ファイル・ディレクトリについて，UNIX 上での所有者・グループを変更します。

最低限，インストールディレクトリ（/opt/hitachi/httpsd ディレクトリ）以下のリソースに対しては変更が必要です。

将来，リソースの所有者・グループを元に戻したい場合は，変更作業の前に現在のリソースに対して，所有者とグループを保存しておきます。

保存作業は，スーパーユーザで実行します。保存例を以下に示します。

（例）

/opt/hitachi/httpsd ディレクトリ以下のリソースに対して，所有者とグループの一覧を作成する。

```
ls -laR /opt/hitachi/httpsd
```

変更作業は，スーパーユーザで実行します。変更例を以下に示します。

（例）

/opt/hitachi/httpsd ディレクトリ以下のリソースに対して，所有者(hwsuser)とグループ(hwsgroup)を変更する。

```
chown -R hwsuser:hwsgroup /opt/hitachi/httpsd
```

(4) httpsd の起動

HTTP Server を運用する一般ユーザを使用して，httpsd を起動してください。

httpsd の停止または再起動をする場合は，起動時と同じ一般ユーザで操作してください。

(5) 制限事項

次に示すコマンドは，一般ユーザによる運用に対応していません。スーパーユーザで運用してください。

- htpasswd コマンド
- hwscollect コマンド
- hwsserveredit コマンド
- logresolve コマンド
- openssl.sh コマンド

一般ユーザによる運用では次に示すディレクティブは指定できません。指定があっても無視します。

- Group ディレクティブ
- User ディレクティブ

一般ユーザによる運用では，well-known ポート（1023 番以下のポート）を開くことができません。

以下のディレクティブにポート番号を指定する際は注意してください。

- Listen ディレクティブ
- Port ディレクティブ

3 運用の準備と起動，停止（Windows 版）

この章では，HTTP Server を運用する前に，知っておいていただきたいことおよび起動と停止について説明します。

3.1　HTTP Server を運用するためのシステム構成

HTTP Server を運用するために必要なシステム構成について説明します。

3.1.1　ハードウェア構成

(1)　サーバ

HTTP Server の適用機種や，使用するメモリ所要量およびディスク占有量については，リリースノートを参照してください。

(2)　クライアント

Web ブラウザが動作できる端末

(3)　ネットワーク関連

- イーサネットなどのネットワーク（必須）
- ドメインネームシステムサーバ（任意）
- ロードバランサ（任意）
- SSL アクセラレータ（任意）
- ファイアウォール（任意）

3.1.2　Windows 使用時の注意事項

(1)　コマンド実行時の注意事項

Windows で HTTP Server を動作させる場合，このマニュアルに記載されているコマンドはすべて管理者権限で実行する必要があります。

HTTP Server の前提 OS については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」を参照してください。

HTTP Server のコマンドは，「管理者：コマンドプロンプト」で実行してください。「管理者：コマンドプロンプト」は，各 OS で提供されている機能を使用して起動してください。

(2)　設定ファイル更新時の注意事項

Windows で HTTP Server の設定ファイルを更新する場合は，更新するプログラムを必ず管理者権限で実行してください。

HTTP Server の前提 OS については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」を参照してください。

3.2 インストールとアンインストール

HTTP Server は Application Server のインストールによって使用できます。

Application Server のインストールとアンインストールの方法は次のとおりです。詳細手順については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

- インストール
  インストーラを使用します。HTTP Server は次のディレクトリにインストールされます。
  <Application Server のインストールディレクトリ>¥httpsd
- アンインストール
  PP インストーラを使用します。

3.3　運用環境を定義する

HTTP Server の動作を定義するファイルについて説明します。

3.3.1　環境の定義方法

(1)　ディレクトリ構成

HTTP Server をインストールしたときの，ディレクトリ構成を次に示します。この構成は変更しないでください。

図 3-1　ディレクトリ構成

```
インストール先ディレクトリ
└httpsd             ルートディレクトリ(httpsd.exe)
  ├admin
  │ └bin            複数サーバ環境生成コマンド格納ディレクトリ(httpsd.conf.org,
  │                  hwsserveredit.exe, hwsconfigedit.exe)
  ├bin              実行ファイル格納ディレクトリ(htpasswd.exe)
  ├cgi-bin          CGIプログラム格納ディレクトリ
  ├conf             設定ファイル格納ディレクトリ(httpsd.conf, mime.types)
  │ └ssl            SSL用ディレクトリ
  │   ├cacert       CA証明書格納ディレクトリ
  │   ├crl          CRL格納ディレクトリ
  │   └server       サーバ秘密鍵，サーバ証明書用ディレクトリ
  ├htdocs           デフォルトドキュメントルートディレクトリ(index.html)
  ├icons            アイコン画像格納ディレクトリ
  ├include          ヘッダファイル格納ディレクトリ
  ├libexec          共有ライブラリ格納ディレクトリ
  ├logs             ログ，プロセスIDファイル格納ディレクトリ
  ├modules          モジュール格納ディレクトリ(mod_expires.so, mod_headers.so,
  │                  mod_http2.so, mod_hws_qos.so, mod_proxy.so, mod_proxy_http.so,
  │                  mod_proxy_http2.so, mod_proxy_wstunnel.so, mod_reqtimeout.so)
  ├sbin             コマンド格納ディレクトリ(logresolve.exe, rotatelogs.exe,
  │                  rotatelogs2.exe, hwstraceinfo.exe, openssl.bat)
  ├servers          複数サーバ環境生成ディレクトリ
  └ssl              OpenSSLディレクトリ
```

(2)　コンフィグファイル

HTTP Server の動作環境を定義するファイルをコンフィグファイルといいます。なお，コンフィグファイルのコメント行以外に，マルチバイト文字および Unicode の補助文字は指定できません。また，コンフィグファイルの 1 行に指定可能な文字数の上限は，8191 文字です。

各ファイルの用途を次に示します。

表 3-1　コンフィグファイルの用途

ファイル名	用途	標準提供
httpsd.conf	HTTP Server の動作環境を各種ディレクティブで定義します。システム管理者が管理します。	○

ファイル名	用途	標準提供
mime.types	コンテンツのファイル拡張子とコンテントタイプ（MIME タイプ）の関連づけを定義します。システム管理者が管理します。 指定形式は，TypesConfig ディレクティブの説明を参照してください。	○
.htaccess	アクセス制御を定義するアクセスコントロールファイル。必要に応じてエンドユーザがアクセス制御するディレクトリ下に作成します（デフォルトファイル名は.htaccess）。	×
Include ディレクティブで指定したファイル	HTTP Server の動作環境を各種ディレクティブで定義します。システム管理者が管理します。	×

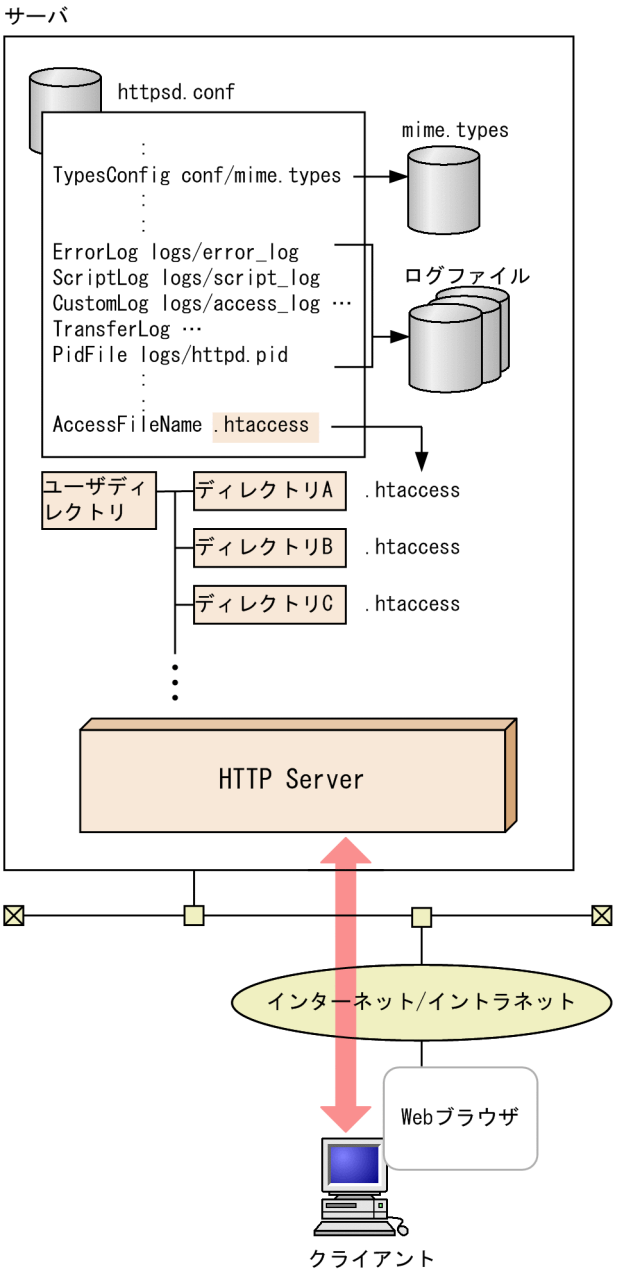
（凡例）

○：標準提供する。

×：標準提供しない。

コンフィグファイルの関連を次に示します。

図 3-2　コンフィグファイルの関連



3.4　起動と停止

HTTP Server の起動および停止方法について説明します。

3.4.1　HTTP Server の起動，停止

HTTP Server をインストールすると，"Cosminexus HTTP Server"という名称のサービスとしてシステムに登録されます。このとき，手動起動するサービスとして登録されるため，システム起動時には自動起動されません。

HTTP Server を起動，停止および再起動するには，次の方法があります。

- Management Server からサービスとしての起動，停止および再起動
- コントロールパネルからサービスとしての起動，停止
- コマンドプロンプトからの起動，停止および再起動

HTTP Server をサービスとして実行する場合のユーザアカウントは，インストール時点では"LocalSystem"です。HTTP Server は，CGI プログラム，API 接続モジュールを含め，このユーザアカウントで実行されます。それ以外のユーザアカウントで実行したい場合は，「3.4.2　一般ユーザアカウントによる運用」を参照してください。

(1)　Management Server からサービスとしての起動，停止および再起動

詳細は，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「4.1.2　システムの起動方法」または「4.1.4　システムの停止方法」を参照してください。

(2)　コントロールパネルからサービスとしての起動，停止

コントロールパネルからサービス画面を表示し，次に「Cosminexus HTTP Server」を選択して，起動する場合は「開始(S)」ボタン，停止する場合は「停止(T)」ボタンを押します。サービス画面からの再起動はできません。コンフィグファイルは，インストールパス/httpsd/conf/httpsd.conf を使用します。

(3)　コマンドプロンプトからの起動，停止および再起動

コマンドプロンプトから httpsd コマンドを入力します。httpsd コマンドについて次に説明します。

(a)　形式

```
"<Application Serverのインストールディレクトリ>¥httpsd¥httpsd.exe" 〔〔-d ディレクトリ〕 〔-f
 ファイル名〕 〔〔-n "サービス名"〕 〔-k {start | stop | restart | gracefulstop | install |
uninstall} 〕〕 | -v | -t 〕
```

(b) オプション

- -d ディレクトリ

  ServerRoot ディレクティブがコンフィグファイルに指定されていない場合の，デフォルト値を設定できます。

- -f ファイル名

  コンフィグファイルのパスを指定できます。絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスで指定します。

- -n "サービス名"

  HTTP Server のサービス名を指定します。サービス名は，"（引用符）で囲んで指定してください。サービス名に指定できる文字数の上限値は，128 文字です。サービス名は，ASCII コードで指定してください。また，次に示す文字は指定できません。

  '¥'，'/'，'"'，制御コード，マルチバイト文字

  サービス名のデフォルト値は「Cosminexus HTTP Server」です。

  本オプションを指定する場合は，-k オプションも合わせて指定する必要があります。

- -k start

  HTTP Server を起動します。-n "サービス名"が指定されている場合は，該当するサービスを起動します。

- -k stop

  HTTP Server を停止します。-n "サービス名"が指定されている場合は，該当するサービスを停止します。

- -k restart

  HTTP Server を再起動します。

- -k gracefulstop

  HTTP Server を計画停止します。実行中のサーバスレッドは，実行終了後に停止します。KeepAlive 接続中または HTTP/2 通信中の場合，KeepAlive 接続の解除または HTTP/2 通信の終了までサーバプロセスは実行中の状態となります。実行が終了しない場合は，HWSGracefulStopTimeout ディレクティブに指定した待ち時間が経過すると終了します。

- -k install

  HTTP Server をサービスとして登録します。-n "サービス名"が指定されている場合は，該当するサービスを登録します。サービス登録時，スタートアップの種類は「手動」になります。サービス起動する HTTP Server の ServerRoot ディレクティブのデフォルト値は，このコマンド実行時の httpsd.exe のパスまたは-d オプションで指定した値になります。

- -k uninstall

  HTTP Server をサービスから削除します。-n "サービス名"が指定されている場合は，該当するサービスを削除します。削除しようとしたサービスが起動中の場合は，サービスを停止してからサービスを削除します。

- -v

バージョン情報を表示します。このオプションを指定した場合は，HTTP Server は起動しません。

- -t

  コンフィグファイルの文法をチェックします。文法エラーがあると，画面にエラーメッセージを表示します。このオプションを指定した場合は，HTTP Server は起動しません。

(4) ターミナルサービスを利用したリモートマシンからの HTTP Server の操作

HTTP Server では，Windows のターミナルサービス（リモートデスクトップサービス）機能を利用して，サーバマシン上にある HTTP Server の起動・停止，コマンドの実行などの操作をリモートマシンから実行できます。

HTTP Server の前提 OS については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」を参照してください。

ターミナルサービスの操作については，OS のマニュアルを参照してください。

(5) 注意事項

- コントロールパネルからの停止，コマンドプロンプトからの-k stop オプションによる停止時に，サーバスレッドが実行中の場合には，実行終了を最大 30 秒間待ったあとに停止します。
- KeepAlive 接続中または HTTP/2 通信中の状態で httpsdctl gracefulstop による Web サーバ停止操作を実行すると，KeepAlive 接続または HTTP/2 通信の終了までサーバ停止が待たされます。

3.4.2 一般ユーザアカウントによる運用

HTTP Server をサービスとして実行する場合のユーザアカウントは，インストール時点では"LocalSystem"です。HTTP Server は，CGI プログラム，API 接続モジュールを含め，このユーザアカウントで実行されます。

ここでは，さまざまな権限が与えられたグループには所属しないで，Web サーバの動作に必要な権限だけが設定された一般ユーザアカウントで運用する方法について説明します。

(1) 一般ユーザアカウントの作成

HTTP Server サービスを起動する一般ユーザアカウントを作成する方法について説明します。

一般ユーザアカウントの作成方法

1. コントロールパネルから［管理ツール］－［コンピュータの管理］を開きます。
2. ［コンピュータの管理］－［システムツール］－［ローカルユーザーとグループ］－［ユーザー］を開きます。
3. 操作メニューから［新しいユーザー］を選択し，必要事項を入力します。

パスワードは必ず入力してください。

新規作成した一般ユーザアカウントは，デフォルトではグループの設定が付加されています。次の手順に従って，グループの設定を削除してください。

グループの設定の削除方法

1. コントロールパネルから［管理ツール］－［コンピュータの管理］を開きます。
2. ［コンピュータの管理］－［システムツール］－［ローカルユーザーとグループ］－［ユーザー］を開きます。
3. 新規作成したユーザの［プロパティ］を開き，［所属するグループ］タブを表示します。
4. 登録されているグループを削除します。

(2)　ユーザ権利の割り当て

新規作成した一般ユーザアカウントに，ユーザ権利を割り当てる方法について説明します。

ユーザ権利の割り当て方法

1. コントロールパネルから［管理ツール］－［ローカルセキュリティポリシー］を開きます。
2. ［セキュリティの設定］－［ローカルポリシー］－［ユーザー権利の割り当て］を開きます。
3. ［サービスとしてログオン］をダブルクリックして開きます。
4. 「ユーザーまたはグループの追加」ボタンで該当するユーザアカウントを追加します。

［サービスとしてログオン］の権限を明示的に設定しない場合でも，サービスのログオンアカウントを変更した一般ユーザアカウントには，権限が自動的に付加されます。サービスのログオンアカウントの変更については，「3.4.2(3)　サービスのログオンアカウントの変更」を参照してください。

(3)　サービスのログオンアカウントの変更

HTTP Server サービスのログオンアカウントを一般ユーザアカウントに変更する方法について説明します。

サービスのログオンアカウントの変更方法

1. コントロールパネルから［管理ツール］－［サービス］を開きます。
2. Cosminexus HTTP Server の［プロパティ］－［ログオン］タブを開きます。
3. 「アカウント」ラジオボタンを選択し，一般ユーザアカウントを設定します。このとき，「3.4.2(1)　一般ユーザアカウントの作成」で設定したパスワードを正しく入力してください。また，パスワードを無期限にするかどうかを指定してください。

(4) ディレクトリおよびファイルのアクセス権限の設定

HTTP Server がアクセスするディレクトリおよびファイルのアクセス権限に，作成した一般ユーザアカウントのフルコントロール権限を追加してください。

(5) サービスの起動

サービス起動権限を持つユーザアカウントで HTTP Server サービスを起動してください。一般ユーザアカウントには，サービス起動権限はありません。

(6) 注意事項

hwstraceinfo コマンドを使用する場合は，「3.4.2(3) サービスのログオンアカウントの変更」で指定した一般ユーザアカウントで実行してください。Administrators 権限を持つユーザアカウントでは実行できません。

4 システムの運用方法

この章では，Web サーバ環境を運用に合わせて設定するディレクティブおよびコマンドの使用方法について説明します。

4.1　HTTP Server の処理とディレクティブとの関係

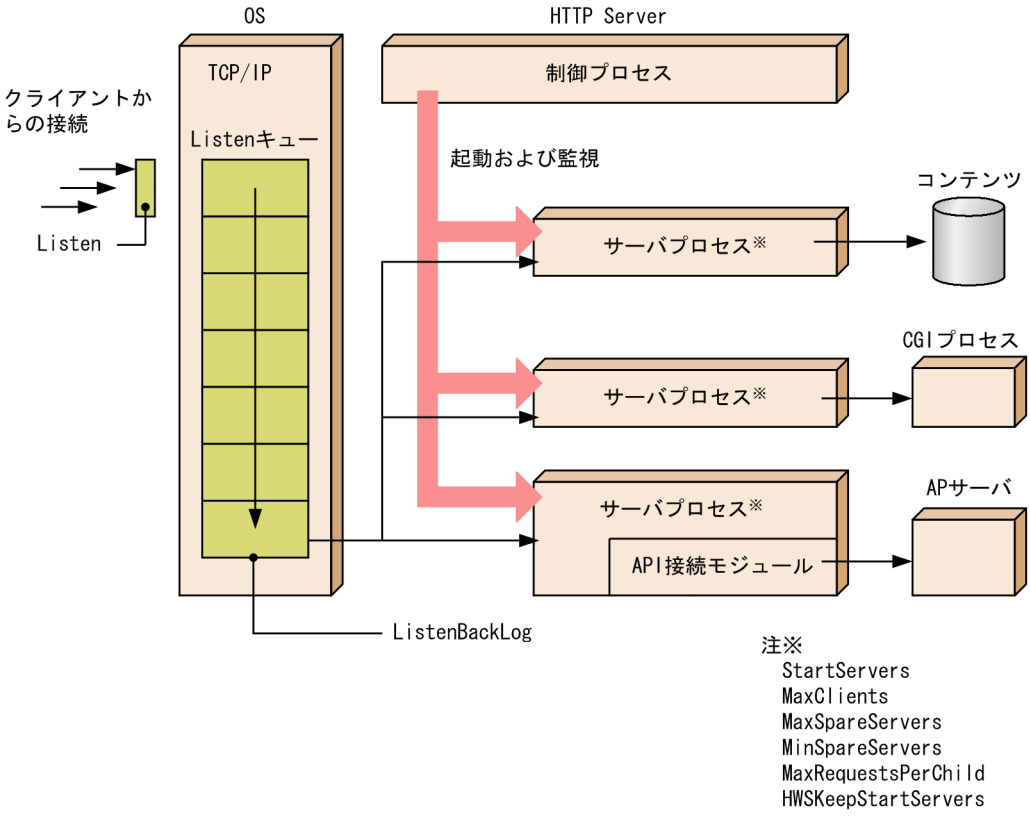
HTTP Server の処理とディレクティブとの関係について，説明します。

4.1.1　HTTP Server のプロセス構造（UNIX 版かつ prefork MPM モジュールの場合）

(1)　プロセス構造

HTTP Server のプロセス構造を次に示します。

図 4-1　HTTP Server のプロセス構造（UNIX 版かつ prefork MPM モジュールの場合）



HTTP Server を起動すると，制御プロセスが起動します。制御プロセスは，リクエストを処理するサーバプロセスを起動し，その稼働を監視します。制御プロセスは，最初に StartServers ディレクティブで指定した個数のサーバプロセスを生成します。その後のサーバプロセス数は，MinSpareServers，MaxSpareServers ディレクティブ指定値に基づいて増減していきます。サーバプロセス数の最大値は，MaxClients ディレクティブで指定します。サーバプロセス数の増減は，制御プロセスが管理します。この処理をメンテナンスと呼びます。

クライアントからの TCP 接続は，Listen ディレクティブで指定した IP アドレスとポートから OS が受信し，OS 内の Listen キューに保留します。Listen キューのサイズは，ListenBacklog ディレクティブで指

定できます。Listen キューに格納できなかった TCP 接続は確立されません（このとき，HTTP Server はクライアントにステータスコードを返却しません）。Listen キューに格納された TCP 接続は，サーバプロセスの一つが取り出して処理を行います。

一つのサーバプロセスは，一つの TCP 接続を取得して処理します。また，一つのサーバプロセスは，MaxRequestsPerChild ディレクティブに指定した個数の HTTP リクエストを処理すると終了します。このときは，制御プロセスが新たなサーバプロセスを生成して処理を続行します。

制御プロセスは，HTTP Server を起動したユーザ，グループ権限で動作します。サーバプロセスは，User，Group ディレクティブで指定したユーザ，グループ権限で動作します。制御プロセスおよびサーバプロセスともに，プロセス名（実行プログラム名）は httpsd です。制御プロセスのプロセス ID は，PidFile ディレクティブに指定したファイルに出力します。

また，HTTP Server のリバースプロキシ機能を使用した場合，バックエンドサーバとの間にコネクションが確立されます。そのコネクションの最大数については，「4.7.4(5)　性能に関する注意事項」を参照してください。

(2)　プロセス数の遷移

メンテナンスは，サーバの負荷集中を避けるために，1 秒ごとに 2^{n-1}（n は連続メンテナンス実行回数，6 以上は n=6）個ずつサーバプロセスを生成します。サーバプロセスは，MinSpareServers ディレクティブで指定した数の待ちプロセスができるかまたは全プロセス数が MaxClients ディレクティブで指定した数になるまで生成されます。1 回のメンテナンスで 8 個以上のサーバプロセスを生成した場合は，エラーログ（info レベル）にその旨出力します。

リクエストの処理が終了すると，サーバプロセスは待ち状態になります。待ち状態のプロセスが増加すると，メンテナンスのタイミングで，MaxSpareServers ディレクティブで指定した数だけ残して，サーバプロセスを終了させます。

(a)　留意点

- StartServers ディレクティブには，Web サーバの起動・再起動直後から大量のリクエストを処理しなければならないような場合は，大きな値を指定してください。
- Web サーバを起動したあとのプロセス数は，MaxSpareServers および MinSpareServers ディレクティブによって制御されるため，StartServers ディレクティブ指定値は意味がありません（HWSKeepStartServers ディレクティブで On を指定した場合を除く）。MinSpareServers および MaxSpareServers ディレクティブは，急にリクエストが多発しても対応できるように待ち状態のプロセスを準備するために指定します。プロセスのメンテナンスでエラーログ（info レベル）が頻繁に出力されるような場合には，待ちプロセス数を増やすよう調整してください。
- より多くのサーバプロセスを常時待ち状態にしておくと，より多くのクライアントからの同時接続要求を受け付けられます。しかし，それだけサーバリソースを消費するために注意が必要です。
- CGI プログラムの負荷が高く CPU を使い尽くしているような場合は，MaxClients ディレクティブの値を小さくしリクエストを受け付けないようにする必要があります。MaxClients ディレクティブで指

定した数のプロセスがすべて処理中の場合は，ListenBacklog ディレクティブの指定によって，キューに保留されます。

- 一つのサーバプロセスは，MaxRequestsPerChild ディレクティブで指定された回数のリクエスト処理を実行したあと，終了します。ただし，MaxRequestsPerChild ディレクティブに 0 を指定した場合，リクエスト処理の回数でサーバプロセスが終了することはありません。MaxRequestsPerChild ディレクティブの指定は，エンドユーザが作成したアプリケーションプログラムなどでメモリリークを起こすおそれがある場合に有効です。
- サーバプロセスに異常終了シグナルが送信された場合（API 接続モジュールで障害になった場合も含む），プロセスは，エラーログ（notice レベル）にそのことを出力します。notice レベルのエラーログは，LogLevel ディレクティブの指定に関係なく出力されます。
- StartServers ディレクティブで指定した数のサーバプロセスを，MaxSpareServers，MinSpareServers ディレクティブの指定に関係なく，常に起動しておきたい場合，HWSKeepStartServers ディレクティブで On を指定してください。サーバプロセス数が StartServers ディレクティブで指定した数を下回った場合に，新しいプロセスを生成して回復します。
- HTTP Server が停止すると PidFile ディレクティブに指定したファイルは削除されます。しかし，HTTP Server を停止しないでマシンをシャットダウンした場合など，HTTP Server が外部から強制的に終了させられたときには，PidFile ディレクティブに指定したファイルは削除されないで，次回の HTTP Server の起動に失敗する場合があります。HTTP Server が起動していない状態で PidFile ディレクティブに指定したファイルが存在する場合は，このファイルを削除してから HTTP Server を起動してください。

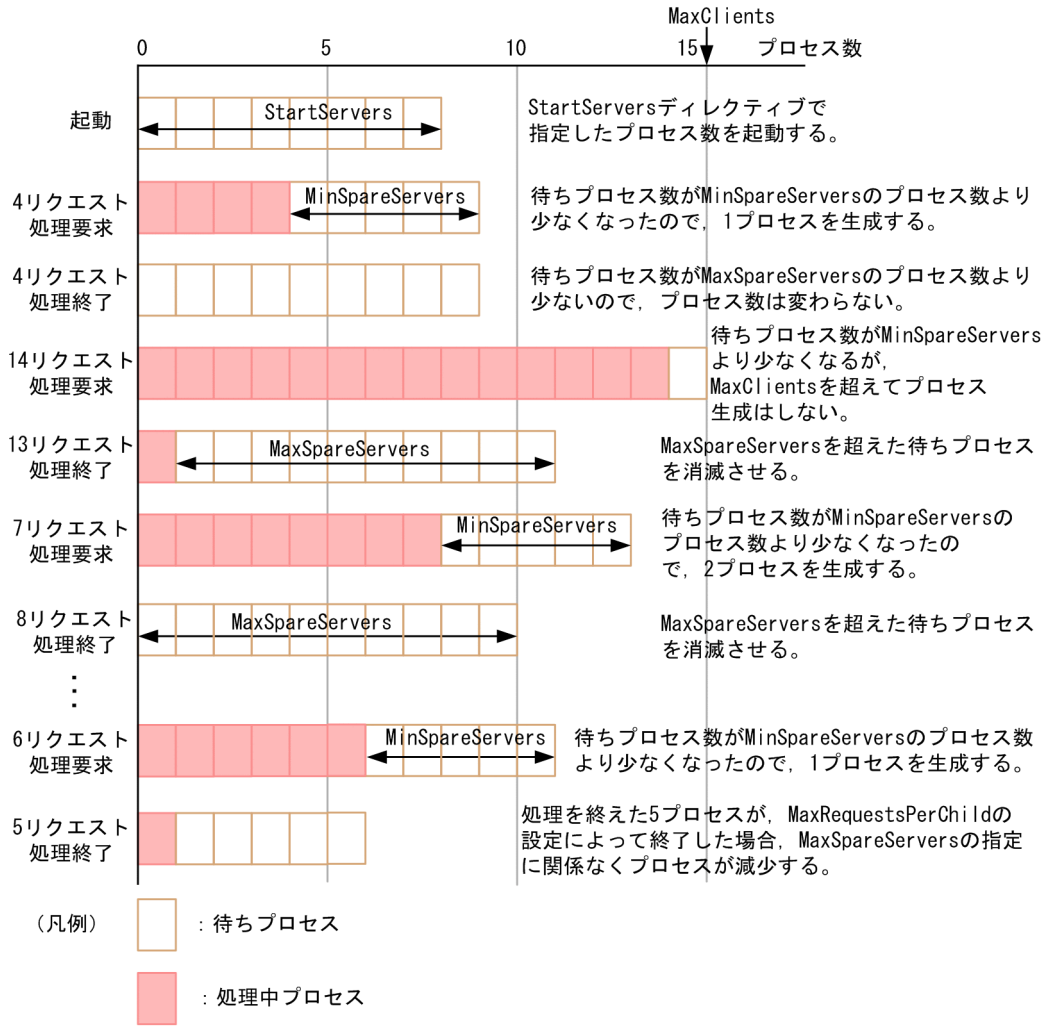
(b) プロセス数の遷移例

HWSKeepStartServers の指定が Off の場合の，プロセス数の遷移例を図 4-2 に示します。

- ディレクティブの指定値（HWSKeepStartServers の指定が Off の場合）

```
LoadModule mpm_prefork_module libexec/mod_mpm_prefork.so
StartServers 8
MaxSpareServers 10
MinSpareServers 5
MaxClients 15
HWSKeepStartServers Off
MaxRequestsPerChild 10000
KeepAlive Off
```

図 4-2　プロセス数の遷移例（HWSKeepStartServers の指定が Off の場合）

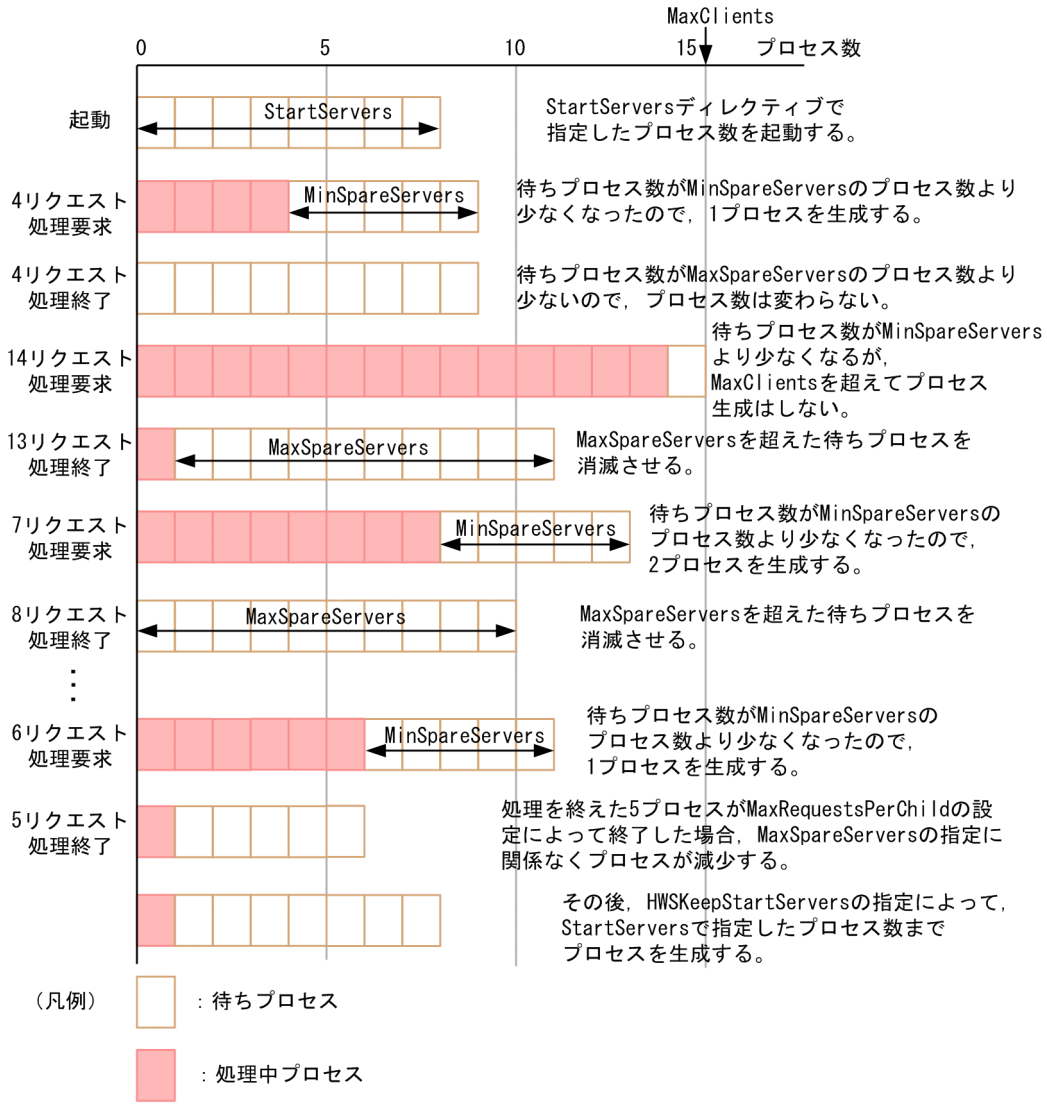


HWSKeepStartServers の指定が On の場合の，プロセス数の遷移例を図 4-3 に示します。

- ディレクティブの指定値（HWSKeepStartServers の指定が On の場合）

```
LoadModule mpm_prefork_module libexec/mod_mpm_prefork.so
StartServers 8
MaxSpareServers 10
MinSpareServers 5
MaxClients 15
HWSKeepStartServers On
MaxRequestsPerChild 10000
KeepAlive Off
```

図 4-3　プロセス数の遷移例（HWSKeepStartServers の指定が On の場合）

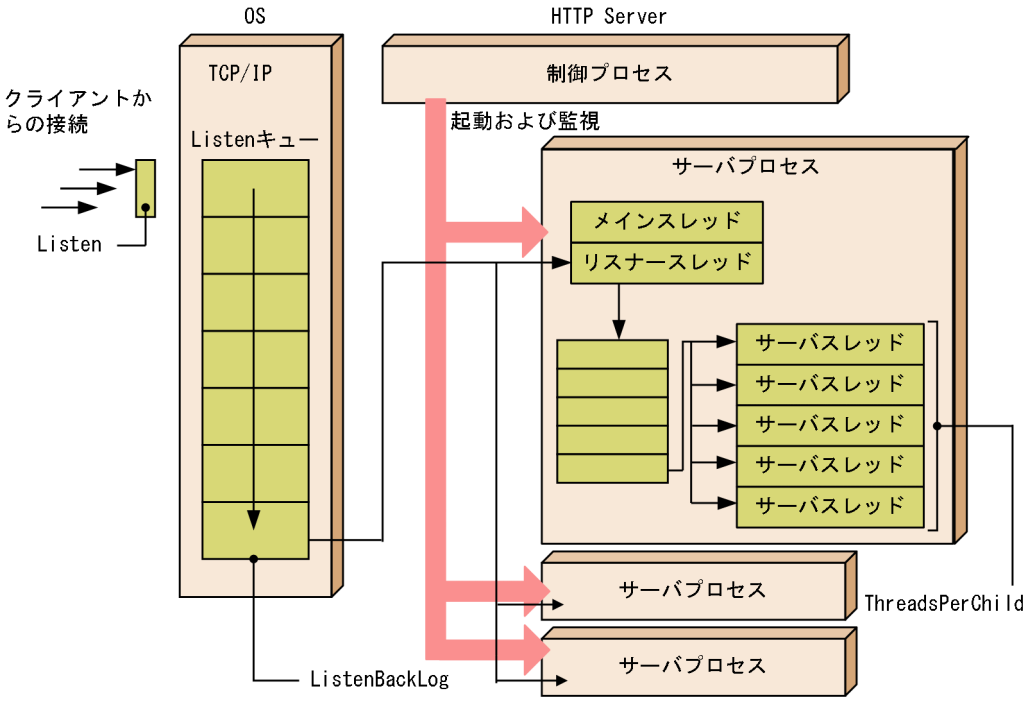


4.1.2　HTTP Server のプロセス構造（UNIX 版かつ worker MPM モジュールの場合）

(1)　プロセス構造

HTTP Server のプロセス構造を次に示します。

図 4-4　HTTP Server のプロセス構造（UNIX 版かつ worker MPM モジュールの場合）



HTTP Server を起動すると，制御プロセスが起動します。制御プロセスは，リクエストを処理するサーバプロセスを起動し，その稼働を監視します。サーバプロセスはメインスレッドのほかに，一つのリスナースレッドと ThreadsPerChild ディレクティブで指定した数だけサーバスレッドを起動します。サーバプロセス数の最大値は，ServerLimit ディレクティブで指定します。

クライアントからの TCP 接続は，Listen ディレクティブで指定した IP アドレスとポートから OS が受信して，OS 内の Listen キューに保留します。Listen キューのサイズは，ListenBacklog ディレクティブで指定できます。Listen キューに格納できなかった TCP 接続は確立されません（このとき，HTTP Server はクライアントにステータスコードを返却しません）。Listen キューに格納された TCP 接続は，サーバプロセスのリスナースレッドが一つ取り出してサーバプロセス内のキューに登録します。そのあと，サーバスレッドがサーバプロセス内のキューから取り出して処理を行います。サーバプロセス内のキューは，ThreadsPerChild ディレクティブで指定した数だけ登録できます。

制御プロセスは，HTTP Server を起動したユーザ，グループ権限で動作します。サーバプロセスは，User，Group ディレクティブで指定したユーザ，グループ権限で動作します。制御プロセスおよびサーバプロセスともに，プロセス名（実行プログラム名）は httpsd です。制御プロセスのプロセス ID は，PidFile ディレクティブに指定したファイルに出力します。

また，HTTP Server のリバースプロキシ機能を使用した場合，バックエンドサーバとの間にコネクションが確立されます。そのコネクションの最大数については，「4.7.4(5)　性能に関する注意事項」を参照してください。

(2) プロセス数・スレッド数の遷移

メンテナンスは，サーバの負荷集中を避けるために，1 秒ごとに 2^{n-1}（n は連続メンテナンス実行回数，6 以上は n=6）個ずつサーバプロセスを生成します。サーバプロセスは，MinSpareThreads ディレクティブで指定した数の待ち状態のサーバスレッドができるか，または全サーバスレッド数が MaxClients ディレクティブで指定した数になるまで生成されます。1 回のメンテナンスで 8 個以上のサーバプロセスを生成した場合は，エラーログ（info レベル）にその旨出力します。

リクエストの処理が終了すると，サーバプロセスは待ち状態になります。待ち状態のサーバスレッドが MaxSpareThreads ディレクティブの値を超えて増加すると，メンテナンスのタイミングでサーバプロセスを終了させます。

(a) 留意点

- worker MPM ではサーバスレッド数に比例して，CGI プロセスの生成処理にコストが掛かり性能が劣化します。そのため CGI プログラムを実行する場合には，prefork MPM を使用することを推奨します。
- StartServers ディレクティブには，Web サーバの起動・再起動直後から大量のリクエストを処理しなければならないような場合は，大きな値を指定してください。
- Web サーバを起動したあとのプロセス数は，MaxSpareThreads および MinSpareThreads ディレクティブによって制御されるため，StartServers ディレクティブ指定値は意味がありません（HWSKeepStartServers ディレクティブで On を指定した場合を除く）。MinSpareThreads および MaxSpareThreads ディレクティブは，急にリクエストが多発しても対応できるように待ち状態のサーバスレッド数を調整するために指定します。プロセスのメンテナンスでエラーログ（info レベル）が頻繁に出力されるような場合には，待ちプロセス数を増やすよう調整してください。
- より多くのサーバプロセスを常時待ち状態にしておくと，より多くのクライアントからの同時接続要求を受け付けられます。しかし，それだけサーバリソースを消費するために注意が必要です。
- すべてのサーバプロセス内のキューが満杯となった場合は，ListenBacklog ディレクティブの指定によって，Listen キューに保留されます。
- 一つのサーバプロセスは，MaxRequestsPerChild ディレクティブで指定された回数のリクエスト処理を実行したあと，終了します。ただし，MaxRequestsPerChild ディレクティブに 0 を指定した場合，リクエスト処理の回数でサーバプロセスが終了することはありません。MaxRequestsPerChild ディレクティブの指定は，エンドユーザが作成したアプリケーションプログラムなどでメモリリークを起こすおそれがある場合に有効です。
- サーバプロセスに異常終了シグナルが送信された場合（API 接続モジュールで障害になった場合も含む），プロセスは，エラーログ（notice レベル）にそのことを出力します。notice レベルのエラーログは，LogLevel ディレクティブの指定に関係なく出力されます。
- StartServers ディレクティブで指定した数のサーバプロセスを，MaxSpareThreads，MinSpareThreads ディレクティブの指定に関係なく，常に起動しておきたい場合，HWSKeepStartServers ディレクティブで On を指定してください。サーバプロセス数が StartServers ディレクティブで指定した数を下回った場合に，新しいプロセスを生成して回復します。

- HTTP Server が停止すると PidFile ディレクティブに指定したファイルは削除されます。しかし，HTTP Server を停止しないでマシンをシャットダウンした場合など，HTTP Server が外部から強制的に終了させられたときには，PidFile ディレクティブに指定したファイルは削除されないで，次回の HTTP Server の起動に失敗する場合があります。HTTP Server が起動していない状態で PidFile ディレクティブに指定したファイルが存在する場合は，このファイルを削除してから HTTP Server を起動してください。
- HTTP/2 プロトコルを使用している場合は，サーバスレッドのほかに HTTP/2 プロトコルを処理するためのワーカスレッドが生成されます。詳細は「4.17.3　ワーカスレッド」を参照してください。

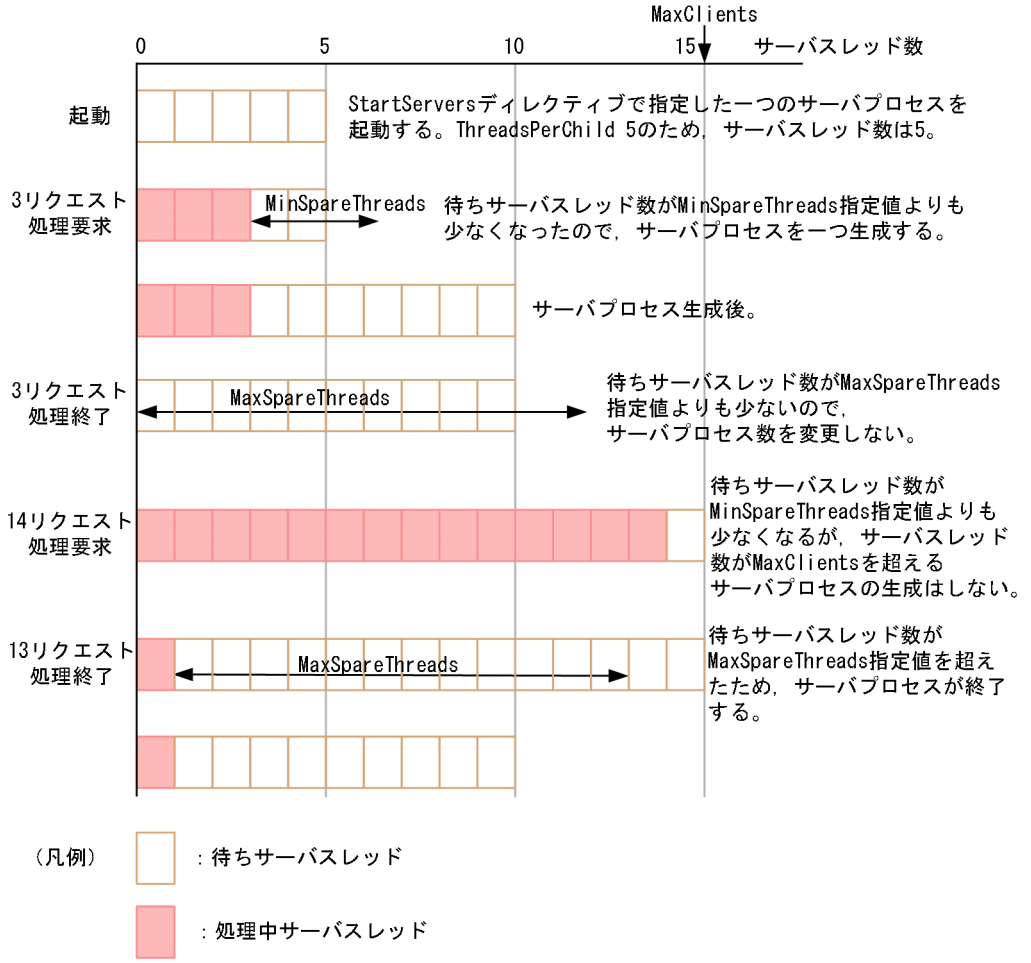
(b)　プロセス数・スレッド数の遷移例

プロセス数・スレッド数の遷移例を図 4-5 に示します。

- ディレクティブの指定値

```
LoadModule mpm_worker_module libexec/mod_mpm_worker.so
ServerLimit 3
StartServers 1
MinSpareThreads 4
MaxSpareThreads 12
ThreadsPerChild 5
MaxClients 15
```

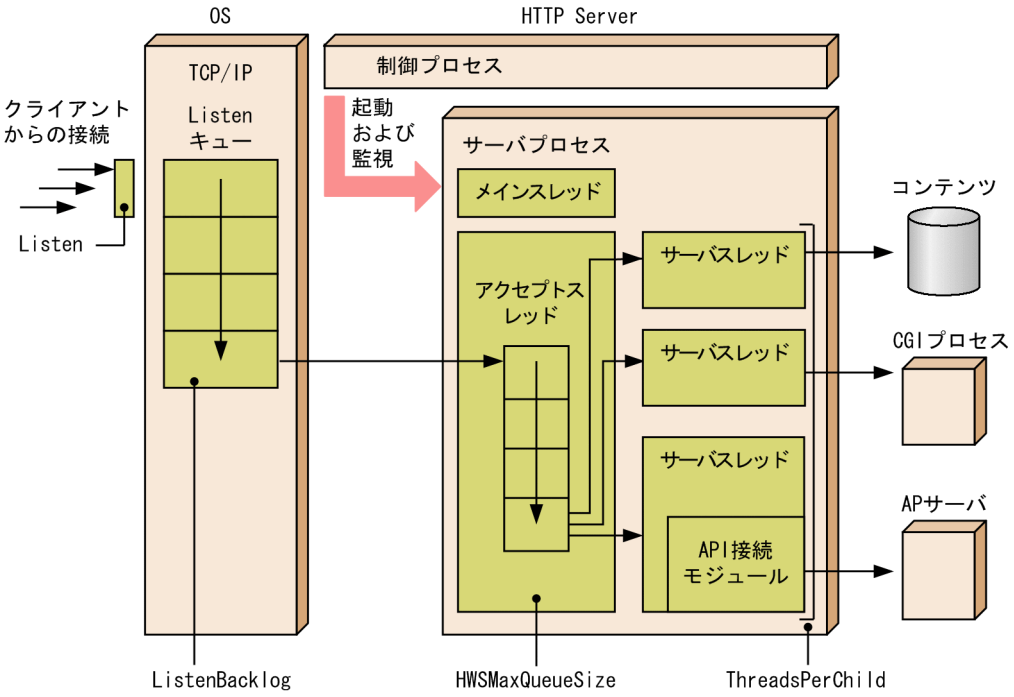
図 4-5　プロセス数・スレッド数の遷移例



4.1.3　HTTP Server のプロセス構造（Windows 版）

HTTP Server のプロセス構造を次に示します。

図 4-6　HTTP Server のプロセス構造（Windows 版）



HTTP Server を起動すると，制御プロセスが起動します。制御プロセスは，サーバプロセスを起動しその稼働を監視します。サーバプロセスが起動すると同時にメインスレッドが起動します。メインスレッドは，リクエストを受信するアクセプトスレッドを一つと，リクエストを処理するサーバスレッドを ThreadsPerChild ディレクティブで指定した個数だけ起動します。

クライアントからの TCP 接続は，Listen ディレクティブで指定した IP アドレスとポートから OS が受信し，OS 内の Listen キューに保留します。Listen キューのサイズは ListenBacklog ディレクティブで指定できます。Listen キューに格納できなかった TCP 接続は確立されません（このとき，HTTP Server はクライアントにステータスコードを返却しません）。アクセプトスレッドは Listen キューから TCP 接続を取り出し，HWSMaxQueueSize ディレクティブで指定した大きさのリクエストキューに登録します。それをサーバスレッドの一つが取り出して HTTP リクエストを受信し処理します。HWSMaxQueueSize ディレクティブ指定値を超えたためリクエストキューに格納できなかった TCP 接続は，アクセプトスレッドによって閉じられます（このとき，HTTP Server はクライアントにステータスコードを返却しません）。

サーバプロセス数およびサーバスレッド数が増減することはありません。

制御プロセスおよびサーバプロセスともに，プロセス名（実行プログラム名）は httpsd.exe です。制御プロセスのプロセス ID は，PidFile ディレクティブに指定したファイルに出力します。

HTTP/2 プロトコルを使用している場合は，サーバスレッドのほかに HTTP/2 プロトコルを処理するためのワーカスレッドが生成されます。詳細は「4.17.3　ワーカスレッド」を参照してください。

また，HTTP Server のリバースプロキシ機能を使用した場合，バックエンドサーバとの間にコネクションが確立されます。そのコネクションの最大数については，「4.7.4(5)　性能に関する注意事項」を参照してください。

4.1.4　稼働管理について

サーバプロセス（Windows 版の場合はサーバスレッド）の稼働状況を管理する上で必要となる，持続型接続の動作原理とタイマ監視機能について，説明します。

(1)　持続型接続（KeepAlive）

持続型接続（KeepAlive）は，クライアントからのリクエストに対してレスポンスを返したあとも TCP コネクションを切断しないで，同じクライアントからの次のリクエストを待つ機能です。

この機能は，KeepAlive ディレクティブに On を指定し，かつクライアント側が対応している場合に使用できます。TCP コネクションを切断しないため，クライアントが複数のリクエストを連続して送信する場合にはレスポンス時間を短縮できます。

次のリクエストを待つ間はサーバプロセスがクライアントに占有されますが，この待ち時間は KeepAliveTimeout ディレクティブで設定できます。また，1 クライアントが持続型接続を利用して何回までリクエストを処理できるかを MaxKeepAliveRequests ディレクティブで設定します。

なお，HTTP/2 通信では KeepAlive ディレクティブの設定に関係なく，コネクションが持続します。コネクションが持続する時間は，KeepAliveTimeout ディレクティブ値に従います。

(2)　タイマ監視

次の場合，Timeout ディレクティブに設定された値によって，タイマ監視ができます。

- クライアントからのリクエスト受信（コネクション確立後，HTTP プロトコルの受信）時
- クライアントへのレスポンス送信時
- CGI プログラムへのリクエスト送信時
- CGI プログラムへのリクエスト送信後からレスポンス受信まで
- CGI プログラムからのレスポンス受信時
- CGI プログラムからのレスポンス受信後，入出力用のパイプを閉じるまでの待ち時間
- リバースプロキシを使用している場合，バックエンドサーバへのリクエスト送信時
- リバースプロキシを使用している場合，バックエンドサーバへのリクエスト送信後からレスポンス受信まで
- リバースプロキシを使用している場合，バックエンドサーバからのレスポンス受信時

(3)　制御プロセスの監視

PidFile ディレクティブで指定したファイルに出力される ID のプロセスを監視すると，HTTP Server の制御プロセスを監視できます。監視するプロセス名（実行プログラム名）は Windows 版の場合 httpsd.exe，UNIX 版の場合 httpsd です。

制御プロセスを監視する際には，PidFile ディレクティブで指定したファイルに格納された ID のプロセスが HTTP Server のプロセスであることを必ず確認してください。HTTP Server のプロセスであることを確認するには，プロセスの実行プログラム名が，Windows 版の場合 httpsd.exe，UNIX 版の場合 httpsd であることを確認してください。

4.2　ログを採取する

出力されたメッセージの意味については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」の「21.　Web サーバ(HTTP Server)が出力するメッセージ」およびマニュアル「アプリケーションサーバ メッセージ(監査者用)」の「3.　KAWS（HTTP Server が出力する監査ログメッセージ)」を参照してください。

4.2.1　ログの種類

ログの種類を次に示します。

表 4-1　ログの種類

ログの種類	指定するディレクティブ	機能
アクセスログ	TransferLog	• デフォルトフォーマットのログを採取します。 • rotatelogs プログラムまたは rotatelogs2 プログラムを使用して，定期的または定量的に分割できます。 • LogFormat ディレクティブでフォーマットを変更できます。
	CustomLog	• カスタマイズしたフォーマットでログを採取します。 • rotatelogs プログラムまたは rotatelogs2 プログラムを使用して，定期的または定量的に分割できます。 • LogFormat ディレクティブで定義したフォーマットを CustomLog ディレクティブに指定できます。
エラーログ	ErrorLog	• エラー発生時のメッセージのログを採取します。 • rotatelogs プログラムまたは rotatelogs2 プログラムを使用して，定期的または定量的に分割できます。 • LogLevel ディレクティブで，採取するログのレベルを指定できます。 • HWSRequestLog ディレクティブを指定していない場合に，モジュールトレースを採取できます。 モジュールトレースの詳細については「4.2.6　モジュールトレースの採取」を参照してください。
	ScriptLog	• CGI スクリプトのエラーログを採取します。
リクエストログ	HWSRequestLog	• リクエストログを採取します。リクエストログとして，次のトレースを採取できます。 **モジュールトレース** モジュールの各関数の実行時および CGI プログラムの実行時に採取されるトレースです。モジュールトレースの詳細については「4.2.6　モジュールトレースの採取」を参照してください。 **リクエストトレース** リクエスト処理開始時や完了時などで採取されるトレースです。リクエストトレースの詳細については「4.2.7　リクエストトレースの採取」を参照してください。

ログの種類	指定するディレクティブ	機能
		I/O フィルタトレース モジュールが実装している入出力フィルタ関数の実行時に採取されるトレースです。I/O フィルタトレースの詳細については「4.2.8　I/O フィルタトレースの採取」を参照してください。 プロキシトレース リバースプロキシ機能を使用している場合に，リバースプロキシの処理状況を把握するためのトレースです。プロキシトレースの詳細については「4.2.9　プロキシトレースの採取」を参照してください。 • rotatelogs プログラムまたは rotatelogs2 プログラムを使用して，定期的または定量的に分割できます。
プロセス ID ログ	PidFile	• 制御プロセス ID のログを採取します。
イベントログ	なし	• サービスから起動する際に発生するエラーを記録します（Windows 版）。
内部トレース	HWSTraceLogFile	• 共有メモリのトレース情報を出力します。
共有メモリ ID ログ	HWSTraceIdFile	• 共有メモリ ID を格納します。
コアファイル※	CoreDumpDirectory	• HTTP Server 障害発生時のコアダンプの出力先を指定します（UNIX 版）。
WebSocket ログ	HWSWebSocketLog	• WebSocket 通信でのログを採取します。 • rotatelogs プログラムまたは rotatelogs2 プログラムを使用して，定期的または定量的に分割できます。

注※

OS でコアファイルを出力する設定を行った場合に出力されます。

設定方法については各 OS のマニュアルを参照してください。

アクセスログ，エラーログ，リクエストログ，WebSocket ログのサイズが 2GB を超えた場合，HTTP Server が異常終了したり，再起動できなかったりする場合があります。定期的にログファイルを退避するか，「4.2.3　ログを分割する（rotatelogs プログラム）」や「4.2.4　ログファイルをラップアラウンドさせて使用する（rotatelogs2 プログラム）」を参照して，ログファイルのサイズが 2GB を超えないように設定してください。

4.2.2　ログの採取方法

アクセスログ，エラーログ，プロセス ID のログおよびリクエストログの採取方法について説明します。

(1)　アクセスログ

(a)　デフォルトフォーマットのアクセスログ

TransferLog ディレクティブを指定して，ログを採取します。

デフォルトフォーマットのアクセスログの形式を次に示します。

```
クライアントホスト名△クライアントの識別情報△クライアントユーザ名△アクセス時刻△”リクエストライン”△ステータスコード△送信バイト数
```

（凡例）

△：空白

（出力例）

```
172.17.40.30 - - [25/Dec/2000:16:23:59 +0900] ”GET / HTTP/1.0” 200 3546
```

(b)　カスタムフォーマットのアクセスログ

CustomLog ディレクティブを指定して，ログを採取します。フォーマットの指定方法には，二つあります。

- 直接 CustomLog ディレクティブにフォーマットを指定する

  （例）

  ```
  CustomLog logs/access.log "%h %l %u %t ¥"%r¥" %>s %b"
  ```

- LogFormat ディレクティブでフォーマットに対するラベル名を定義して，そのラベル名を CustomLog ディレクティブに指定する

  （例）

  ```
  LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common
  CustomLog logs/access.log common
  ```

(2)　エラーログ

(a)　エラーメッセージログ

ErrorLog ディレクティブを指定して，ログを採取します。LogLevel ディレクティブで採取するエラーのレベルを指定します。

(b)　CGI スクリプトのエラーログ

ScriptLog ディレクティブを指定して，CGI スクリプトのエラーログを採取します。

(3)　プロセス ID のログ

PidFile ディレクティブを指定して，制御プロセス ID のログを採取します。

複数環境を生成する場合には，ファイルパスはほかの環境と競合しないように変更してください。

(4) リクエストログ

HWSRequestLog ディレクティブと HWSRequestLogType ディレクティブを指定して，リクエストログを採取します。リクエストログとは，モジュールトレース，リクエストトレースおよび I/O フィルタトレースの総称です。

モジュールトレースの詳細については「4.2.6　モジュールトレースの採取」，リクエストトレースの詳細については「4.2.7　リクエストトレースの採取」，I/O フィルタトレースの詳細については「4.2.8　I/O フィルタトレースの採取」を参照してください。

(5) WebSocket ログ

(a) 指定方法

HWSWebSocketLog ディレクティブを指定して，WebSocket 通信でのログを採取します。

rotatelogs2 で 512MB でファイルを分割する例を次に示します。

```
HWSWebSocketLog "|インストールパス/httpsd/sbin/rotatelogs2 ログ出力先/hws_websocket_log 524288 2"
```

HWSWebSocketLog ディレクティブを指定するには，mod_proxy_wstunnel モジュールの組み込みが必要です。mod_proxy_wstunnel モジュールの組み込み方法については「4.15.1　mod_proxy_wstunnel モジュール」を参照してください。mod_proxy_wstunnel モジュールを組み込んでいない場合は，起動エラーとなります。

(b) 出力フォーマット

各 WebSocket 通信で出力するログには，ルート AP 情報を含みます。ただし，ルート AP 情報が取得できない場合は，ルート AP 情報部分が"-"となります。出力するフォーマットを次に示します。

- クライアントからのデータをバックエンドに送信成功

  ```
  [時刻]△(ID)△クライアントIP:ポート△-->△WebサーバIP：ポート△-->△WebサーバIP：ポート△-->△バックIP:ポート△(ルートAP情報)
  ```

- バックエンドからのデータをクライアントに送信成功（初回）

  ```
  [時刻]△(ID)△クライアントIP:ポート△<--△WebサーバIP：ポート△<--△WebサーバIP：ポート△<--△バックIP:ポート△(ルートAP情報)(ステータスコード)
  ```

  ただし，ステータスコードを取得できない場合は，ステータスコード部分が"-1"となります。

- バックエンドからのデータをクライアントに送信成功（2 回目以降）

  ```
  [時刻]△(ID)△クライアントIP:ポート△<--△WebサーバIP：ポート△<--△WebサーバIP：ポート△<--△バックIP:ポート△(ルートAP情報)
  ```

- クライアントからのデータ受信でエラー

```
[時刻]△(ID)△クライアントIP:ポート△-X-△WebサーバIP：ポート△-->△WebサーバIP：ポート△
---△バックIP:ポート△(ルートAP情報)
```

- バックエンドへのデータ送信でエラー

```
[時刻]△(ID)△クライアントIP:ポート△-->△WebサーバIP：ポート△---△WebサーバIP：ポート△
-X-△バックIP:ポート△(ルートAP情報)
```

- バックエンドからのデータ受信でエラー

```
[時刻]△(ID)△クライアントIP:ポート△---△WebサーバIP：ポート△<--△WebサーバIP：ポート△
-X-△バックIP:ポート△(ルートAP情報)
```

- クライアントへのデータ送信でエラー

```
[時刻]△(ID)△クライアントIP:ポート△-X-△WebサーバIP：ポート△<--△WebサーバIP：ポート△
<--△バックIP:ポート△(ルートAP情報)
```

- バックエンドとのコネクション切断を検知

```
[時刻]△(ID)△クライアントIP:ポート△---△WebサーバIP：ポート△---△WebサーバIP：ポート△
-X-△バックIP:ポート△(ルートAP情報)
```

- クライアントとのコネクション切断を検知

```
[時刻]△(ID)△クライアントIP:ポート△-X-△WebサーバIP：ポート△---△WebサーバIP：ポート△
---△バックIP:ポート△(ルートAP情報)
```

(凡例)

△：空白

-->：クライアントからバックエンドサーバ方向へのデータの流れ

<--：バックエンドサーバからクライアント方向へのデータの流れ

-X-：データの送受信でエラー，またはコネクション切断

---：事象なし

(出力例)

```
[Mon Dec 09 21:13:09.008 2019] (9796) 192.168.10.10:61682 --> 192.168.10.20:80 --> 192.16
8.10.20:61683 --> 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:09.019 2019] (9796) 192.168.10.10:61682 <-- 192.168.10.20:80 <-- 192.16
8.10.20:61683 <-- 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)(101)
[Mon Dec 09 21:13:09.029 2019] (9796) 192.168.10.10:61682 <-- 192.168.10.20:80 <-- 192.16
8.10.20:61683 <-- 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:10.715 2019] (9796) 192.168.10.10:61682 --> 192.168.10.20:80 --> 192.16
8.10.20:61683 --> 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:10.727 2019] (9796) 192.168.10.10:61682 <-- 192.168.10.20:80 <-- 192.16
8.10.20:61683 <-- 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:12.227 2019] (9796) 192.168.10.10:61682 --> 192.168.10.20:80 --> 192.16
8.10.20:61683 --> 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:12.234 2019] (9796) 192.168.10.10:61682 <-- 192.168.10.20:80 <-- 192.16
8.10.20:61683 <-- 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
[Mon Dec 09 21:13:12.236 2019] (9796) 192.168.10.10:61682 --- 192.168.10.20:80 --- 192.16
8.10.20:61683 -X- 192.168.10.30:8008 (192.168.10.20/14756/0x0000000000000087)
```

(6) 各トレースの出力先

(a) モジュールトレースの出力先

モジュールトレースの出力先は，エラーログまたはリクエストログのどちらか一方になります。どちらに出力されるかは，ディレクティブの指定によって決まります。モジュールトレースの出力先と出力条件を次に示します。

表 4-2 モジュールトレースの出力先と出力条件

出力先	出力条件
リクエストログ	HWSRequestLog ディレクティブの指定があり，かつ，HWSRequestLogType ディレクティブに module-info または module-debug を指定した場合
エラーログ	HWSRequestLog ディレクティブの指定がなく，かつ，LogLevel ディレクティブに info または debug を指定した場合

モジュールトレースの詳細については「4.2.6 モジュールトレースの採取」を参照してください。

(b) リクエストトレースおよび I/O フィルタトレースの出力先

リクエストトレースおよび I/O フィルタトレースの出力先はリクエストログになります。

HWSRequestLog ディレクティブの指定があり，かつ，HWSRequestLogType ディレクティブが出力条件を満たしている場合にリクエストログに出力されます。HWSRequestLogType ディレクティブの出力条件については，「4.2.7 リクエストトレースの採取」および「4.2.8 I/O フィルタトレースの採取」を参照してください。

4.2.3 ログを分割する（rotatelogs プログラム）

アクセスログ，エラーログ，リクエストログを一定時間単位（例えば，24 時間ごと）に分割して，複数のファイルに出力できます。rotatelogs プログラムは次のディレクティブに指定できます。

- CustomLog ディレクティブ
- ErrorLog ディレクティブ
- HWSRequestLog ディレクティブ
- HWSWebSocketLog ディレクティブ
- TransferLog ディレクティブ

プログラムの指定方法を次に示します。

(1) 形式

```
rotatelogs 分割ログファイルのプリフィックス　ログ分割時間間隔 〔-fnum ファイル数〕〔-diff GMTに対する時差〕〔-opt use_mtime〕〔-opt date_suffix〕
```

(2) オペランド

- **分割ログファイルのプリフィックス**

  分割ログファイルのプリフィックスを絶対パスで指定します。

  「プリフィックス.nnnnnnnnnn」というファイルに，ログを採取します。

  nnnnnnnnnn：ログ採取開始時刻を表します。ログ採取時刻とは次の式で示す値です。

  ((1970 年 1 月 1 日の 0 時 0 分 0 秒(GMT：Greenwich Mean Time)を起点とした，ログを出力する時間の通算秒数÷ログ分割時間間隔）の小数点以下を切り捨てた値）×ログ分割時間間隔

- **ログ分割時間間隔**　～((1－31536000))

  一つのログファイルを採取する時間間隔を秒単位に指定します。指定した時間間隔を超えた最初のログ出力時，新規ファイルにログを採取します。ただし，-opt date_suffix 指定時は 86400 秒（＝ 24 時間）以上を指定してください。86400 秒よりも小さい値を指定した場合，同じ日付のファイルに出力し続けることがあります。

- **-fnum ファイル数**　～((1－256))

  分割したログファイルのファイル数を指定します。分割したファイル数がここで指定した数を超えた場合，最も古いファイルから削除されます。このオペランドを省略した場合，ファイルは削除されません。

- **-diff GMT に対する時差**　～((-1439－1439))

  ログファイルを分割する基準となる時間を，GMT に対する時差として分単位で指定します。指定しないまたは 0 を指定すると，1970 年 1 月 1 日 0 時 0 分 0 秒(GMT)が基準時間となります。GMT に対するローカルタイムの差が n 時間である場合に，ローカルタイムの m 時 0 分 0 秒を基準にする場合には，(n-m)×60 を指定します。JST の 0 時 0 分 0 秒を基準にする場合には，(+9-0)×60 で 540 を指定します。

- **-opt use_mtime**

  通常-fnum 指定時に最も古いファイルを選択する際，分割ログファイルの ctime の日時を使用します。このオプションを指定すると，分割ログファイルの ctime の日時の代わりに mtime の日時を使用します。ウィルススキャンソフトなどによって，分割ログファイルの ctime の日時が更新されることで最も古いファイルが正しく削除されなくなる場合にこのオプションを指定します。

- **-opt date_suffix**

  年月日を含む分割ログファイル名にする場合に指定します。

(3) 使用方法

ディレクティブに，"|プログラム名"の形式で指定して使用します。ログファイルを定期的に別ファイルに分割して採取します。

（例）Windows 版

24 時間ごとに，アクセスログを分割して<Application Server のインストールディレクトリ>¥httpsd¥logs¥access.nnnnnnnnnn ファイルに採取します。分割時間を日本時間に設定し，日本時間の毎 0 時にログファイルを分割する場合の指定を次に示します。

```
TransferLog "|¥"¥"<Application Serverのインストールディレクトリ>/httpsd/sbin/rotatelogs.exe¥" ¥"<Application Serverのインストールディレクトリ>/httpsd/logs/access¥" 86400 -diff 540¥""
```

- ログファイル名：<Application Server のインストールディレクトリ>¥httpsd¥logs¥access.nnnnnnnnnn
- ログ分割時間間隔：86400 秒（＝ 24 時間）

年月日を含むファイル名（access.YYYYMMDD）のファイルにアクセスログを採取する指定を次に示します。

```
TransferLog "|¥"¥"<Application Serverのインストールディレクトリ>/httpsd/sbin/rotatelogs.exe¥" ¥"<Application Serverのインストールディレクトリ>/httpsd/logs/access¥" 86400 -diff 540 -opt date_suffix¥""
```

- ログファイル名：<Application Server のインストールディレクトリ>¥httpsd¥logs¥access.YYYYMMDD

（例）UNIX 版

24 時間ごとに，アクセスログを分割して/opt/hitachi/httpsd/logs/access.nnnnnnnnnn ファイルに採取します。分割時間を日本時間に設定し，日本時間の毎 0 時にログファイルを分割する場合の指定を次に示します。

```
TransferLog "|/opt/hitachi/httpsd/sbin/rotatelogs /opt/hitachi/httpsd/logs/access 86400 -diff 540"
```

- ログファイル名：/opt/hitachi/httpsd/logs/access.nnnnnnnnnn
- ログ分割時間間隔：86400 秒（＝ 24 時間）

年月日を含むファイル名（access.YYYYMMDD）のファイルにアクセスログを採取する指定を次に示します。

```
TransferLog "|/opt/hitachi/httpsd/sbin/rotatelogs /opt/hitachi/httpsd/logs/access 86400 -diff 540 -opt date_suffix"
```

- ログファイル名：/opt/hitachi/httpsd/logs/access.YYYYMMDD

(4) 注意事項

(a) UNIX 版の注意事項

- rotatelogs プログラムは SIGTERM，SIGUSR1 および SIGHUP シグナルを受信してもプロセス終了処理を実施しませんが，制御プロセスとサーバプロセスが終了すればプロセス終了します。

(b) Windows 版の注意事項

- サービスとして起動した場合には，制御プロセスのログは採取されません。
- ログファイルは，そのファイルを開いているプロセスがある間は削除できません。このため，-fnum で指定した値より多いファイルが残ることがあります。例えば，制御プロセスがログを出力したファイルは，制御プロセスが終了するまで削除されません。
- 起動時に引数誤りなどがあった場合に Web サーバは起動しますが，rotatelogs プログラムは起動しません。その際，rotatelogs プログラムではイベントログに次に示す属性を持ったメッセージを出力します。
  - 種類：エラー
  - ソース：CosminexusHTTPServer
  - 分類：なし
  - イベント：3299
  - 説明：rotatelogs.exe: (メッセージ)
    出力されたメッセージの意味については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」の「21.7.4　rotatelogs プログラム」を参照してください。

  このメッセージが出力されていた場合，メッセージ内容に従って引数を見直したあとにサーバを再起動してください。なお，同じメッセージが複数回出力されることがあります。

(c) 共通の注意事項

- -fnum オペランドの指定によるログファイルの制御は Web サーバの再起動時に，ディレクトリ名またはログファイルのプリフィックスを変更すると，以前に採取したログファイルは削除されません。この場合は運用に応じて削除してください。
- Web サーバを起動または再起動してから，指定したログ分割間隔時間が経過した場合，分割したログファイルのプリフィックスに一致するファイルの数が-fnum オペランドの指定値を超えると，作成時間または更新時間（-opt use_mtime 指定時）の古いログファイルから削除されます。
- 分割ログファイルのプリフィックスは絶対パスで指定してください。
- TransferLog，CustomLog，ErrorLog，および HWSRequestLog ディレクティブの rotatelogs プログラムの引数指定に誤りがある場合，rotatelogs プログラムの起動に失敗しますが，Web サーバは起動します。この場合はログが出力されません。TransferLog，CustomLog，ErrorLog，および HWSRequestLog ディレクティブに rotatelogs プログラムを指定した場合は，ログファイルが作成され，意図した分割が実施されることを確認してください。

4.2.4 ログファイルをラップアラウンドさせて使用する（rotatelogs2 プログラム）

アクセスログ，エラーログ，リクエストログをログファイルサイズで分割して，複数のファイルにラップアラウンドして出力できます。rotatelogs2 プログラムは次のディレクティブに指定できます。

- CustomLog ディレクティブ
- ErrorLog ディレクティブ
- HWSRequestLog ディレクティブ
- HWSWebSocketLog ディレクティブ
- TransferLog ディレクティブ

プログラムの指定方法を次に示します。

(1) 形式

```
rotatelogs2 ログファイルプリフィックス名　ログファイルサイズ　ログファイル個数
```

(2) オペランド

- **ログファイルプリフィックス名**

  出力するログファイルのプリフィックス名を絶対パスで指定します。

  出力するログファイルは「プリフィックス.nnn」のファイル名となります。

  .nnn は.001 からログファイル個数で指定した値までです。

  「ログファイル個数」を nnn 面とすると，nnn 面のうち，HTTP Server 起動時の更新時刻が最新のものが，カレントのログファイルとなります。ログファイルは，ファイル名称のプリフィックスに拡張子.001～.nnn を付けて区別します。カレントのログファイルの拡張子が.mmm であった場合，カレントのログファイルがいっぱいになると，続きは，.mmm+1 のログファイルをクリアして出力されます。.mmm が.nnn と一致した場合，続きは.001 に出力されます。

  Windows 版の場合，「プリフィックス.index」のインデクス番号格納用ファイルが作成されます。このファイルは.nnn 管理用ファイルであり，rotatelogs2 プログラムの起動時に作成され，停止時に削除されます。ただし，起動エラーの一部などで削除されないことがありますが，以降の Web サーバの動作に影響はありません。

- **ログファイルサイズ**　～((1－2097151))

  ログファイルの最大サイズ（単位：KB）を指定します。

  ログを出力するタイミングで，最大サイズを超えていると，次のログファイルをクリアして続きが出力されます。

- **ログファイル個数**　～((1－256))

  出力するログファイルの最大数を指定します。

最大サイズを超えて次のログファイルに移る場合，それまで処理していたログファイルの拡張子が最大個数と同じとき，再度.001 のファイルから使用します。

(3) 使用方法

ディレクティブに，"|プログラム名"の形式で指定して使用します。

(例) 4,096KB ごとにエラーログを最大 5 個採取する場合

```
ErrorLog ”|¥”¥”<Application Serverのインストールディレクトリ>/httpsd/sbin/rotatelogs2.exe ¥” ¥”<Application Serverのインストールディレクトリ>/httpsd/logs/errorlog¥” 4096 5¥””
```

errorlog.001～errorlog.005 の順番にログが出力されます。errorlog.005 が 4,096KB を超えると errorlog.001 をクリアして続きが出力されます。HTTP Server 起動時に，すでにこれらのログファイルがある場合には，更新時刻の最も新しいログファイルが出力対象のログファイルとなります。このログファイルのサイズがすでに 4,096KB を超えている場合には，次のログファイルをクリアして続きが出力されます。4,096KB を超えない場合は，このファイルの続きに出力されます。

(4) 注意事項

(a) UNIX 版の注意事項

- rotatelogs2 プログラムは SIGTERM，SIGUSR1 および SIGHUP シグナルを受信してもプロセス終了処理を実施しませんが，制御プロセスとサーバプロセスが終了すればプロセス終了します。

(b) Windows 版の注意事項

- Web サーバはサービスとして起動してください。サービスとして起動しない場合，Web サーバの停止または再起動の際に，不当にログファイルがクリアされることがあります。
- インデクス番号格納用ファイルは，rotatelogs2 プログラムが動作中の間は絶対に編集や削除をしないでください。編集するとログが正しく出力されないことがあります。
- Web サーバの起動時にインデクス番号格納用ファイル「プリフィックス.index」と同名のファイルが存在した場合，ファイルは上書きされます。
- 起動時に引数誤りなどがあった場合に Web サーバは起動しますが，rotatelogs2 プログラムは起動しません。その際，rotatelogs2 プログラムではイベントログに次に示す属性を持ったメッセージを出力します。
  - 種類：エラー
  - ソース：CosminexusHTTPServer
  - 分類：なし
  - イベント：3299
  - 説明：rotatelogs2.exe: (メッセージ)

出力されたメッセージの意味については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」の「21.7.5　rotatelogs2 プログラム」を参照してください。

このメッセージが出力されていた場合，メッセージ内容に従って引数を見直したあとにサーバを再起動してください。なお，同じメッセージが複数回出力されることがあります。

(c)　共通の注意事項

- ログファイルプリフィックス名は絶対パスで指定してください。
- HTTP Server 起動時の出力ログファイルは，更新日時が最新のものを対象とするため，誤ってファイルを更新した場合は正しいファイルへの出力ができなくなります。
- ログファイルサイズには，同一秒内に複数のファイルが指定サイズを超えるような小さいサイズを指定しないでください。このようなサイズを指定した場合には，最も新しいログファイル以外が出力対象となり正しくローテーションされなくなることがあります。
- コンフィグファイル内に，同一のログファイルプリフィックス名を複数個所で指定しないでください。複数個所で指定した場合には，最も新しいログファイル以外が出力対象となり正しくローテーションされなくなることがあります。
- TransferLog，CustomLog，ErrorLog，および HWSRequestLog ディレクティブの rotatelogs2 プログラムの引数指定に誤りがある場合，rotatelogs2 プログラムの起動に失敗しますが，Web サーバは起動します。この場合はログが出力されません。TransferLog，CustomLog，ErrorLog，および HWSRequestLog ディレクティブに rotatelogs2 プログラムを指定した場合は，ログファイルが作成され，意図した分割が実施されることを確認してください。

4.2.5　ログファイルの IP アドレスをホスト名に変換する（logresolve コマンド）

logresolve コマンドは，レコードの先頭が IP アドレスであるアクセスログファイル内の IP アドレスをホスト名に変換し，新規ログファイルに出力します。変換規則は，ホスト名のルックアップの逆引きによります。

(1)　形式

```
logresolve 〔-s ファイル名〕 〔-c〕 < アクセスログファイル名 > 新ログファイル名
```

(2)　オペランド

- -s ファイル名

  変換したときの情報を出力するファイルを指定します。このファイルには次のような情報が出力されます。

  - ホスト名の数

- 異なる IP アドレスの数
- 変換できなかった IP アドレスの数

- -c
  変換後のホスト名が変換前の IP アドレスと一致するかどうかチェックする場合に指定します。
- アクセスログファイル名
  入力ログファイル名を指定します。入力したファイルの IP アドレスからホスト名のルックアップの逆引きをします。レコードの先頭は，必ず IP アドレスでなければなりません。ホスト名の検索に失敗した場合，新ログファイルには IP アドレスが出力されます。
- 新ログファイル名
  IP アドレスをホスト名に変換したアクセスログを出力するファイル名を指定します。

(3) 使用方法

logs¥access.log に格納しているアクセスログ内の IP アドレスをホスト名に変換します。

アクセスログファイル：logs¥access.log

新ログファイル：logs¥new_access.log

```
logresolve < logs¥access.log > logs¥new_access.log
```

4.2.6 モジュールトレースの採取

Web サーバは複数のモジュール※から構成され，これらのモジュールは特定のタイミングで実行される複数の関数から構成されています。**モジュールトレース**とは，モジュールの各関数の実行時および CGI プログラムの実行時に採取されるトレースのことです。モジュールトレースは，HWSRequestLog ディレクティブの指定の有無によって，採取先などの採取方法が変わります。

注※　モジュールには，Web サーバに LoadModule ディレクティブで動的に組み込んで使用する外部モジュールと，httpsd 実行ファイルに含まれる内部モジュールとがあります。

(1) トレース対象

モジュールトレースのトレース対象を次に示します。

表 4-3　モジュールトレースのトレース対象

トレース対象	トレースの契機
モジュール	モジュールは複数の関数から構成されています。これらの関数は，初期化処理とリクエスト対応処理に分類されます。トレースはリクエスト対応処理についての関数に対して採取します。
CGI プログラム	CGI プログラム実行時にトレースを採取します。

(2) 採取方法

HWSRequestLog ディレクティブを指定していない場合は，ErrorLog ディレクティブに指定したファイルに対して，LogLevel ディレクティブの指定に従って採取します。

HWSRequestLog ディレクティブを指定している場合は，HWSRequestLog ディレクティブに指定したファイルに対して，HWSRequestLogType ディレクティブの指定に従って採取します。

注意事項

ErrorLog ディレクティブに指定したファイルに対してログを採取する場合は，バーチャルホスト単位にファイルを分けることができます。HWSRequestLog ディレクティブに指定したファイルに対してログを採取する場合は，バーチャルホスト単位にファイルを分けることができません。

(3) 採取レベル

LogLevel ディレクティブまたは HWSRequestLogType ディレクティブの指定によって，採取するモジュールトレースのレベルを変更できます。各レベルで採取するトレースの内容を次に示します。

(a) info レベル

障害発生の原因となるおそれのある外部モジュールおよび CGI プログラムについて採取します。

LogLevel ディレクティブに info を指定または HWSRequestLogType ディレクティブに module-info を指定した場合に採取します。

(b) debug レベル

info レベルのほかに，リクエストごとに動作する内部モジュールについてのトレースも採取します。

LogLevel ディレクティブに debug を指定または HWSRequestLogType ディレクティブに module-debug を指定した場合に採取します。

(4) トレースフォーマット

モジュールトレースの出力項目は次のとおりです。

なお，以降の記述で「サーバプロセス ID」となっている部分は UNIX 版で prefork MPM を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:サーバスレッド ID」，Windows 版の場合「サーバスレッド ID」です。

(a) モジュール

- info レベルで出力される場合

  コール時

  ```
  [時刻]△(サーバプロセスID)△[info]△hws△:△module△-->△(モジュールファイル名称[関数オフセット])
  ```

リターン時

```
[時刻]△(サーバプロセスID)△[info]△hws△:△module△<--△(モジュールファイル名称[関数オフセット])(結果コード)
```

(凡例)

△：空白

関数オフセットと関数の対応を次に示します。

表 4-4　関数オフセットと関数の対応

関数オフセット	関数名	意味
[0]	create_request	リクエスト開始処理を実行中
[1]	post_read_request	リクエスト読み込み後処理を実行中
[2]	quick_handler	リクエストされた URL 変換前処理を実行中
[3]	translate_name	リクエストされた URL から，実際のファイル名への変換処理実行中
[4]	map_to_storage	ディスクアクセスを伴わないリクエスト処理を実行中
[5]	header_parser	リクエストヘッダ解析処理実行中
[6]	access_checker	認証済みユーザからリクエストされた URL に対してホスト名および IP アドレスによるアクセス権限チェック実行中
[7]	check_user_id	ユーザ ID のチェック処理実行中
[8]	auth_checker	認証済みユーザからリクエストされた URL に対してアクセス権限（Require）チェック実行中
[9]	type_checker	MIME タイプのチェック処理実行中
[10]	fixups	リクエスト実行前処理を実行中
[11]	insert_filter	フィルタ挿入処理を実行中
[12]	handler	ハンドラを実行中
[13]	insert_error_filter	エラー応答前処理を実行中
[14]	log_transaction	ログ出力処理を実行中
[15]	error_log	エラーログ出力後処理を実行中
[16]	get_suexec_identity	ユーザ情報取得処理を実行中
[17]	pre_read_request	リクエスト読み込み前処理を実行中
[18]	access_checker_ex	追加のアクセス権限チェック実行中

(出力例)

```
[Fri Jul 15 17:29:43 2005] (1864) [info] hws : module --> (mod_example.c[1])
[Fri Jul 15 17:29:43 2005] (1864) [info] hws : module <-- (mod_example.c[1])(-1)
```

- debug レベルで出力される場合

コール時

```
[時刻]△(サーバプロセスID)△[debug]△ファイル名称(行番号):△hws△:△module△-->△(モジュールファイル名称[関数オフセット])
```

リターン時

```
[時刻]△(サーバプロセスID)△[debug]△ファイル名称(行番号):△hws△:△module△<--△(モジュールファイル名称[関数オフセット])(結果コード)
```

(凡例)

△：空白

(出力例)

```
[Fri Jul 15 17:29:43 2005] (1864) [debug] request.c(69): hws : module --> (mod_alias.c[3])
[Fri Jul 15 17:29:43 2005] (1864) [debug] request.c(69): hws : module <-- (mod_alias.c[3])(-1)
```

(b) CGI プログラム

- info レベルで出力される場合

コール時

```
[時刻]△(サーバプロセスID)△[info]△hws△:△cgi△-->△(exec=cgiファイル名称)(argv0=実行プログラム名称)(args=引数 ※)(CGIプロセスID)
```

注※　args による引数は，GET /cgi-bin/isindex?aaa+bbb+ccc HTTP/1.0 のように，＝ではなく，＋で連結されたクエリーが指定された場合にだけ表示します。

リターン時

```
[時刻]△(サーバプロセスID)△[info]△hws△:△cgi△<--△(exec=cgiファイル名称)(argv0=実行プログラム名称)(CGIプロセスID)
```

(凡例)

△：空白

(出力例)

```
[Fri Jul 15 19:48:08 2012] (1784) [info] hws : cgi --> (exec=<Application Serverのインストールディレクトリ>/httpsd/cgi-bin/isindex)(argv0=isindex)(args=aaa+bbb+ccc)(1144)
[Fri Jul 15 19:48:08 2012] (1784) [info] hws : cgi <-- (exec=<Application Serverのインストールディレクトリ>/httpsd/cgi-bin/isindex)(argv0=isindex)(1144)
```

(5) 使用方法

(a) 使用例

リクエストログに，info レベルのモジュールトレースおよびリクエストトレースを出力する例を示します。

```
HWSRequestLogType module-info request
HWSRequestLog logs/hwsrequest.log
```

4.2.7　リクエストトレースの採取

リクエストトレースとは，次のときに採取されるトレースのことです。

HTTP/1.1 以前のリクエスト処理の場合

- リクエスト処理開始時
- リクエスト処理完了時
- KeepAlive 接続の場合，次のリクエストラインの受信完了時
- リクエスト処理開始からリクエストライン受信完了前のコネクション切断時

HTTP/2 のリクエスト処理の場合

- サーバスレッドによる HTTP/2 プロトコルの初回リクエスト受付時
- ワーカスレッドによる HTTP/2 プロトコルのリクエスト処理開始時
- サーバスレッドによる HTTP/2 プロトコルのレスポンス送信時
- サーバスレッドによる HTTP/2 プロトコルの通信終了時

HWSRequestLog ディレクティブが指定されていた場合，かつ HWSRequestLogType ディレクティブで request が指定された場合にリクエストトレースの採取が有効となります。障害発生時に Web サーバにリクエストが届いているかどうかを確認する場合などに有用です。

(1)　トレースフォーマット

(a)　HTTP/1.1 以前の場合のリクエストトレースの出力項目

HTTP/1.1 以前の場合のリクエストトレースの出力項目は次のとおりです。

なお，以降の記述で「サーバプロセス ID」となっている部分は UNIX 版で prefork MPM を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:サーバスレッド ID」，Windows 版の場合「サーバスレッド ID」です。

- **リクエスト処理開始時**

```
[時刻]△(サーバプロセスID)△client△:△hws△-->△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[A])
```

- **リクエスト処理完了時**

```
[時刻]△(サーバプロセスID)△client△:△hws△<--△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[R])
```

- KeepAlive 接続での次のリクエストライン受信完了時

```
[時刻]△(サーバプロセスID)△client△:△hws△-->△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[K])
```

- リクエスト処理開始からリクエストライン受信完了前のコネクション切断時

```
[時刻]△(サーバプロセスID)△client△:△hws△<--△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[X])
```

(凡例)

△：空白

(出力例)

```
[Tue Nov 21 15:18:40 2006] (1716) client : hws --> (192.168.2.1:5245,192.168.1.1:80[A])
[Tue Nov 21 15:18:41 2006] (1716) client : hws <-- (192.168.2.1:5245,192.168.1.1:80[R])
```

(b) HTTP/2 の場合のリクエストトレースの出力項目

HTTP/2 の場合のリクエストトレースの出力項目は次のとおりです。

なお，以降の記述で「サーバスレッド ID」となっている部分は Windows 版を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:サーバスレッド ID」です。

また，次の「ワーカスレッドによる HTTP/2 プロトコルのリクエスト処理開始時」の「ワーカスレッド ID」となっている部分も Windows 版を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:ワーカスレッド ID」です。

- サーバスレッドによる HTTP/2 プロトコルの初回リクエスト受付時

```
[時刻]△(サーバスレッドID)△client△:△hws△-->△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[A])
```

- ワーカスレッドによる HTTP/2 プロトコルのリクエスト処理開始時

```
[時刻]△(ワーカスレッドID)△client△:△hws△-->△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[S])
```

- サーバスレッドによる HTTP/2 プロトコルのレスポンス送信時

```
[時刻]△(サーバスレッドID)※△client△:△hws△<--△(クライアントIPアドレス:ポート番号,
サーバIPアドレス:ポート番号[R])(ワーカスレッドID)
```

注※

プロトコルエラーなどで HTTP/2 通信が終了している場合には，（サーバスレッド ID）が（ワーカスレッド ID）になることがあります。

- サーバスレッドによる HTTP/2 プロトコルの通信終了時

```
[時刻]△(サーバスレッドID)△client△:△hws△<--△(クライアントIPアドレス:ポート番号,サーバIPアドレス:ポート番号[X])
```

(凡例)

△：空白

(出力例)

```
[Wed Oct 19 10:28:48.142 2022] (1304) client : hws --> (127.0.0.1:53327,127.0.0.1:443[A])
[Wed Oct 19 10:28:48.145 2022] (10860) client : hws --> (127.0.0.1:53327,127.0.0.1:443[S]
)
[Wed Oct 19 10:28:48.145 2022] (1304) client : hws <-- (127.0.0.1:53327,127.0.0.1:443[R])
(10860)
[Wed Oct 19 10:28:48.175 2022] (10860) client : hws --> (127.0.0.1:53327,127.0.0.1:443[S]
)
[Wed Oct 19 10:28:48.175 2022] (1304) client : hws <-- (127.0.0.1:53327,127.0.0.1:443[R])
(10860)
[Wed Oct 19 10:28:51.184 2022] (1304) client : hws <-- (127.0.0.1:53327,127.0.0.1:443[X])
```

4.2.8　I/O フィルタトレースの採取

I/O フィルタトレースとは，モジュールが実装している入出力フィルタ関数の実行時に採取されるトレースのことです。

HWSRequestLog ディレクティブが指定されていた場合，かつ HWSRequestLogType ディレクティブで filter が指定された場合に有効となります。モジュール内のフィルタで発生した障害を切り分ける場合などに有用です。ただし，出力量が多いため，デバッグ目的以外では使用しないでください。

(1)　トレースフォーマット

I/O フィルタトレースの出力項目は次のとおりです。

なお，以降の記述で「サーバプロセス ID」となっている部分は UNIX 版で prefork MPM を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:サーバスレッド ID」，Windows 版の場合「サーバスレッド ID」です。

- 入力フィルタコール時

```
[時刻]△(サーバプロセスID)△hws△:△in-filter※△-->△(フィルタ名称[フィルタタイプ番号])
```

- 入力フィルタリターン時

```
[時刻]△(サーバプロセスID)△hws△:△in-filter※△<--△(フィルタ名称[フィルタタイプ番号])
(戻り値)
```

(凡例)

△：空白

注※　出力フィルタの場合は，「out-filter」になります。

(出力例)

```
[Tue Nov 21 15:18:40 2006] (1716) hws : in-filter --> (core_in[60])
[Tue Nov 21 15:18:40 2006] (1716) hws : in-filter <-- (core_in[60])(0)
```

4.2.9 プロキシトレースの採取

プロキシトレースとは，リバースプロキシ機能を使用している場合に採取される，次のトレースのことです。

- バックエンドサーバへの接続処理開始時
- バックエンドサーバへの接続処理終了時
- バックエンドサーバへのリクエストの転送処理開始時
- バックエンドサーバからのレスポンス受信処理開始時

HWSRequestLog ディレクティブが指定されている場合，かつ HWSRequestLogType ディレクティブで proxy を指定した場合に有効となります。バックエンドサーバへリクエストを転送する場合に，処理状況の把握に有用です。

(1) トレースフォーマット

プロキシトレースの出力項目は次のとおりです。

なお，以降の記述で「サーバプロセス ID」となっている部分は UNIX 版で prefork MPM を使用している場合のフォーマットであり，UNIX 版の worker MPM では「サーバプロセス ID:サーバスレッド ID」，Windows 版の場合「サーバスレッド ID」です。

- 接続処理開始時

  ```
  [時刻]△(サーバプロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△connect
  ```

- 接続処理終了時（接続成功時）

  ```
  [時刻]△(サーバプロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△connect(0)
  ```

- 接続処理終了時（接続失敗時）

  ```
  [時刻]△(サーバプロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△connect(X)
  ```

- リクエスト転送処理開始時

  HTTP/1.1 以前の場合

  ```
  [時刻]△(サーバプロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△req△send(タイプ)(Content-Lengthヘッダ値)
  ```

  HTTP/2 の場合

```
[時刻]△(サーバプロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△req△send
```

タイプは次の 3 種類です。

CHUNK：チャンク形式で転送する方式

CL：Content-Length サイズ分転送を繰り返す方式

SPOOL：一時ファイルにボディを保存して，まとめて転送する方式

Content-Length ヘッダ値が存在しない場合には，"-"を出力します。

- レスポンス受信処理開始時

```
[時刻]△(プロセスID)△(ルートアプリケーション情報)△proxy△接続先IPアドレス:接続先ポート番号△:△res△recv
```

(凡例)

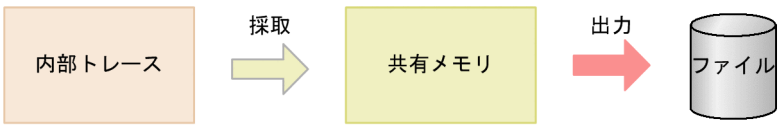
△：空白

(出力例)

```
[Thu Aug 29 11:03:58.639 2013] (21999) (192.168.0.33/250/0x0000000000000001) proxy 192.168.1.5:80 : connect
[Thu Aug 29 11:03:58.639 2013] (21999) (192.168.0.33/250/0x0000000000000001) proxy 192.168.1.5:80 : connect(0)
[Thu Aug 29 11:03:58.639 2013] (21999) (192.168.0.33/250/0x0000000000000001) proxy 192.168.1.5:80 : req send(CL)(4096)
[Thu Aug 29 11:03:58.682 2013] (21999) (192.168.0.33/250/0x0000000000000001) proxy 192.168.1.5:80 : res recv
```

4.2.10　内部トレースの採取（hwstraceinfo コマンド）

アプリケーションプログラムの実行時やリクエスト受け取り時など，システムで発生した事象は内部トレースとして採取されています。内部トレースは，共有メモリにいったん出力され，その後ディレクティブの指定やコマンドによって，ファイルに出力されます。



(1)　トレース情報の採取

Web サーバの各種事象発生を契機に内部トレースが共有メモリに採取されます。共有メモリのメモリ識別子は，HWSTraceIdFile ディレクティブに指定したファイルに格納されます。

(2) ファイルへの出力方法

共有メモリに採取された内部トレースは，サーバプロセスの異常終了時または hwstraceinfo コマンドの実行によって，ファイルに出力されます。サーバプロセスが異常終了した場合は，HWSTraceLogFile ディレクティブで指定したファイルに出力されます。

hwstraceinfo コマンドでは，共有メモリのメモリ識別子，出力先のファイル名を指定します。hwstraceinfo コマンドは，UNIX 版の場合，User ディレクティブで指定したユーザまたはスーパーユーザだけが実行できます。また，Windows 版の場合，管理者権限を持つユーザだけが実行できます。

内部トレースの出力ファイルサイズは次のとおりです。

UNIX 版の場合

prefork MPM を使用している場合

ps -efl コマンドの出力サイズ＋ vmstat コマンドの出力サイズ＋ ipcs -a コマンドの出力サイズ＋ 7KB×MaxClient 値

worker MPM を使用している場合

ps -efl コマンドの出力サイズ＋ vmstat コマンドの出力サイズ＋ ipcs -a コマンドの出力サイズ＋ 2KB×総サーバスレッド数(ServerLimit 値×ThreadLimit 値)

Windows 版の場合

7KB×ThreadPerChild 値

(3) hwstraceinfo コマンド

hwstraceinfo コマンドの指定方法を説明します。

(a) 形式

```
hwstraceinfo -i 共有メモリ識別子 {-l ファイル名|-r}
```

(b) オペランド

- **-i 共有メモリ識別子**

  HWSTraceIdFile ディレクティブで指定したファイルに出力されている共有メモリ識別子を指定します。

- **-l ファイル名**

  -i で指定した共有メモリ識別子に該当するトレースを出力するファイルを指定します。

- **-r**

  -i で指定した共有メモリ識別子に割り当てられている共有メモリを解放します。UNIX 版では，Web サーバが終了してもトレース用の共有メモリは残ります。残った共有メモリを解放するためにこのオペランドを使用します。Windows 版では，Web サーバ終了時にトレース用の共有メモリは解放されますので，このオペランドは提供していません。

(c) 使用例

共有メモリ識別子 1800_1133780652_0 に該当するトレースを traceinfo.log ファイルに出力する例を示します。

```
hwstraceinfo -i 1800_1133780652_0 -l traceinfo.log
```

(4) 共有メモリの解放および再起動時の注意（UNIX 版の場合）

Web サーバが終了しても，トレース情報を残すために共有メモリは解放しません。また，Web サーバを再起動する場合は，共有メモリが再利用されます。

Web サーバを停止したあとに起動した場合は，HWSTraceIdFile ディレクティブに指定したファイルの値を基に，いったん共有メモリを解放して，再度確保します。ただし，次のような場合は，以前使用していた共有メモリが解放できなくなりますので，注意してください。

- 同一ユーザで再起動していない（User ディレクティブまたは Group ディレクティブの値が変更されている）
- HWSTraceIdFile ディレクティブの値を変更している
- HWSTraceIdFile ディレクティブで指定していたファイルが消去されている

共有メモリを解放する場合は，-r を指定した hwstraceinfo コマンドを実行してください。

4.2.11 保守情報収集機能（hwscollect コマンド）

Web サーバが異常終了および無応答となった場合などに，保守員が障害調査を実施するためのコアダンプ，エラーログ，アクセスログなどの資料が必要となります。hwscollect コマンドによって，これら障害調査のための資料を一括して収集できます。hwscollect コマンドは UNIX 版だけで有効です。

hwscollect コマンドは，root 権限で実行する必要があります。

(1) 形式

```
hwscollect 収集情報出力先ディレクトリ 〔-f 定義ファイル名〕
```

(2) オペランド

- **収集情報出力先ディレクトリ**

  収集した情報を tar のアーカイブファイルとして出力する場合の，出力先のディレクトリを指定します。

  アーカイブファイルの名称は，HWSyyyymmddhhmmss.tar となります。ここで yyyymmdd は hwscollect を起動した日付，hhmmss は hwscollect を起動した 24 時間制の時刻で，それぞれローカルタイムです。

- -f 定義ファイル名

  hwscollect.conf ファイルを指定します。絶対パスまたはカレントディレクトリからの相対パスで指定します。

(3) 使用方法

HTTP Server を標準的な構成でインストールした場合の使用方法を示します。

```
/opt/hitachi/httpsd/maintenance/hwscollect /tmp
```

(4) コンフィグファイルの設定

hwscollect.conf で hwscollect の動作を定義します。hwscollect.conf は，キーワードと値をスペースで区切って記述します。キーワードは，大文字小文字を区別しません。行の最初に#を付けるとコメント行になります。ファイル名はすべて絶対パスで指定してください。コンフィグファイルのキーワードと指定について次に示します。

表 4-5 コンフィグファイルのキーワードと指定

キーワード	指定する値	指定	複数指定	ワイルドカード
ServerRoot	httpsd.conf の ServerRoot ディレクティブの値を指定します。	必須	×	×
conf	httpsd.conf のファイル名を指定します。	必須	×	×
trcinfo	hwstraceinfo コマンドの存在するディレクトリを指定します。	必須	×	×
trcid	httpsd.conf の HWSTraceIdFile ディレクティブで指定されるファイル名を指定します。	必須	×	×
PidFile	httpsd.conf の PidFile ディレクティブで指定されるファイル名を指定します。	必須	×	×
CORE	httpsd.conf の CoreDumpDirectory ディレクティブの値と，SSLCacheServerRunDir ディレクティブの値を指定します。	任意	○	○
LOG	httpsd.conf の ErrorLog ディレクティブ，HWSRequestLog ディレクティブおよび TransferLog ディレクティブや CustomLog ディレクティブなどのログを指定するファイル名を指定します。	任意	○	○
FILES	そのほか，障害解析に役立つファイルがあれば指定します。	任意	○	○

(凡例)

○：指定できる。

×：指定できない。

(5) コンフィグファイルの指定例

コンフィグファイルの指定例を示します。

```
ServerRoot /opt/hitachi/httpsd
conf /opt/hitachi/httpsd/conf/httpsd.conf
trcinfo /opt/hitachi/httpsd/sbin/
trcid /opt/hitachi/httpsd/logs/hws.trcid
PidFile /opt/hitachi/httpsd/logs/httpd.pid
CORE /opt/hitachi/httpsd/logs/core*
LOG /opt/hitachi/httpsd/logs/error*
LOG /opt/hitachi/httpsd/logs/access*
LOG /opt/hitachi/httpsd/logs/hws.trclog*
LOG /opt/hitachi/httpsd/logs/hwsrequest*
```

(6) ディスク使用量

- 一時的に使用するファイル

  200KB+7KB×MaxClients 値
- 収集した情報を出力する tar ファイル

  core ファイルの容量＋ log ファイルの容量＋一時的に使用するファイルの容量

(7) 注意事項

- 収集情報出力先ディレクトリには，core ファイルを含む保守情報のアーカイブファイルが作成されるため，空き領域を確保してください。
- 収集情報出力先ディレクトリに出力ファイルおよび一時ファイルを作成します。このため，収集情報出力先ディレクトリは書き込み可能としてください。
- CORE，LOG および FILES にディレクトリを指定すると，指定されたディレクトリ下のファイルのすべてを採取します。このため，ルートディレクトリなど上位ディレクトリを指定すると，大量かつ不要な情報を採取してしまうため，注意が必要です。

4.3　サーバマシンのバーチャル化（バーチャルホスト）

バーチャルホストは 1 台のサーバマシンを複数台のマシンに見せます。その方法は次に示す二つがあります。

- サーバ名に基づくバーチャルホスト（Name-Based Virtual Hosts）
- IP アドレスに基づくバーチャルホスト（IP-Based Virtual Hosts）

4.3.1　サーバ名に基づくバーチャルホスト

サーバ名に基づくバーチャルホストは，一つの IP アドレスに対して複数のホスト名を DNS サーバなどで定義しておき，クライアントからそのホスト名でアクセスすることで，複数ホストのように見せます。ネットワークインタフェースを複数設定する必要はありません。サーバ名に基づくバーチャルホストでは，SSL 対応ホストと非 SSL 対応ホストの組み合わせ，または異なる複数の SSL 対応ホストの組み合わせでは構築できません。それらの組み合わせで構築する場合は，IP アドレスに基づくバーチャルホストで構築してください。サーバ名に基づくバーチャルホストの場合には，<VirtualHost>ディレクティブに IP アドレスを明示的に指定してください。

（例） 1 台のサーバマシン（IP アドレス：172.17.40.10）上の一つの Web サーバでポートを一つオープンし，Web ブラウザからのリクエストに応じてホストを切り替える運用をする。

Web ブラウザからの要求が，http://www1.xxx.soft.hitachi.co.jp/の場合

<Application Server のインストールディレクトリ>/httpsd/htdocs1/index.html（DirectoryIndex の指定が index.html の場合）を参照します。

Web ブラウザからの要求が，http://www3.xxx.soft.hitachi.co.jp/の場合

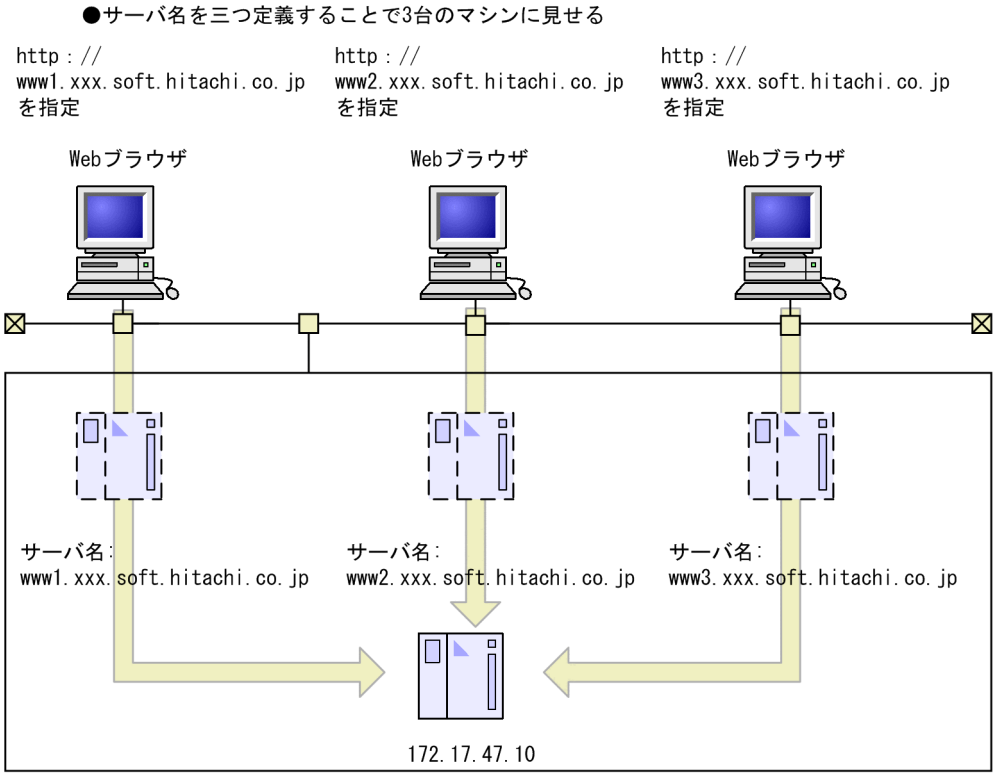
<Application Server のインストールディレクトリ>/httpsd/htdocs3/index.html（DirectoryIndex の指定が index.html の場合）を参照します。

ただし，この方法は Web ブラウザからのリクエスト中の Host ヘッダで，"Host: www1.xxx.soft.hitachi.co.jp"のようにホスト名（必要に応じてポート番号）を指定してきた場合だけ利用できます。古い Web ブラウザや，簡易タイプの Web ブラウザでは利用できないことがあるので注意が必要です。その場合，最も上位に記述された<VirtualHost>ブロックの指定（この例では www1.xxx.soft.hitachi.co.jp）が有効になります。

```
Listen 80                                                    …1.
<VirtualHost 172.17.40.10>                                   …2.
DocumentRoot ”<Application Serverのインストールディレクトリ>/httpsd/htdocs1”  …3.
ServerName www1.xxx.soft.hitachi.co.jp                       …4.
</VirtualHost>
<VirtualHost 172.17.40.10>                                   …5.
DocumentRoot ”<Application Serverのインストールディレクトリ>/httpsd/htdocs2”  …6.
ServerName www2.xxx.soft.hitachi.co.jp                       …7.
</VirtualHost>
<VirtualHost 172.17.40.10>                                   …8.
DocumentRoot ”<Application Serverのインストールディレクトリ>/httpsd/htdocs3”  …9.
ServerName www3.xxx.soft.hitachi.co.jp                       …10.
</VirtualHost>
```

1. ポート番号は一つ
2. バーチャルホスト 1 の定義
3. ルートディレクトリの定義
4. サーバ名 1 の定義
5. バーチャルホスト 2 の定義
6. ルートディレクトリの定義
7. サーバ名 2 の定義
8. バーチャルホスト 3 の定義
9. ルートディレクトリの定義
10. サーバ名 3 の定義

注　www1.xxx.soft.hitachi.co.jp，www2.xxx.soft.hitachi.co.jp，www3.xxx.soft.hitachi.co.jp は，DNS サーバなどに 172.17.40.10 ホストのホスト名として登録されていなければなりません。



4.3.2　IP アドレスに基づくバーチャルホスト

IP アドレスに基づくバーチャルホストは次の三つの方法でクライアントには複数ホストのように見せます。

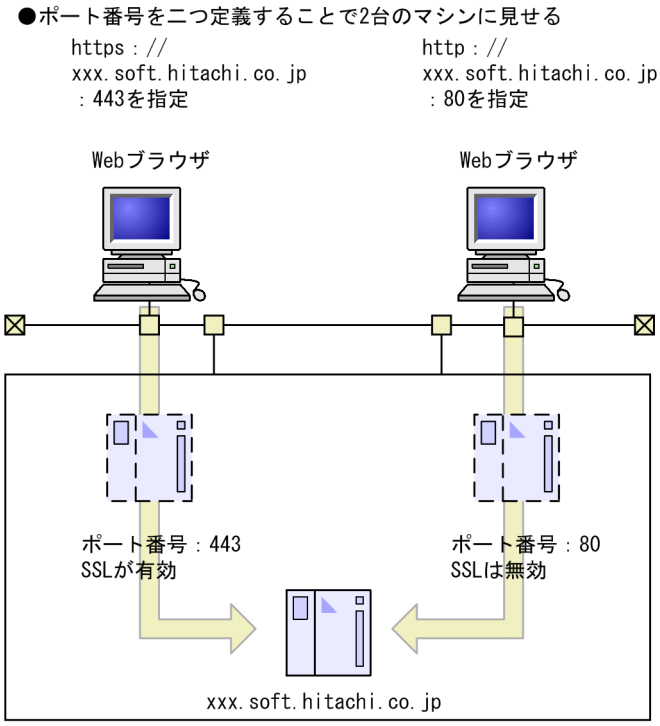
- 複数のポートを使用
- 1 台のサーバマシンに複数のネットワークインタフェースを指定

- IP アドレスのエイリアスを指定

(例 1) 1 台のサーバマシン上の一つの Web サーバでポートを二つオープンし，SSL 対応 Web サーバと非対応 Web サーバの二つのホストとして運用する。

```
Listen 443                                          …1
Listen 80                                           …2
<VirtualHost xxx.soft.hitachi.co.jp:443>            …3
    DocumentRoot "<Application Serverのインストールディレクトリ>/httpsd/ssldocs"
    SSLEngine On                                    …4
    SSLCertificateFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/se
rver/httpsd.pem"
    SSLCertificateKeyFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl
/server/httpsdkey.pem"
</VirtualHost>
<VirtualHost xxx.soft.hitachi.co.jp:80>             …5
    DocumentRoot "<Application Serverのインストールディレクトリ>/httpsd/htdocs"
    SSLEngine Off                                   …6
</VirtualHost>
```

1. ポート番号の定義
2. ポート番号の定義
3. ポート番号 443 のバーチャルホストの定義
4. SSL 有効
5. ポート番号 80 のバーチャルホストの定義
6. SSL 無効



(例 2) 1 台のサーバマシン上に二つの NIC（Network Interface Card）(IP アドレス:172.17.40.10, 172.17.40.20）を備え，一つの Web サーバで Web ブラウザからのリクエストに応じてホストを切り替えて運用する。

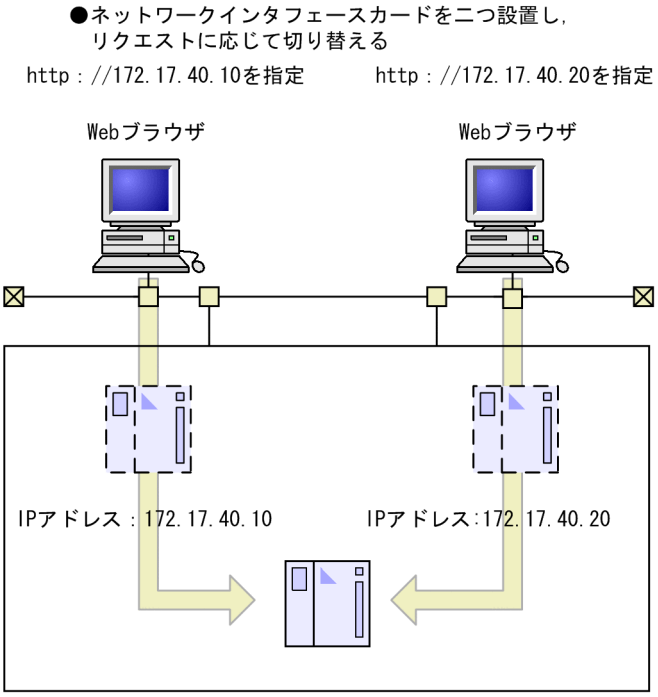
Web ブラウザからのリクエストが，http://172.17.40.10/の場合

<Application Server のインストールディレクトリ>/httpsd/htdocs1/index.html (DirectoryIndex の指定が index.html の場合）を参照します。

Web ブラウザからのリクエストが，http://172.17.40.20/の場合

<Application Server のインストールディレクトリ>/httpsd/htdocs2/index.html (DirectoryIndex の指定が index.html の場合）を参照します。

```
Listen 80
<VirtualHost 172.17.40.10>
DocumentRoot "<Application Serverのインストールディレクトリ>/httpsd/htdocs1"
ServerName www10.xxx.soft.hitachi.co.jp
</VirtualHost>
<VirtualHost 172.17.40.20>
DocumentRoot "<Application Serverのインストールディレクトリ>/httpsd/htdocs2"
ServerName www20.xxx.soft.hitachi.co.jp
</VirtualHost>
```



4.4 Web サーバでの CGI プログラムの実行

CGI プログラムとは，Web サーバ上で動作するプログラムです。この CGI プログラムを使用すれば，静的な HTML へのアクセスだけでは実現できないインタラクティブな Web アクセスができます。

4.4.1 CGI プログラムの定義

CGI プログラムを実行するには，ScriptAlias ディレクティブで CGI プログラムがあるディレクトリを指定する方式，AddHandler ディレクティブを使用しファイル拡張子に cgi-script ハンドラを指定する方式および SetHandler ディレクティブで cgi-script ハンドラを指定する方式があります。

httpsd.conf で設定する場合は，CGI プログラムの管理のしやすさの点で，ScriptAlias ディレクティブによる設定を推奨します。

(1) ScriptAlias ディレクティブの指定例

CGI プログラムのパス名を<Application Server のインストールディレクトリ>/httpsd/cgi-bin/CGI プログラムファイル名とし，これに対してクライアントから/cgi-bin/CGI プログラムファイル名でアクセスする場合

```
ScriptAlias /cgi-bin/ "<Application Serverのインストールディレクトリ>/httpsd/cgi-bin/"
```

(2) AddHandler ディレクティブの指定例

- ファイル拡張子.cgi に cgi-script ハンドラを指定する場合

```
AddHandler cgi-script .cgi
```

なお，Options ディレクティブで ExecCGI オプションの設定が必要です。

(3) SetHandler ディレクティブの指定例

- script で始まるファイル名に対するリクエストに対して，cgi-script ハンドラを指定する場合

```
<FilesMatch ^script>
  SetHandler cgi-script
  Options ExecCGI
</FilesMatch>
```

4.4.2 CGI プログラムの呼び出し

CGI プログラムは Web ブラウザから次の形式の URL を指定して呼び出します。

```
http://ホスト名〔:ポート番号〕/パス名〔?問い合わせ文字列〕
```

- **ホスト名〔:ポート番号〕**

  Web サーバが起動しているホスト名または IP アドレスと，ポート番号を指定します。ポート番号を省略すると，ポート番号 80 にリクエストを送信します。

- **パス名**

  パス名は CGI プログラムのパスを指定します。

- **問い合わせ文字列**

  CGI プログラムに渡すパラメタです。そのキーワードと値の組を指定します。Web ブラウザのフォームにデータを記述した場合，リクエストラインに自動的に設定されます。

4.4.3 CGI プログラムに渡す情報

Web サーバから CGI プログラムに環境変数を渡します。詳細は「付録 B CGI プログラムに渡す環境変数」を参照してください。

4.4.4 CGI プログラムの例

CGI プログラムのサンプルプログラムと，その実行例を説明します。

サンプル CGI プログラム

Windows 版で使用可能なサンプルプログラムのソース例を次に示します。これは Perl 言語で書かれたプログラムで，ファイル名を test-cgi.pl とします。

```
#! c:¥bin¥perl.exe

$argc=$#ARGV+1;
print "Content-Type: text/plain¥n";
print "¥n";
print "argc is $argc. argv is ¥"@ARGV¥".¥n";
print "SERVER_SOFTWARE = $ENV{'SERVER_SOFTWARE'}¥n";
print "SERVER_NAME = $ENV{'SERVER_NAME'}¥n";
print "GATEWAY_INTERFACE = $ENV{'GATEWAY_INTERFACE'}¥n";
print "SERVER_PROTOCOL = $ENV{'SERVER_PROTOCOL'}¥n";
print "SERVER_PORT = $ENV{'SERVER_PORT'}¥n";
print "REQUEST_METHOD = $ENV{'REQUEST_METHOD'}¥n";
print "HTTP_ACCEPT = ¥"$ENV{'HTTP_ACCEPT'}¥"¥n";
print "PATH_INFO = ¥"$ENV{'PATH_INFO'}¥"¥n";
print "PATH_TRANSLATED = ¥"$ENV{'PATH_TRANSLATED'}¥"¥n";
print "SCRIPT_NAME = ¥"$ENV{'SCRIPT_NAME'}¥"¥n";
print "QUERY_STRING = ¥"$ENV{'QUERY_STRING'}¥"¥n";
print "REMOTE_HOST = $ENV{'REMOTE_HOST'}¥n";
print "REMOTE_ADDR = $ENV{'REMOTE_ADDR'}¥n";
print "REMOTE_USER = $ENV{'REMOTE_USER'}¥n";
print "AUTH_TYPE = $ENV{'AUTH_TYPE'}¥n";
```

```
print "CONTENT_TYPE = $ENV{'CONTENT_TYPE'}¥n";
print "CONTENT_LENGTH = $ENV{'CONTENT_LENGTH'}¥n";
```

CGI プログラムの実行

Web ブラウザに次に示すように指定して，サンプル CGI プログラムを呼び出します。

```
http://www.example.com/cgi-bin/test-cgi.pl/ABC?X=1&Y=2
```

サンプルプログラムの実行結果

```
argc is 0. argv is "".
SERVER_SOFTWARE = Cosminexus HTTP Server
SERVER_NAME = www.example.com
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = "*/*"
PATH_INFO = "/ABC"
PATH_TRANSLATED = "C:¥Program Files¥HITACHI¥Cosminexus¥httpsd¥htdocs¥ABC"
SCRIPT_NAME = "/cgi-bin/test-cgi.pl"
QUERY_STRING = "X=1&Y=2"
REMOTE_HOST =
REMOTE_ADDR = xxx.xxx.xxx.xxx
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

4.4.5 CGI プログラムに渡す追加情報

CGI/1.1 の環境変数以外に Web サーバから CGI プログラムに情報を渡す場合の指定方法について説明します。

コンフィグファイルに CGI プログラムに渡す環境変数や，その値を指定できます。CGI プログラムに渡さない環境変数の指定もできます。

PassEnv 環境変数	CGIプログラムに渡す環境変数の指定
SetEnv 環境変数 値	CGIプログラムに渡す環境変数とその値の指定
UnsetEnv 環境変数	CGIプログラムに渡さない環境変数の指定

4.4.6 環境変数の定義

クライアントのリクエストを基に，環境変数を定義できます。リクエストしているクライアントのホスト名や IP アドレスなどを基に環境変数を定義したり，環境変数の設定を解除したりできます。

```
SetEnvIfNoCase Request_URI "¥.(gif|jpg)$" request_is_image
```

この場合，ファイル拡張子が.gif または.jpg のとき（このディレクティブの場合，大文字，小文字の区別はしません），request_is_image という環境変数を CGI プログラムに渡します。

4.4.7　Windows で CGI プログラムを利用するときの注意事項

(1)　CGI プログラム作成時の注意

CGI プログラムとサーバスレッド間のデータ送受信には，CGI プログラムの標準入力，標準出力，標準エラー出力を使用しています。データ送受信中には Timeout ディレクティブは有効になります。CGI プログラム作成時には，データの送受信完了後は，標準入出力などを閉じるかまたは終了してください。

(2)　CGI プログラムの強制終了

CGI プログラムは Web サーバが停止しても，CGI プログラム自身が処理を終えるまで終了しません。CGI プログラムを強制終了するには「タスクマネージャ」から終了させます。

4.4.8　UNIX 版で CGI プログラムを利用するときの注意事項

(1)　実行権限

CGI プログラムには，User，Group ディレクティブ指定値での実行権限が必要です。

(2)　性能

worker MPM ではサーバスレッド数に比例して，CGI プロセスの生成処理にコストが掛かり性能が劣化します。そのため，CGI プログラムを実行する場合には，prefork MPM を使用することを推奨します。

4.4.9　パス情報指定時の注意事項

リクエスト URL に，CGI プログラムに渡すパス情報が指定された場合，そのパス情報を環境変数 PATH_INFO に，パス情報をファイルシステム上のパスに変換した値を環境変数 PATH_TRANSLATED に設定します。パス情報をファイルシステム上のパスに変換する際には，DocumentRoot ディレクティブに指定されたパスを基点とします。パス情報に対し，Alias ディレクティブなどで別名を指定している場合は，その指定に従って変換します。

Web サーバの設定によって，環境変数 PATH_TRANSLATED に設定されたパスへのアクセスを許可していない場合には，エラーログにアクセス拒否のメッセージを出力します。このメッセージを出力した場合でも，Web サーバは CGI プログラムを実行し，リクエスト処理を続行します。この際，環境変数 PATH_TRANSLATED も CGI プログラムに渡されます。

(例) ドキュメントルートが "C:/Program Files/Hitachi/Cosminexus/httpsd"，Web ブラウザからの要求が "http://www.example.com/cgi-bin/test-cgi.pl/ABC"（CGI プログラム"test-cgi.pl"を実行するリクエストに，パス情報として "/ABC" を付加）の場合の，エラーログ出力例を次に示します。

```
[Thu Apr 02 15:33:10 2020] [error] [pid 11111:tid 12345] [client 192.168.1.1:45678] AH017
97: client denied by server configuration: C:/Program Files/Hitachi/Cosminexus/httpsd/ABC
```

4.5　ユーザ認証とアクセス制御

Web サーバに対するアクセス制御方法には次に示す方法があります。

- ユーザ名およびパスワードによるアクセス制御
- クライアントのホスト名または IP アドレスによるアクセス制御
- ディレクトリに対するアクセス制御
- ディレクトリサービスを利用したアクセス制御

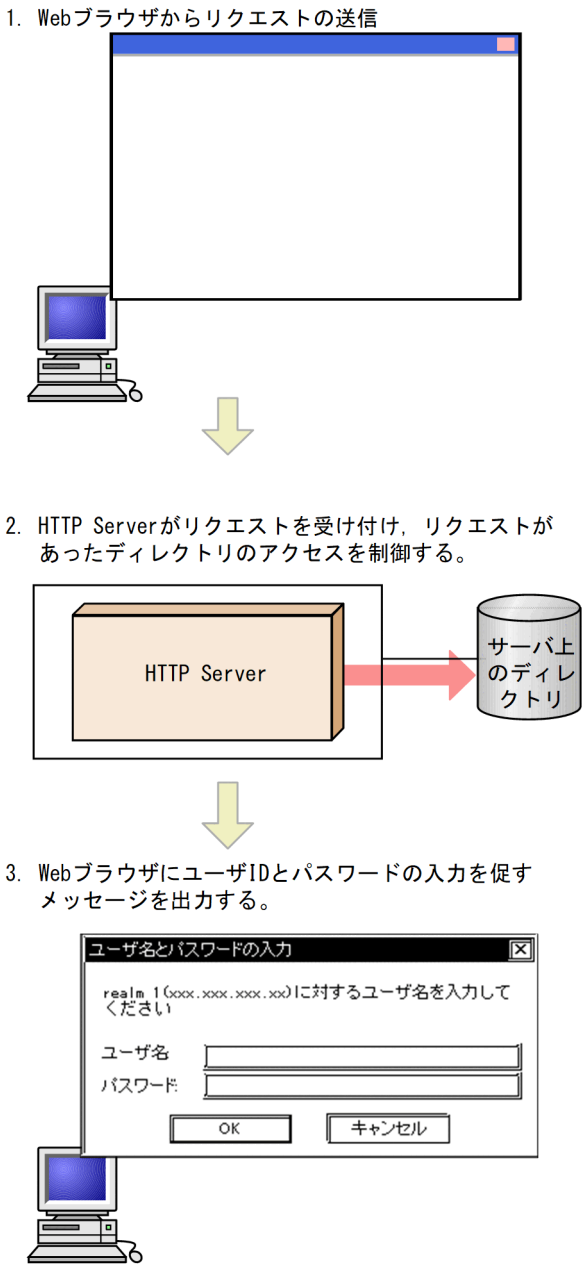
4.5.1　ユーザ名およびパスワードによるアクセス制御

ユーザ名とそのパスワードは htpasswd コマンドを使用して，パスワードファイルに登録します。登録されているユーザ名に対して，ホスト内のディレクトリやファイルなどのアクセス権を定義できます。htpasswd コマンドの使用方法については，「4.5.1(1)　ユーザ名とパスワードのパスワードファイルへの登録およびパスワードの変更」を参照してください。

（例）<Application Server のインストールディレクトリ>¥httpsd¥htdocs¥ディレクトリ下を特定のユーザだけに公開する

htpasswd コマンドを使用してあらかじめユーザ名とパスワードをパスワードファイル（<Application Server のインストールディレクトリ>¥httpsd¥htdocs¥.htpasswd）に登録しておいてください。
httpsd.conf ファイルに次に示すディレクティブを設定します。ユーザが<Application Server のインストールディレクトリ>¥httpsd¥htdocs¥にアクセスすると Web サーバはステータスコード 401 Authorization Required を応答し，Web ブラウザでユーザ名およびパスワードの入力を要求します。

```
<Directory ”<Application Serverのインストールディレクトリ>/httpsd/htdocs”>
    AuthType Basic
    AuthName ”realm 1”
    AuthUserFile ”<Application Serverのインストールディレクトリ>/httpsd/htdocs/.htpasswd”
    Require valid-user
</Directory>
```



(1) ユーザ名とパスワードのパスワードファイルへの登録およびパスワードの変更

htpasswd コマンドを使用して，パスワードファイルにユーザ名，パスワードの登録および変更ができます。

htpasswd コマンドの使用方法について次に説明します。

(a) 形式

```
htpasswd 〔-b〕 〔-c | -D〕 パスワードファイル名 ユーザ名〔パスワード〕
```

(b) オペランド

- -b
  パスワードをコマンドラインに指定する場合に指定します。
- -c
  新規にパスワードファイルを作成する場合に指定します。すでに作成しているパスワードファイルにユーザを追加する場合や，パスワードを変更する場合には，指定する必要はありません。
- -D
  ユーザの登録を削除する場合に指定します。指定したパスワードファイルに，指定したユーザが登録されている場合に，パスワードファイルから該当するユーザを削除します。
- パスワードファイル名
  パスワードを登録，変更または削除するパスワードファイルを指定します。
- ユーザ名
  パスワードを登録，変更または削除するユーザ名を指定します。
- パスワード
  登録または変更するパスワードを指定します。-b オプションを指定したときだけ指定できます。

(c) 使用方法

パスワードファイル名と，登録するユーザ名またはパスワードを変更するユーザ名を指定して htpasswd を起動すると，そのユーザのパスワードの入力が要求されます。入力確認を含め，2 回パスワードを入力すると，パスワードファイルにそのユーザのユーザ名と，パスワードが登録されます。

```
 C:¥>"<Application Serverのインストールディレクトリ>¥httpsd¥bin¥htpasswd.exe" .htpasswd user
xx     …1.
 New password:                                                                  …2.
 Re-type new password:                                                          …3.
 Updating password for userxxx                                                  …4.
 C:¥>
```

1. userxx のパスワードの変更
2. 新パスワード入力
3. 新パスワード再入力
4. 新パスワードの登録終了

登録を削除する場合は，-D オプション，パスワードファイル名および削除するユーザ名を指定して htpasswd を起動します。

```
 C:¥>"<Application Serverのインストールディレクトリ>¥httpsd¥bin¥htpasswd.exe" -D .htpasswd u
serxx  …1.
 Deleting password for userxx                                                   …2
.
 C:¥>
```

1. userxx の登録削除
2. userxx の登録削除終了

(d) 注意事項

- ユーザ名の最大長とパスワードの最大長は 128 文字です。
- htpasswd コマンド実行時は，パスワードファイルの作成先と同じディレクトリに，作業ファイルが一時的に作成されます。作業ファイル名は，「パスワードファイル名.プロセス ID」です。この作業ファイルは，htpasswd コマンドの終了時に自動的に削除されます。ただし，実行中にキャンセルした場合など，作業ファイルが削除されないことがあります。作業ファイルが残っている場合は，手動で削除してください。

4.5.2 クライアントのホスト名または IP アドレスによるアクセス制御

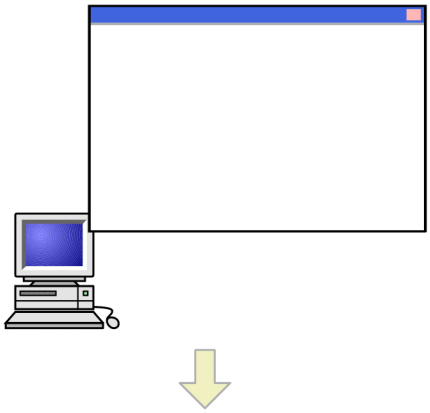
クライアントのホスト名や IP アドレスによってアクセス制御するには，Allow from ディレクティブや Deny from ディレクティブを使用します。Allow from ディレクティブでアクセスを許可するホスト，Deny from ディレクティブでアクセスを禁止するホストを指定します。

(例) <Application Server のインストールディレクトリ>¥httpsd¥htdocs¥ディレクトリ下を，プロキシ経由のリクエストによる参照を禁止する

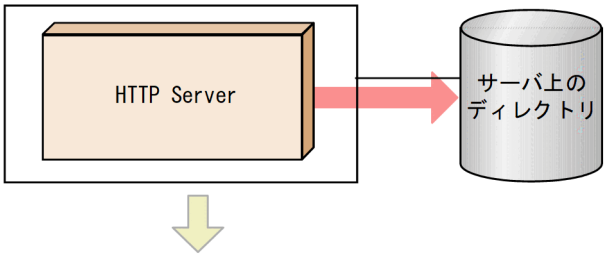
httpsd.conf ファイルに次に示すディレクティブを設定します。ユーザが<Application Server のインストールディレクトリ>¥httpsd¥htdocs¥にアクセスする場合，proxy.xxx.soft.hitachi.co.jp をプロキシとして利用している Web ブラウザはステータスコード 403 Forbidden でアクセスを拒否されます。

```
<Directory "<Application Serverのインストールディレクトリ>/httpsd/htdocs">    ディレクトリの定義
    Order deny,allow                              アクセス許可と禁止の優先順位定義
    Deny from proxy.xxx.soft.hitachi.co.jp  アクセスの禁止
</Directory>
```

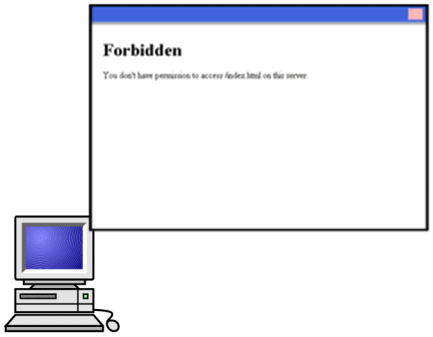
1. Webブラウザからリクエストの送信



2. HTTP Serverがリクエストを受け付け，リクエストが
あったディレクトリのアクセスを制御する。



3. proxy.xxx.soft.hitachi.co.jpをプロキシとして利用して
いるブラウザはステータスコード403 Forbiddenでアクセス
を拒否される。



4.5.3　ディレクトリに対するアクセス制御

アクセスコントロールファイル（.htaccess）を特定のディレクトリ下に作成すれば，そのディレクトリに対するアクセス権を設定できます。そのファイルにアクセスを許可または拒否するクライアント名（IP アドレス）やユーザ名を指定します。

(1)　アクセスコントロールファイル

アクセスコントロールファイルを特定のディレクトリ下に作成すれば，そのディレクトリに対するアクセス権を設定できます。アクセスコントロールファイルの名称は，AccessFileName ディレクティブで指定します。デフォルトは.htaccess です。

アクセスコントロールファイルによるアクセス制御は，Web サーバを再起動することなく，有効になります。ただし，正しく機能させるためには，httpsd.conf の AllowOverride ディレクティブを適切な上書き許可レベルに設定する必要があります。

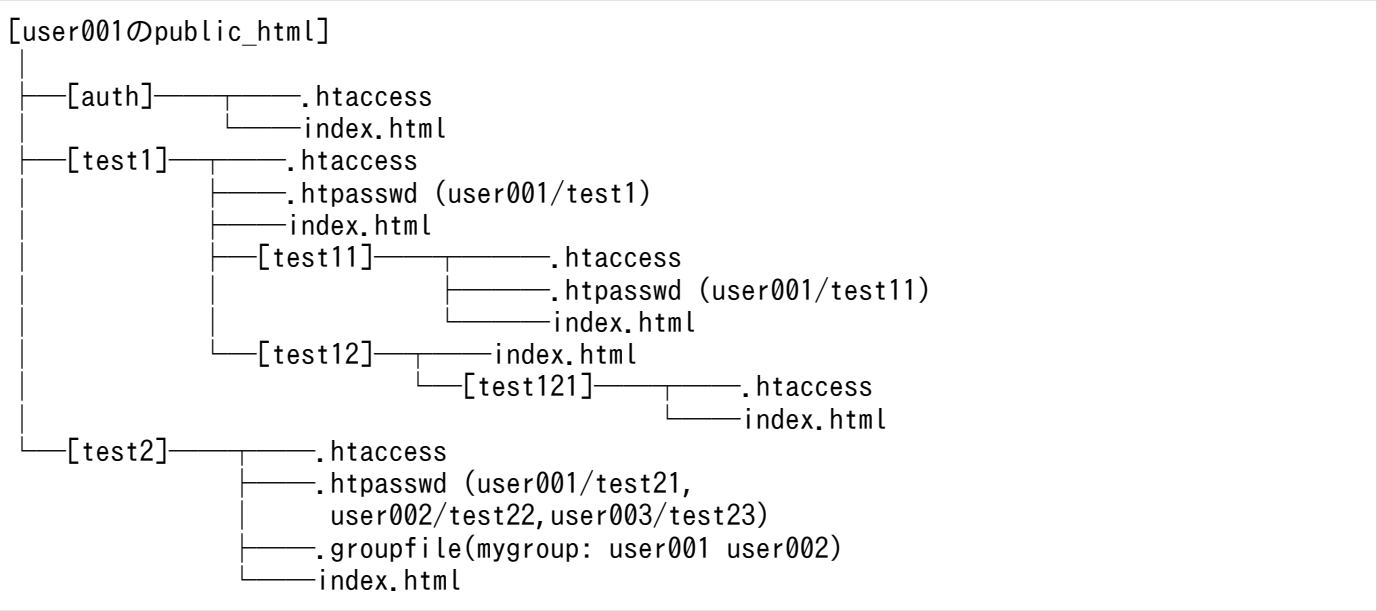
アクセスコントロールファイルにパスワードファイルを指定すると，ユーザがそのディレクトリにアクセスする場合にユーザ名およびパスワードの入力を要求します。

注意事項

アクセスコントロールファイル（.htaccess）とパスワードファイル（.htpasswd）は 1 対 1 である必要はありません。異なるアクセスコントロールファイルの AuthUserFile ディレクティブに同じパスワードファイルを指定できます。

(2) アクセス権の設定例

次のようなディレクトリ構成で，各ディレクトリに対してアクセスコントロールファイルにアクセス権を設定します。



- auth ディレクトリ下のアクセス権の定義（auth/.htaccess ファイル）

  IP アドレスが 172.18.102.11 および 172.16.202.4 のサーバからのアクセスを拒否します。

```
Order deny,allow                        …1.
Deny from 172.18.102.11 172.16.202.4  …2.
```

  1. アクセス拒否の定義を先に評価
  2. アクセス拒否の定義

- test1 ディレクトリ下のアクセス権の定義（test1/.htaccess ファイル）

  ユーザ名=user001，パスワード=test1 を入力した場合だけ，test1/index.html および test1/test12/index.html へのアクセスを許可します。

```
AuthUserFile C:/user001/public_html/test1/.htpasswd      …1.
AuthName ”test1 Directory”                               …2.
AuthType Basic
<Limit GET POST>                                         …3.
   Require user user001                                  …4.
</Limit>
```

1. パスワードファイルの定義

   パスワードファイルに登録しているユーザ名とパスワード

   ユーザ名：user001，パスワード：test1

2. realm 名の定義

3. メソッドに対する定義

4. ユーザ名：user001 のアクセスを許可

- test1/test11 ディレクトリ下のアクセス権の定義（test1/test11/.htaccess ファイル）

  ユーザ名=user001，パスワード=test11 を入力した場合だけ，test1/test11/index.html へのアクセスを許可します。

```
AuthUserFile C:/user001/public_html/test1/test11/.htpasswd      …1.
AuthName ”test11 Directory”                                     …2.
AuthType Basic
<Limit GET POST>                                                …3.
   Require user user001                                         …4.
</Limit>
```

1. パスワードファイルの定義

   パスワードファイルに登録しているユーザ名とパスワード

   ユーザ名：user001，パスワード：test11

2. realm 名の定義

3. メソッドに対する定義

4. ユーザ名：user001 のアクセスを許可

- test1/test12/test121 ディレクトリ下のアクセス権の定義（test1/test12/test121/.htaccess ファイル）

  ユーザ名=user001，パスワード=test1 を入力し，Web ブラウザが MSIE の場合だけ，test1/test12/test121/index.html へのアクセスを許可します。

```
Order deny,allow      …1.
Allow from env=MSIE   …2.
Deny from all         …3.
```

1. アクセス拒否の定義を先に評価

2. Web ブラウザが MSIE の場合，アクセスを許可

3. すべてのホストからのアクセスを拒否

ただし，httpsd.conf に次のディレクティブを定義しているものとします。

```
SetEnvIf User-Agent ".*MSIE.*" MSIE
```

- test2 ディレクトリ下のアクセス権の定義（test2/.htaccess ファイル）

  mygroup グループのユーザ名，パスワードを入力した場合だけ，test2/index.html へのアクセスを許可します。

```
AuthUserFile C:/user001/public_html/test2/.htpasswd        …1.
AuthGroupFile C:/user001/public_html/test2/.groupfile      …2.
AuthName "test2 Directory"                                 …3.
AuthType Basic
<Limit GET POST>                                           …4.
   Require group mygroup                                   …5.
</Limit>
```

1. パスワードファイルの定義

   パスワードファイルに登録しているユーザ名とパスワード

   ユーザ名：user001，パスワード：test21

   ユーザ名：user002，パスワード：test22

   ユーザ名：user003，パスワード：test23

2. グループファイルの定義

   グループファイルに登録しているグループ名

   グループ名：mygroup

   mygroup に登録しているユーザ名：user001，user002，user003

3. realm 名の定義

4. メソッドに対する定義

5. グループ名：mygroup のアクセスを許可

4.6　ファイル名一覧の表示

ディレクトリ内のファイル名一覧を Web ブラウザに表示する機能をディレクトリインデクスといいます。ディレクトリインデクス機能を有効にするには次に示すディレクティブを定義します。

```
Options +Indexes
```

このとき，すべてのファイルを表示させることはセキュリティ上危険です。IndexIgnore ディレクティブでインデクス表示させないファイルを指定する必要があります。

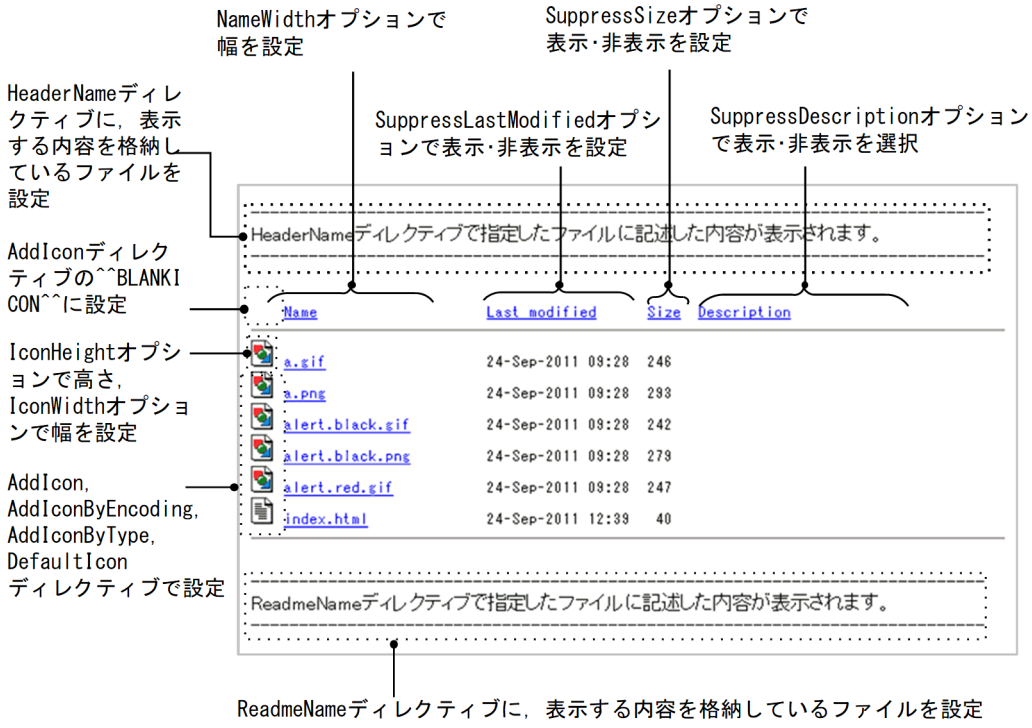
ただし，Options +Indexes を指定していても，DirectoryIndex ディレクティブに指定しているファイル（デフォルトは index.html ファイル）がそのディレクトリ下にある場合は，その指定されているファイルが表示されます。

さらに，ディレクトリインデクスを整形表示する場合は，次のディレクティブを指定します。

```
IndexOptions +FancyIndexing
```

整形表示機能の詳細設定は IndexOptions ディレクティブ，AddIcon ディレクティブで指定します。ディレクトリインデクス機能で表示される画面と，各ディレクティブで設定する内容を次に示します。

図 4-7　整形表示機能についての定義内容



なお，マルチバイト文字列を含むファイル名の表示はできません。

また，HeaderName ディレクティブおよび ReadmeName ディレクティブで指定したファイルで使用している文字セットが，デフォルトの文字セット（UNIX 版：ISO-8859-1，Windows 版：UTF-8）と異

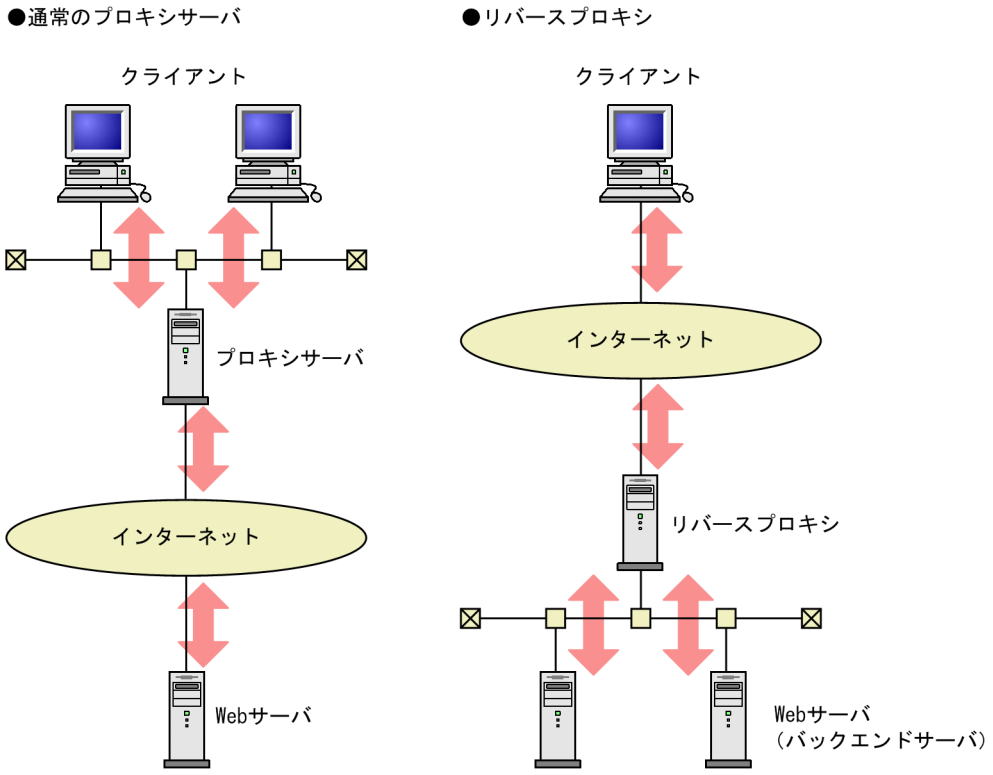
なる場合は，ディレクトリインデクス表示で文字化けが発生します。この場合，IndexOptions ディレクティブの Charset オプションで，HeaderName ディレクティブや ReadmeName ディレクティブで指定したファイルで使用している文字セットを指定してください。

4.7　リバースプロキシの設定

直接インターネットに接続できないクライアントからのリクエストをクライアントに代わって Web サーバに送信する代行サーバを**プロキシサーバ**といいます。通常，プロキシサーバは，クライアントとインターネットとの接点に設置されます。これに対し，インターネットと Web サーバとの接点にプロキシサーバを設置した場合を**リバースプロキシ**といいます。リバースプロキシでは，クライアントからのリクエストを Web サーバに代わってプロキシサーバが処理します。

通常のプロキシサーバとリバースプロキシの相違を次に示します。

図 4-8　通常のプロキシサーバとリバースプロキシの相違



リバースプロキシを使用してできることを次に示します。

- コンテンツへの直接アクセスを防止できます。

  Web サーバに重要な情報（クレジットカード番号のデータベースなど）を保持している場合，リバースプロキシと Web サーバを別のマシンに設定し，悪意のあるアクセスから Web サーバを守り，情報の漏えいを防げます。

- プロキシサーバに負荷の高い SSL 処理を集約できます。

  リバースプロキシを使用し，SSL の処理を別のマシンですれば，Web サーバに掛かる負荷を分散できます。

- クライアントに影響を与えないで，Web サーバを分割できます。

  Web サーバを分割した場合でも，リバースプロキシが代行するので，クライアントは分割前と同じインタフェースでアクセスできます。

4.7.1　プロキシモジュールの組み込み

リバースプロキシ機能を使用するためには，プロキシモジュールの組み込みが必要です。プロキシモジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。UNIX版の場合は，必ず次に示す順序でLoadModuleディレクティブを指定してください。

- UNIX版

```
LoadModule proxy_module libexec/mod_proxy.so
LoadModule proxy_http_module libexec/mod_proxy_http.so※1
LoadModule proxy_http2_module libexec/mod_proxy_http2.so※2
```

- Windows版

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so※1
LoadModule proxy_http2_module modules/mod_proxy_http2.so※2
```

注※1

バックエンドサーバとの通信でHTTP/1.1を使用する場合に必要となります。

注※2

バックエンドサーバとの通信でHTTP/2を使用する場合に必要となります。

4.7.2　ディレクティブの設定方法

リバースプロキシ機能を使用するための，ディレクティブの設定例を次に示します。

ここではHTTP/1.1を使用するものとし，各アドレスを次のように仮定しています。

リバースプロキシ：www.example.com

バックエンドサーバ：backend.example.com

(1)　リクエストURLの再割り当ておよびリクエストヘッダの再割り当て

次のようにProxyPassディレクティブを設定すると，クライアントからの"http://www.example.com/news/oct-2001"というリクエストは"http://backend.example.com/oct-2001"というリクエストに変更されます。

```
ProxyPass /news/ http://backend.example.com/
```

Host:ヘッダは"Host:www.example.com"から"Host:backend.example.com"に再割り当てします。そして，リバースプロキシはバックエンドサーバからのレスポンスをクライアントに応答します。

(2)　応答ヘッダの再割り当て

Redirect ディレクティブの指定，イメージマップの利用または末尾を/（スラッシュ）で閉じないディレクトリ指定のリクエストなど，バックエンドサーバでリダイレクトが指示された場合には，バックエンドサーバからのレスポンスの Location ヘッダにバックエンドサーバのアドレスが記載されます。これをそのままクライアントに応答すると，クライアントはリダイレクトをリバースプロキシではなく，直接バックエンドサーバにリクエストします。そこで，ProxyPassReverse ディレクティブに次のように指定し，リダイレクトリクエストもリバースプロキシを通るリクエストになるようにします。

```
ProxyPassReverse /news/ http://backend.example.com/
```

これで，Location ヘッダはリバースプロキシのアドレスに変更されます。

(3)　Set-Cookie ヘッダの再割り当て

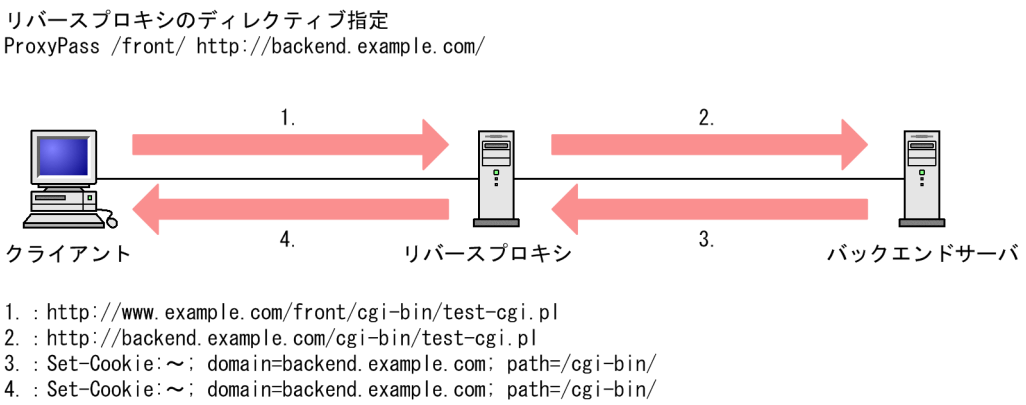
バックエンドサーバがクライアントに返す Set-Cookie ヘッダには，ドメイン名およびパス名が指定される場合があります。これは，Set-Cookie ヘッダのドメイン名およびパス名に一致したリクエストの場合だけ，クライアントにクッキーを送信させるためです。

Set-Cookie ヘッダの再割り当てをしない場合と再割り当てをする場合について説明します。

- Set-Cookie ヘッダの再割り当てをしない例

  バックエンドサーバが応答したドメイン名およびパス名を含む Set-Cookie ヘッダをリバースプロキシがそのままクライアントに応答する例を次の図に示します。なお，図中の数字は，説明文の項番と対応しています。

  図 4-9　Set-Cookie ヘッダの再割り当てをしない例

  1. クライアントからリバースプロキシに対して，http://www.example.com/front/cgi-bin/test-cgi.pl がリクエストされます。
  2. リバースプロキシは，URL を変換してバックエンドサーバへ転送します。
  3. リバースプロキシは，バックエンドサーバからドメイン名 domain=backend.example.com，パス名 path=/cgi-bin/の Set-Cookie ヘッダを受信します。

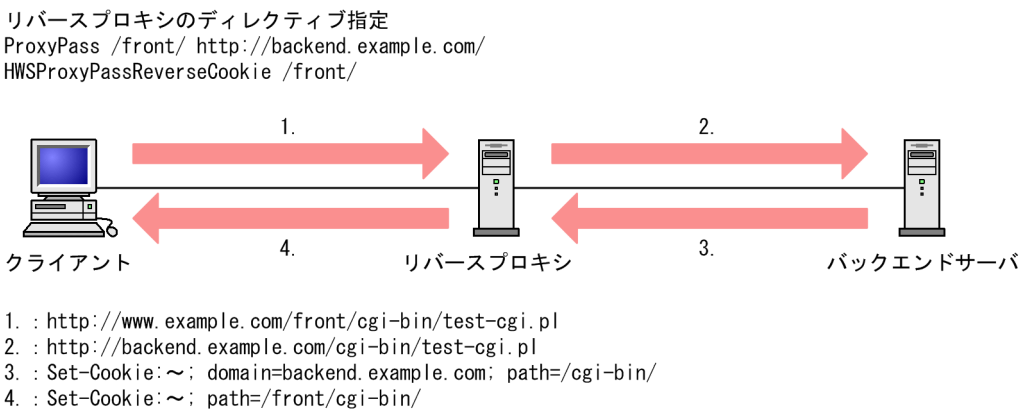
4. リバースプロキシは，バックエンドサーバから受信した Set-Cookie ヘッダをそのままクライアントに返します。

この場合，クライアントはリバースプロキシを経由する/front/cgi-bin/以下へのリクエストについて，Set-Cookie ヘッダで受信したクッキーを送信しません。これは，クライアントが受信した Set-Cookie ヘッダのドメイン名 backend.example.com が，リバースプロキシのドメイン名 www.example.com と異なるためです。また，パス名についても同様に適合しません。

- Set-Cookie ヘッダの再割り当てをする例

バックエンドサーバが Set-Cookie ヘッダで応答したクッキーをクライアントから受け取るためには，HWSProxyPassReverseCookie ディレクティブの指定が必要です。HWSProxyPassReverseCookie ディレクティブを指定して Set-Cookie ヘッダの再割り当てをする例を次の図に示します。なお，図中の数字は，説明文の項番と対応しています。

図 4-10　Set-Cookie ヘッダの再割り当てをする例



1. クライアントからリバースプロキシに対して，http://www.example.com/front/cgi-bin/test-cgi.pl がリクエストされます。
2. リバースプロキシは，URL を変換してバックエンドサーバへ転送します。
3. リバースプロキシは，バックエンドサーバからドメイン名 domain=backend.example.com，パス名 path=/cgi-bin/の Set-Cookie ヘッダを受信します。
4. リバースプロキシは，再割り当てした Set-Cookie ヘッダをクライアントに返します。

この場合，クライアントはリクエスト URL のパス部分/front/cgi-bin/test-cgi.pl に対して，前方一致するパス名/front/cgi-bin/の Set-Cookie ヘッダを受信します。また，クライアントが受信する Set-Cookie ヘッダにはドメイン名が含まれていません。これは，クライアントがリクエストした URL のドメイン名 www.example.com が Set-Cookie ヘッダに指定されている場合と同じ意味となります。したがって，リバースプロキシを経由したバックエンドサーバへのリクエストに，Set-Cookie ヘッダで設定したクッキーを送信させることができます。

4.7.3　システム構築例

リバースプロキシとバックエンドサーバに HTTP Server を使用してシステムを構築する場合の設定例を次に示します。

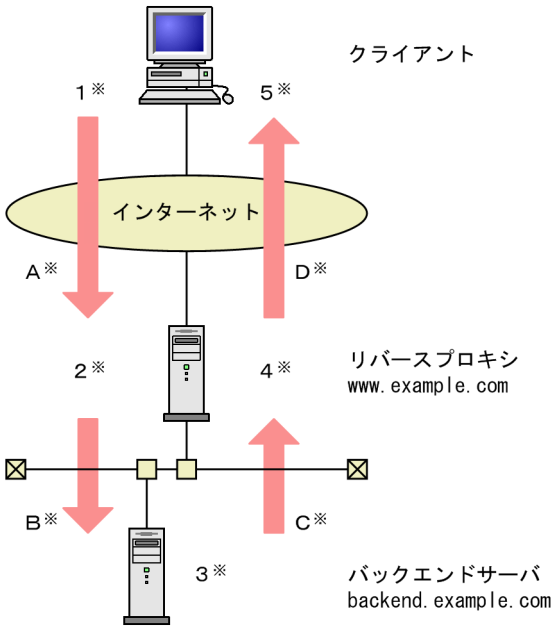
システムの構築時には，リダイレクト処理に注意して設定する必要があります。バックエンドサーバ上のディレクトリに対し，URL の最後に/ (スラッシュ)を付けないでアクセスした場合，バックエンドサーバは Location ヘッダを付加したリダイレクト要求を返信します。このとき，Location ヘッダの値をバックエンドサーバのアドレスからリバースプロキシのアドレスに変換し，クライアントの再要求先をリバースプロキシ経由に変更する必要があります。

ここでは，システムのネットワーク構成を図 4-10 のように仮定しています。また各アドレスを次のように仮定しています。

リバースプロキシ：www.example.com

バックエンドサーバ：backend.example.com

図 4-11　ネットワーク構成



注※　以降で説明する(a)，(b)の各リダイレクト処理の流れと対応しています。

(1)　推奨する構成

ProxyPass ディレクティブに指定するホスト名，パス名と，ProxyPassReverse ディレクティブに指定するホスト名，パス名は同一の値としてください。また，バックエンドサーバ側のすべてのバーチャルホストで ServerName ディレクティブを指定し，その値はリバースプロキシ側の ProxyPassReverse ディレクティブに設定したホスト名と同一にしてください。

図 4-10 に示すネットワーク構成で，リバースプロキシおよびバックエンドサーバの設定を表 4-6 のようにした場合のリダイレクト処理の流れは，表 4-7 のようになります。

表 4-6　推奨する構成の設定例

設定場所	設定内容
リバースプロキシ	ServerName www.example.com ProxyPass /before/ http://backend.example.com/after/ ProxyPassReverse /before/ http://backend.example.com/after/
バックエンドサーバ	ServerName backend.example.com

表 4-7　推奨する構成でのリダイレクト処理の流れ

図中の位置	説明
1	"http://www.example.com/before/dir"にアクセスします。
2	ProxyPass ディレクティブの値に従い，"http://backend.example.com/after/dir"にアクセスします。また，Host ヘッダの値を backend.example.com に書き換えて転送します。
3	URL の末尾に /（スラッシュ）が付いていないため，URL の末尾に/ (スラッシュ) を付けた URL を作成し，それを Location ヘッダに設定してリダイレクト要求を返します。
4	ProxyPassReverse ディレクティブの値に従い，Location ヘッダを"http://www.example.com/before/dir/"に書き換えて転送します。
5	Location ヘッダに従い，"http://www.example.com/before/dir/"に改めてアクセスします。
A	Host ヘッダの値は"www.example.com"です。
B	Host ヘッダの値は"backend.example.com"です。
C	Location ヘッダの値は"http://backend.example.com/after/dir/"です。
D	Location ヘッダの値は"http://www.example.com/before/dir/"です。

注

バックエンドサーバからの応答がステータスコード（302 Found や 404 Not Found など）になった場合，リバースプロキシはその HTML ドキュメントをそのままクライアントに転送します。「404 Not Found」などの HTML ドキュメントに記載されるバックエンドサーバ名や，「302 Found」などに記載されるリダイレクト先のリンクアドレスはリバースプロキシの情報に変更されません。バックエンドサーバ側で ErrorDocument ディレクティブを使用またはリバースプロキシ側で ProxyErrorOverride ディレクティブを使用して，バックエンドサーバの情報をクライアントに見せないようにしてください。

(2)　リバースプロキシ側で ProxyPreserveHost ディレクティブに On を設定する構成

通常，リバースプロキシはクライアントから受信した Host ヘッダの値を ProxyPass ディレクティブの値に従って変更し，バックエンドサーバに転送します。クライアントが送信した Host ヘッダの値をバックエンドサーバ側でも Host ヘッダの値として取得したい場合は，リバースプロキシ側で ProxyPreserveHost ディレクティブの値を On に設定します。このとき，次の点に注意してください。

- バックエンドサーバ側の ServerName ディレクティブには，リバースプロキシの ServerName と同じ値を指定してください。

- ProxyPassReverse ディレクティブに設定するホスト名は，リバースプロキシおよびバックエンドサーバの ServerName と同じ値にしてください。

図 4-10 に示すネットワーク構成で，リバースプロキシおよびバックエンドサーバの設定を表 4-8 のようにした場合のリダイレクト処理の流れは表 4-9 のようになります。

表 4-8　リバースプロキシ側で ProxyPreserveHost に On を設定する構成の設定例

設定場所	設定内容
リバースプロキシ	ServerName www.example.com ProxyPass /before/ http://backend.example.com/after/ ProxyPassReverse /before/ http://www.example.com/after/ ProxyPreserveHost On
バックエンドサーバ	ServerName www.example.com

表 4-9　リバースプロキシ側で ProxyPreserveHost に On を設定する構成でのリダイレクト処理の流れ

図中の位置	説明
1	"http://www.example.com/before/dir"にアクセスします。
2	ProxyPass ディレクティブの値に従い，"http://backend.example.com/after/dir"にアクセスします。また，ProxyPreserveHost ディレクティブの値が On に設定されているため，Host ヘッダの値は www.example.com のままです。
3	URL の末尾に /（スラッシュ）が付いていないため，URL の末尾に/ (スラッシュ) を付けた URL を作成し，それを Location ヘッダに設定してリダイレクト要求を返します。
4	ProxyPassReverse ディレクティブの値に従い，Location ヘッダを"http://www.example.com/before/dir/"に書き換えて転送します。
5	Location ヘッダに従い，"http://www.example.com/before/dir/"に改めてアクセスします。
A	Host ヘッダの値は"www.example.com"です。
B	Host ヘッダの値は"www.example.com"です。
C	Location ヘッダの値は"http://www.example.com/after/dir/"です。
D	Location ヘッダの値は"http://www.example.com/before/dir/"です。

4.7.4　注意事項

(1)　基本的な注意事項

- リバースプロキシはリクエスト URL のパターンによって機能を設定します。このため，特定のリクエストはリバースプロキシとしてほかのバックエンドサーバに転送，それ以外のリクエストはリバースプロキシ自身が Web サーバとして応答するという設定もできます。しかし，このような設定は，リクエ

ストがリバースプロキシか Web サーバかどちらで処理したかがわかりにくくなります。したがって，リバースプロキシを使用する場合は次のような設定にして，すべてのリクエストをリバースプロキシからバックエンドサーバへ転送することを推奨します。

```
ProxyPass / http://転送先バックエンドサーバアドレス/
```

リバースプロキシと Web サーバを共用する場合は，バーチャルホストで機能を分けた運用ができます。

- リバースプロキシでは，クライアントから受信した Host ヘッダの値を X-Forwarded-Host ヘッダに格納し，Host ヘッダの値を ProxyPass ディレクティブの指定値に変換してバックエンドサーバへ転送します。このため，バックエンドサーバ側のアプリケーションでクライアントが送信した Host ヘッダの値を参照する場合は，リバースプロキシが送信した X-Forwarded-Host ヘッダの値を参照してください。ただし，ProxyPreserveHost ディレクティブの値に On を設定している場合は，リバースプロキシが送信した Host ヘッダの値をそのまま参照してください。
- リバースプロキシを経由してバックエンドサーバにアクセスする場合，バックエンドサーバが提供する HTML コンテンツでのリンク先は，バックエンドサーバ上の URL ではなく，リバースプロキシにアクセスされる URL を指定する必要があります。このほか，画像やスタイルシートなどのコンテンツの参照先 URL を記述する場合も同じように注意が必要です。

  (例)

  次の状態にあるときに，index.html から index2.html へリンクを張るとします。

  - リバースプロキシ側で ProxyPass ディレクティブの値が /before/ http://バックエンドサーバのアドレス/after/ と指定されている。
  - バックエンドサーバ側で index.html と index2.html が同じディレクトリ内（/after/以下）に存在する。

  この場合の index.html の記述方法とアクセス可否の関係を次に示します。

表 4-10　リンクの記述方法とリンク可否の関係

リンクの記述	リンクをクリックしたときのアクセス可否
<A HREF="index2.html">リンク</A>	○
<A HREF="/before/index2.html">リンク</A>	○
<A HREF="http://リバースプロキシのアドレス/before/index2.html">リンク</A>	○
<A HREF="/after/index2.html">リンク</A>	×

- リバースプロキシは，HTTP バージョン 0.9 をサポートしていません。
- 11-20 以降では，CustomLog ディレクティブで指定するログのフォーマットで%{header_name}i を指定した場合，リバースプロキシ機能で付加するリクエストヘッダはログに出力されません。詳細は「付録 C　旧バージョンからの移行に関する注意点」を参照してください。
- リバースプロキシ機能では，クライアントからリクエスト受信時の送信元 IP アドレスを X-Forwarded-For ヘッダに格納し，バックエンドサーバへ転送します。このため，バックエンドサーバ側のアプリ

ケーションでクライアントの送信元 IP アドレスを参照する場合は，リバースプロキシ機能が送信した X-Forwarded-For ヘッダを参照してください。

- リバースプロキシでは，バックエンドサーバに接続したコネクションを維持したまま再利用する，コネクションプール方式を採用しています。

(2) ProxyPass ディレクティブに関する注意事項

- ProxyPass ディレクティブで指定するパス名とリクエスト URL は完全に等しいか，パス名がリクエスト URL の先頭から含まれていれば，適合と判断します。

  ただし，パス名の終端が/（スラッシュ）でない場合，リクエスト URL と完全に等しいかまたは先頭からディレクトリとして含まれていれば適合と判断します。

  適合すると，ProxyPass ディレクティブに指定したパス名にリクエスト URL の先頭からパス名と等しい部分を除いた残りの部分を追加してリクエストを転送します。

  ProxyPass ディレクティブに指定するパス名は終端を/（スラッシュ）で閉じたものを指定してください。次に ProxyPass ディレクティブの指定とリクエストの関係を次に示します。

表 4–11 ProxyPass ディレクティブの指定とリクエストの関係

ProxyPass ディレクティブの指定例	リクエスト	適合可否	リクエスト転送先
ProxyPass /abc/ http://backend.example.com/	http://リバースプロキシのアドレス/abc/	○	http://backend.example.com/
	http://リバースプロキシのアドレス/abc	×	−
	http://リバースプロキシのアドレス/abc/def	○	http://backend.example.com/def
ProxyPass /abc http://backend.example.com/	http://リバースプロキシのアドレス/abc	○	http://backend.example.com/
	http://リバースプロキシのアドレス/abc/	○	http://backend.example.com//
	http://リバースプロキシのアドレス/abc/def	○	http://backend.example.com//def

(凡例)

○：適合する。

×：適合しない。

−：該当しない。

- ProxyPass ディレクティブを複数指定し，リクエスト URL が複数のパス名に一致した場合，先に指定した ProxyPass ディレクティブが有効になります。

(例)

/abc/def/へのリクエストを処理するバックエンドサーバ：backend1.example.com

/abc/def/以外の/abc/へのリクエストを処理するバックエンドサーバ：backend2.example.com

ほかのすべてのリクエストを処理するバックエンドサーバ：backend3.example.com

このように設定するには次の順序で指定してください。

```
ProxyPass /abc/def/ http://backend1.example.com/
ProxyPass /abc/ http://backend2.example.com/
ProxyPass / http://backend3.example.com/
```

- ProxyPass ディレクティブの指定によるリクエスト URL は Web サーバの機能である自プロセス内の該当ファイルの検索より先に転送します。したがって，リクエスト URL に適合するファイルがある場合でも，ProxyPass ディレクティブのパス名に適合すれば，バックエンドサーバへのリクエストに変換して転送します。
- /（スラッシュ）で閉じていないディレクトリを指定したリクエスト URL の場合，リバースプロキシではリダイレクトを応答しません。

（例）ProxyPass /ab/ http://backend.example.com/　の場合

リクエストが http://リバースプロキシのアドレス/ab のとき，適合していないと判断して，リバースプロキシ内に/ab がなければ，「404 Not Found」を返します。

- ProxyPass ディレクティブの転送先 URL が，先に定義している ProxyPass ディレクティブの転送先 URL に含まれる場合，それらの ProxyPass ディレクティブはキーで指定された値を共有します。そのため，あとで定義している ProxyPass ディレクティブではキーの値を指定できません。

（例）

```
ProxyPass /test1/ http://backend.example.com:81/AAA/                    …  1.
ProxyPass /test2/ http://backend.example.com:81/AAA/BBB/  timeout=10    …  2.
```

2.の ProxyPass ディレクティブの転送先 URL は，1.の転送先 URL に含まれるため，1.と 2.はキーで指定された値を共有しています。あとで指定した 2.ではキーの値を指定できません。

(3) ProxyPassReverse ディレクティブに関する注意事項

- ProxyPassReverse ディレクティブに指定した URL とバックエンドサーバから受信した Location ヘッダの値は，完全に等しいかまたは URL がリクエスト URL の先頭から含まれていれば，適合と判断します。適合すると，アドレスをリバースプロキシとして，ProxyPassReverse ディレクティブの指定に従って，クライアントに送信します。

（例）バックエンドサーバからの応答の Location ヘッダが，Location: http://バックエンドサーバのアドレス/docs/memo/の場合

ProxyPassReverse ディレクティブの指定が，

ProxyPassReverse /path/ http://バックエンドサーバのアドレス/docs/

と指定されていれば，クライアントに返す Location ヘッダは

Location: http://リバースプロキシのアドレス/path/memo/

となります。

ProxyPassReverse ディレクティブを複数指定した場合，先に指定した方が有効になります。

- リバースプロキシが ProxyPassReverse ディレクティブの設定値に従って Location ヘッダの値を変換してクライアントに転送する際，Location ヘッダの値のスキームは現在のコネクションで使用しているものを設定します。例えば，http でアクセスしている場合，http を設定します。このため，http でアクセスしている場合に Location ヘッダで https にリダイレクトさせるときは，バックエンドサーバ側でリバースプロキシのホスト名を Location ヘッダの値に設定しておくなどして ProxyPassReverse ディレクティブの値と一致しないようにしてください。

(4) HWSProxyPassReverseCookie ディレクティブに関する注意事項

- HWSProxyPassReverseCookie ディレクティブは，バックエンドサーバが応答した Set-Cookie ヘッダを変換する場合に指定します。HWSProxyPassReverseCookie ディレクティブに，ProxyPass ディレクティブのパス名と同じ値を指定することで，ProxyPass ディレクティブ単位に設定できます。
- リバースプロキシのディレクティブ指定が次のような場合の Set-Cookie ヘッダの変換規則について説明します。

```
ProxyPass /front/ http://backend.example.com/
HWSProxyPassReverseCookie /front/
```

表 4-12　Set-Cookie ヘッダの変換規則

項番	クライアントに応答する Set-Cookie ヘッダ	バックエンドサーバが応答する Set-Cookie ヘッダ	変換規則の説明
1	Set-Cookie: ～; path=/front/	Set-Cookie: ～; path=/	バックエンドサーバが応答する Set-Cookie ヘッダにドメイン名が指定されていない場合は，Set-Cookie ヘッダのパス名/（スラッシュ）を/front/に置き換えます。
2	Set-Cookie: ～; path=/front/	Set-Cookie: ～; domain=backend.example.com; path=/	バックエンドサーバが応答する Set-Cookie ヘッダのドメイン名が，ProxyPass ディレクティブで指定した転送先 URL のドメイン名と完全に一致している場合は，Set-Cookie ヘッダのパス名/（スラッシュ）を/front/に置き換えます。また，Set-Cookie ヘッダのドメイン名を削除してクライアントに返します。
3	Set-Cookie: ～; domain=.example.com; path=/	Set-Cookie: ～; domain=.example.com; path=/	バックエンドサーバが応答する Set-Cookie ヘッダのドメイン名が，.（ピリオド）から始まるドメイン名である場合は，バックエンドサーバが応答した Set-Cookie ヘッダをそのままクライアントに返します。
4	Set-Cookie: ～; domain=other.example.com; path=/	Set-Cookie: ～; domain=other.example.com; path=/	バックエンドサーバが応答する Set-Cookie ヘッダのドメイン名が ProxyPass ディレクティブで指定した転送先 URL のドメイン名と異なる場合は，バックエンドサーバが応答した Set-

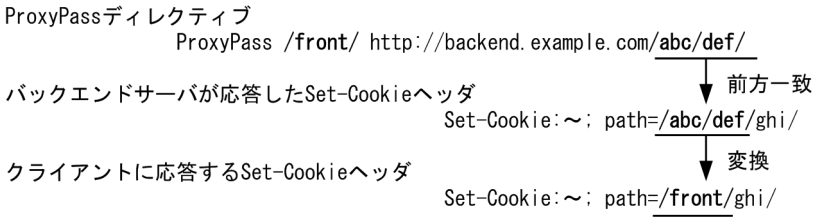
項番	クライアントに応答する Set-Cookie ヘッダ	バックエンドサーバが応答する Set-Cookie ヘッダ	変換規則の説明
			Cookie ヘッダをそのままクライアントに返します。
5	Set-Cookie: ～	Set-Cookie: ～	バックエンドサーバが応答する Set-Cookie ヘッダにドメイン名およびパス名が指定されていない場合は，バックエンドサーバが応答した Set-Cookie ヘッダをそのままクライアントに返します。

- リバースプロキシのディレクティブ指定が次の場合に，バックエンドサーバが応答した Set-Cookie ヘッダのパス名を変換する規則について説明します。

```
ProxyPass /front/ http://backend.example.com/abc/def/
HWSProxyPassReverseCookie /front/
```

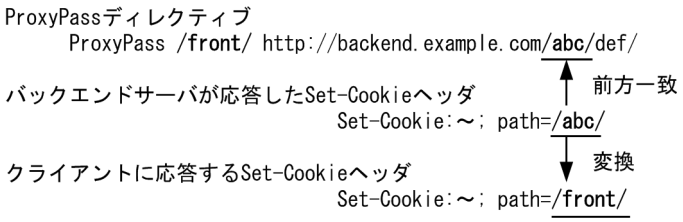
  - **バックエンドサーバが返す Set-Cookie ヘッダのパス名が/abc/def/ghi/の場合**

    ProxyPass ディレクティブの転送先 URL のパス名部分が，Set-Cookie ヘッダのパス名に前方から一致する場合は，一致したパス名部分を ProxyPass ディレクティブのパス名で置き換えます。



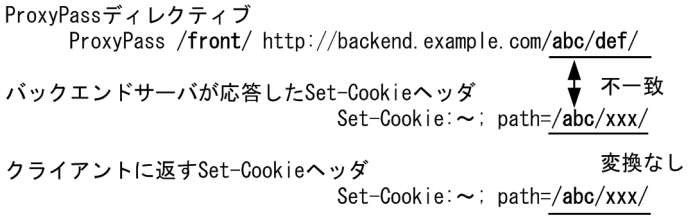
  - **バックエンドサーバが返す Set-Cookie ヘッダのパス名が/abc/の場合**

    Set-Cookie ヘッダのパス名が，ProxyPass ディレクティブの転送先 URL のパス名部分に前方から一致する場合は，Set-Cookie ヘッダのパス名として ProxyPass ディレクティブのパス名で置き換えます。



  - **バックエンドサーバが返す Set-Cookie ヘッダのパス名が/abc/xxx/の場合**

    ProxyPass ディレクティブの転送先 URL のパス名部分と Set-Cookie ヘッダのパス名が一致しない場合は，リバースプロキシでの Set-Cookie ヘッダ変換は実行しません。バックエンドサーバが応答した Set-Cookie ヘッダをそのままクライアントに返します。



(5) 性能に関する注意事項

- ProxyPass ディレクティブにドメイン名またはホスト名を指定している場合，DNS への問い合わせが発生します。バックエンドサーバの IP アドレスがわかっている場合は，hosts ファイルにあらかじめ IP アドレスを記載しておくことによって，名前解決の時間を短縮できます。
- バックエンドサーバとの最大コネクション数は次の計算式で表されます。

  バックエンドサーバが Cosminexus の NIO HTTP サーバの場合，バックエンドサーバとの最大コネクション数以上の値を webserver.connector.nio_http.max_connections キーに設定してください。

  Windows の場合

  ```
  最大コネクション数＝（ThreadsPerChildディレクティブ値
                        + H2MaxWorkersディレクティブ値※1）
                          ×ProxyPassディレクティブの定義数※2
  ```

  UNIX の場合

  ```
  最大コネクション数＝（MaxClientsディレクティブ値
                        + H2MaxWorkersディレクティブ値※1
                          × ServerLimitディレクティブ値）
                            ×ProxyPassディレクティブの定義数※2
  ```

  注※1

  HTTP/2 通信を使用しない場合は 0 とする。

  注※2

  ほかの ProxyPass と設定を共有しているものは除く。

- バックエンドサーバが Cosminexus の NIO HTTP サーバの場合，リバースプロキシのタイムアウト値は，バックエンドサーバのタイムアウト値より大きくなるようにシステム設計および定義設定をしてください。

  これに従ったタイムアウトの設定ができない場合は，バックエンドサーバの webserver.connector.nio_http.max_connections キーおよび webserver.connector.nio_http.max_threads キーに，上限値の 2147483647 を設定してください。

  設定例：

  webserver.connector.nio_http.max_connections=2147483647

  webserver.connector.nio_http.max_threads=2147483647

  上限値の 2147483647 を設定できない場合は，システム要件に合わせてバックエンドサーバとの最大コネクション数よりも，大きい値を設定してください。

  webserver.connector.nio_http.max_connections キーに，次の二つを加算した値以上を設定してください。

  - バックエンドサーバとの最大コネクション数
  - webserver.connector.nio_http.max_threads キーの設定値

4.8　稼働状況の表示（ステータス情報表示）

稼働中のプロセス数，待機中のプロセス数および各プロセスのステータス（R，W，L など）を Web ブラウザに表示します（Windows 版の場合はサーバスレッド数）。この情報を基に，StartServers，MinSpareServers，MaxSpareServers，MaxClients ディレクティブなどをチューニングできます（Windows 版の場合は ThreadsPerChild ディレクティブ）。各ディレクティブの詳細は，「4.1　HTTP Server の処理とディレクティブとの関係」を参照してください。

ExtendedStatus ディレクティブで On を指定すると，より詳細な情報が表示されます。

4.8.1　server-status ハンドラの指定

ステータス情報の表示機能を利用するには，次に示すように server-status ハンドラを指定します。

```
<Location /server-status>
      SetHandler server-status
</Location>
```

ただし，Web サーバのステータス情報はアクセス制御して，エンドユーザには非公開にするのが一般的です。

4.8.2　URL の指定

ステータス情報を表示するには，Web ブラウザから次に示す形式で URL を指定します。なお，server-status は，タイミングによって一時的に正しく表示されない場合があります。

```
http://ホスト名〔:ポート番号〕/server-status〔? {refresh=更新間隔 | auto | notable} 〕
```

refresh=更新間隔，auto，notable はそれぞれ&でつないで指定できます。ただし，auto はプレーンテキスト形式であるため，notable と同時に指定するのは意味がありません。

- refresh=更新間隔　((1-3600))

  Web ブラウザ上のステータス情報を更新する間隔を秒単位で指定します。ただし，Web ブラウザが HTTP レスポンスヘッダの Refresh ヘッダに対応した機能をサポートしている必要があります。指定可能範囲外の値が指定された場合は 60 秒が設定されます。

- auto

  プレーンテキスト形式で表示します。プレーンテキスト形式のため，ほかのプログラムで容易に処理できます。

- notable

  <TABLE>タグを用いない HTML でステータス情報を表示します。

＜指定例＞

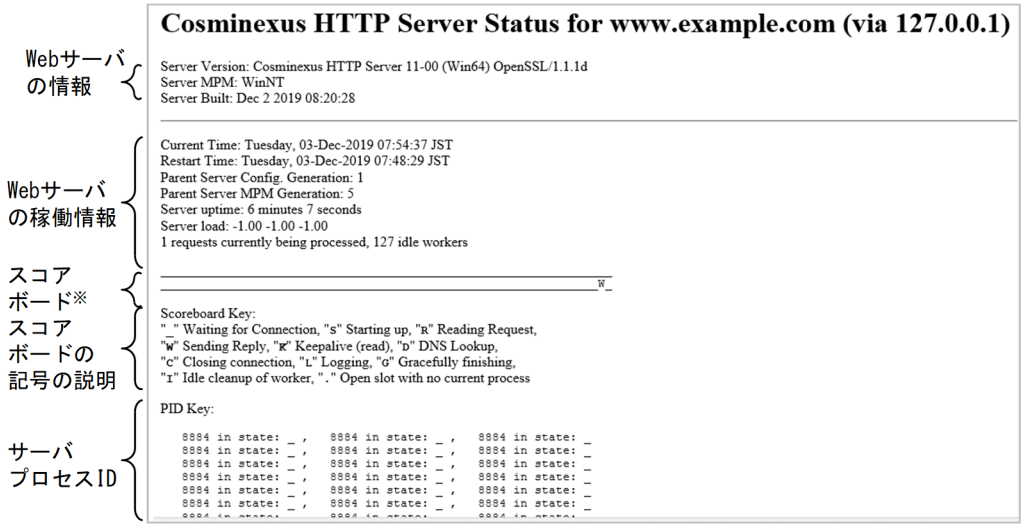
```
http://www.example.com/server-status?refresh=60&notable
```

＜表示例＞

```
http://www.example.com/server-status
```

このように指定した場合の，ステータス情報の表示例を次に示します。表示形式は，UNIX 版と Windows 版で若干異なります。

図 4-12　ステータス情報の表示例



注※　スコアボードには，稼働中のサーバプロセスの状態を記号で示す。
記号の意味を次に示す。
_ ... リクエスト待ち状態
S ... 起動処理中
R ... クライアントからのリクエストを受信中
W ... リクエストの処理実行，およびクライアントへレスポンス送信中
K ... 持続型接続状態でリクエスト受信待ち
D ... ルックアップ中(HostnameLookupsディレクティブ参照)
C ... 接続を終了中
L ... ログ出力処理中
G ... gracefulリスタートにおける処理終了待ち
I ... スレッド停止中
. ... 起動していない状態

4.8.3　取得できる情報

ステータス情報の表示機能で取得できる情報を次に示します。ExtendedStatus ディレクティブで On を指定すると，詳細な情報を取得できます。

表 4-13　ステータス情報の表示機能で取得できる情報（auto 指定がない場合）

項番	内容	説明	ExtendedStatus の値と取得可否	
			Off	On
1	Server Version	サーバのバージョン	○	○
2	Server MPM	サーバで動作している MPM	○	○
3	Server Built	サーバのビルド時間	○	○
4	Current Time	現在時刻	○	○
5	Restart Time	起動時刻	○	○
6	Parent Server MPM Generation	サーバプロセスの再起動回数（初期値 0）	○	○
7	Server uptime	サーバプロセスの稼働時間	○	○
8	Total accesses	合計アクセス回数	×	○
9	Total Traffic	合計通信量	×	○
10	CPU Usage: u vvv s www cu xxx cs yyy - zzz% CPU load	ユーザ時間，システム時間，子プロセスのユーザ時間，子プロセスのシステム時間，CPU 使用率（UNIX 版）	×	○
11	xxx requests/sec - yyy B/second - zzz B/request	1 秒当たりのリクエスト数， 1 秒当たりの通信量， 1 リクエスト当たりの通信量	×	○
12	xxx requests currently being processed, yyy idle workers	リクエスト処理中のサーバプロセス（スレッド）数， リクエスト待ち状態のサーバプロセス（スレッド）数	○	○
13	スコアボード	個々のスレッドの動作状況	○	○
14	Scoreboard Key	スコアボードの凡例	○	○
15	PID Key	個々のスレッドのサーバプロセス ID と動作状況	○	×
16	Srv	サーバプロセスの識別子と再起動回数	×	○
17	PID	プロセス ID	×	○
18	Acc	アクセス数（コネクション単位/スレッド単位/スロット単位）	×	○
19	M	動作状況	×	○
20	CPU	CPU 時間（秒）（UNIX 版）	×	○
21	SS	最後の処理開始からの経過秒	×	○
22	Req	最後の処理に要したミリ秒	×	○
23	Conn	コネクションに対する通信量	×	○
24	Child	プロセスの通信量	×	○

項番	内容	説明	ExtendedStatus の値と取得可否	
			Off	On
25	Slot	スロットの通信量	×	○
26	Client	最後の処理のクライアント	×	○
27	VHost	バーチャルホスト名	×	○
28	Request	最後の処理のリクエストライン （HTTP/2 通信の場合は HTTP/2 セッション情報も表示）	×	○

（凡例）

○：取得できる。

×：取得できない。

表 4-14　ステータス情報の表示機能で取得できる情報（auto 指定がある場合）

項番	内容	説明	ExtendedStatus の値と取得可否	
			Off	On
1	Total accesses	合計アクセス回数	×	○
2	Total kBytes	合計通信量	×	○
3	CPULoad	CPU 使用率（UNIX 版）	×	○
4	Uptime	サーバプロセスの稼働時間（秒）	×	○
5	ReqPerSec	1 秒当たりのリクエスト数	×	○
6	BytesPerSec	1 秒当たりの通信量	×	○
7	BytesPerReq	1 リクエスト当たりの通信量	×	○
8	BusyWorkers	リクエスト処理中のサーバプロセス（スレッド）数	○	○
9	IdleWorkers	リクエスト待ち状態のサーバプロセス（スレッド）数	○	○
10	スコアボード	個々のスレッドの動作状況	○	○

（凡例）

○：取得できる。

×：取得できない。

4.8.4　注意事項

サーバステータス表示機能で表示される「Current Time」および「Restart Time」のタイムゾーンの情報に，マルチバイト文字が設定されることがあります。このとき，HTTP Server では，これらの文字列をすべてエスケープ（「¥x」から始まる接頭辞と 16 進コードで構成される文字列に置換）します。

4.9　流量制限機能

Web サーバへのアクセスの増加や，業務アプリケーションなどの影響で Web サーバの負荷が高くなった場合に，Web サイトにアクセスするユーザ数を制限するなどして，Web サービスの処理効率を維持する機能を流量制限機能といいます。

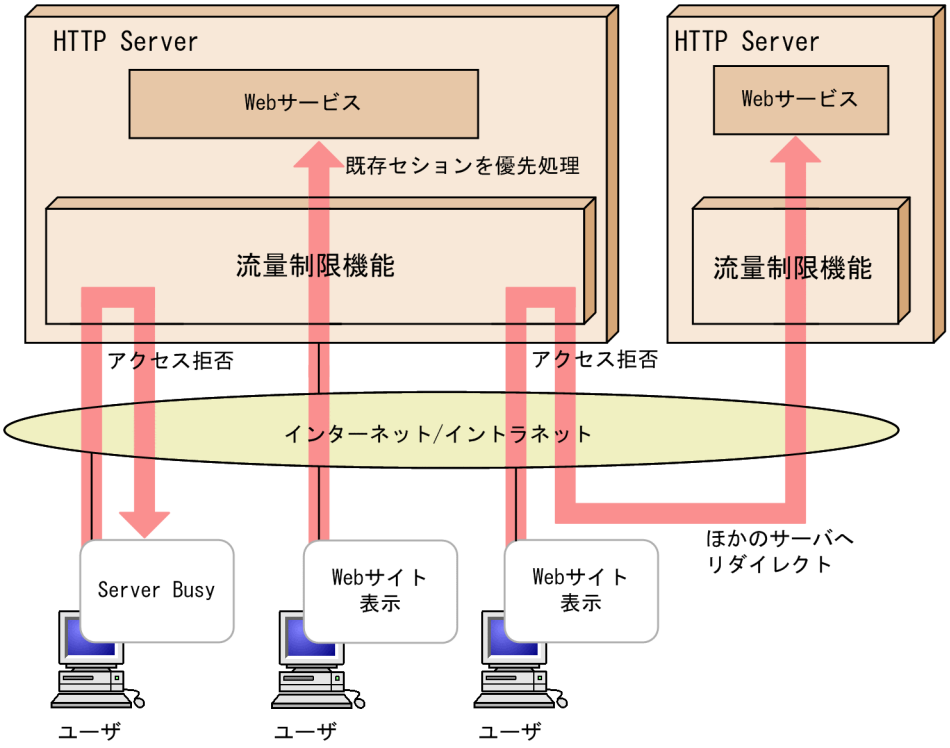
HTTP Server に mod_hws_qos モジュールを組み込めば，流量制限機能を使用できます。流量制限機能を使用すると，次に示すことができます。なお，以降の記述の「サーバプロセス数」は，UNIX 版の worker MPM モジュールの場合および Windows 版の場合「サーバスレッド数」のことです。

- リクエスト処理を実施するサーバプロセス数を制限すれば，Web サイトに同時にアクセスするユーザ数を制限できます。高負荷時には，制限値を超えたリクエストに対してすぐに拒否レスポンスを返したり，ほかの Web サーバへリダイレクトさせたりすれば，レスポンスタイムを維持できます。
- クッキーを使用したセション管理によって，高負荷時，新しいセションを拒否し，すでにアクセスしているユーザのレスポンスタイムを維持できます。

十分な Web サービスを提供するには，同時にアクセスするユーザ数を満たすだけのサーバプロセス数が必要です。Web サーバが 1 台で不十分なら，複数台用意し，負荷分散機でアクセスを分散させるなどして Web サービスを保証できるように運用設計してください。

このように Web サービスの資源を用意しても，一時的に過負荷状態が発生する場合に備えて，流量制限機能を使用してください。mod_hws_qos による流量制限機能の概要を次に示します。

図 4-13　mod_hws_qos による流量制限機能の概要



4.9.1　mod_hws_qos モジュールの組み込み

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。mod_hws_qos モジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule hws_qos libexec/mod_hws_qos.so
```

- Windows 版

```
LoadModule hws_qos modules/mod_hws_qos.so
```

4.9.2　ディレクティブの設定方法

流量制限機能を使用するための，各ディレクティブの設定例を次に示します。

(1)　サーバプロセス数（サーバスレッド数）の制限によるリクエスト拒否

次のように指定すると，リクエスト処理中のサーバプロセス数（サーバスレッド数）が 13 の場合には，新たなリクエスト要求はステータスコード 503 で拒否されます。

- UNIX 版

```
MaxClients 15
QOSRejectionServers 2
QOSCookieServers 0
```

- Windows 版

```
ThreadsPerChild 15
QOSRejectionServers 2
QOSCookieServers 0
```

(2)　クッキーを使用したセション管理

クッキーを使用したセション管理には，HTTP Server で作成したクッキーを使用する HWS 作成モードと，HTTP Server 以外の外部モジュールなどで作成されたクッキーを使用するユーザ作成モードがあります。QOSCookieName ディレクティブを用いて，どちらの方式を使用するかを選択します。QOSCookieName ディレクティブの詳細については，「6.2　ディレクティブの詳細」を参照してください。

HWS 作成モード

リクエスト処理を実施した場合には，HTTP Server で作成したクッキーが，レスポンスヘッダの Set-Cookie ヘッダに付けられます。HTTP Server で作成したクッキーを持ったリクエスト要求は，持っていないリクエスト要求よりも優先して処理されます。

ユーザ作成モード

HTTP Server 以外で作成されたクッキーがレスポンスヘッダの Set-Cookie ヘッダに付けられる場合に，そのクッキーを使用した流量制限が実行されます。そのクッキーを持ったリクエスト要求は，持っていないリクエスト要求よりも優先して処理されます。

次のように指定すると，リクエスト処理中のサーバスレッド数が 10 の場合，クッキーを持っていない新しいセションのリクエスト要求は拒否されますが，クッキーを持った継続セションは処理されます。リクエスト処理中のサーバスレッド数が 13 の場合は，クッキーを持っているかどうかに関係なく拒否されます。この例では HWS 作成モードで動作します。

- UNIX 版

```
MaxClients 15
QOSRejectionServers 2
QOSCookieServers 5
```

- Windows 版

```
ThreadsPerChild 15
QOSRejectionServers 2
QOSCookieServers 5
```

(3) リダイレクト

流量制限機能によってリクエスト処理を拒否する場合には，ステータスコード 503 でレスポンスメッセージを返送しますが，次のように指定すると，ほかの Web サーバへリダイレクトさせることができます。/index.html へのリクエスト要求が流量制限機能によって拒否された場合には，www1.hitachi.co.jp の Web サーバの index.html をレスポンスヘッダに設定し，ステータスコード 302 で返送されます。

```
QOSRedirect /index.html http://www1.hitachi.co.jp/index.html
```

(4) レスポンスメッセージのカスタマイズ

次のように指定すると，拒否されたリクエストに対するレスポンスは，ステータスコード 503 で，レスポンスメッセージは，htdocs/busy.html の内容が返送されます。

```
QOSResponse file ”text/html; charset=ISO-8859-1” htdocs/busy.html
```

4.9.3　レスポンスメッセージ

(1)　サーバからクライアントに送信されるクッキーについて

HWS 作成モード

HTTP Server で作成したクッキーは，Set-Cookie ヘッダによってクライアントへ返送されます。Set-Cookie ヘッダは一つのレスポンスに複数指定できるため，ここで作成するクッキーは，ほかのクッキーに影響しません。HTTP Server で返送する Set-Cookie ヘッダを次に示します。

```
Set-Cookie: NAME=VALUE; expires=DATE; path=/; domain=DOMAIN_NAME; secure
```

NAME=VALUE

NAME には QOSCookieName ディレクティブで指定した名称が設定されます。VALUE にはリクエスト制御用の値が設定されます。

expires=DATE

クッキーが無効となる時刻。"リクエストの受信時刻＋ QOSCookieExpires ディレクティブ設定値"によって求めた値が，RFC822 形式で設定されます。

path=/

クッキーが有効となる URL。このモジュールでは，クッキーが有効となるドメイン内の，すべての URL で有効となるように設定されます。

domain=DOMAIN_NAME

クッキーが有効となるドメイン。QOSCookieDomain ディレクティブで指定されます。

secure

SSL による通信時だけ，クッキーをクライアントからサーバに送信するかどうかの指定。QOSCookieSecure ディレクティブで指定されます。

ユーザ作成モード

HTTP Server ではクッキーを作成しません。HTTP Server 以外で作成されたクッキーが，Set-Cookie ヘッダによってクライアントへ返送されます。

(2)　流量制限機能によって拒否された場合のヘッダについて

流量制限機能によって拒否された場合，レスポンスメッセージをキャッシュできなくするためのヘッダである Expires を，レスポンスヘッダに含めます。これは，サーバ側でリクエスト処理が可能であっても，プロキシまたはブラウザでキャッシュされると，キャッシュされたメッセージがブラウザに表示されて，サーバにリクエストしない場合があるためです。また，拒否メッセージ送信後はサーバ側からコネクションを切断します。

そのほかのレスポンスヘッダは，次のように設定されます。Content-Type には，AddDefaultCharset で指定した文字セットが付加されます。

(i)ステータスコード 503 の標準メッセージ

Content-Type：text/html

(ii)QOSResponse によるカスタマイズされたメッセージ

Content-Type：QOSResponse ディレクティブ指定値

(iii)QOSRedirect によるステータスコード 302 のメッセージ

Content-Type：text/html

Location：QOSRedirect ディレクティブ指定値

4.9.4 注意事項

- クッキーの受け付けを拒否しているクライアントは，クッキーをサーバに送信しないため，「4.9.2(2) クッキーを使用したセション管理」の機能が無効となります。
- KeepAlive によって接続した場合には，接続後最初のリクエストを処理するときだけ判定処理が行われ，同一接続上で 2 回目以降のリクエストでは判定されません。最初に接続したリクエストとは異なる流量制限設定へのリクエストである場合も，2 回目以降のリクエストでは判定されません。
- 流量制限機能によってアクセス拒否された場合に送信されるメッセージは，ErrorDocument ディレクティブを指定しても変更されません。また，Redirect ディレクティブや RedirectMatch ディレクティブの処理は，mod_hws_qos モジュールの制御によって処理を継続すると判定されたあとに実行されます。
- 流量制限機能を使用しても，QOSRejectionServers ディレクティブの設定数を超えた同時リクエストを受信した場合は，リクエストの拒否が正しくできないことがあります。QOSResponse ディレクティブで指定された HTML ファイルに画像データなどのリンクが含まれている場合には，画像データを取得するため，さらにアクセスします。このアクセスも流量制限処理の対象となり，画像データが取得できない場合があります。HTML ファイル作成時には，リンクの設定に注意してください。
- エラードキュメントの文字セットは，Windows 版では HWSErrorDocumentMETACharset ディレクティブの設定を有効とします。
- クッキーを使用したセション管理を URL ごとまたは VirtualHost ごとに設定する場合は，QOSCookieName ディレクティブで別の名称を指定する必要があります。
- QOSCookieServers ディレクティブおよび QOSRejectionServers ディレクティブは，サーバプロセス数（ThreadsPerChild ディレクティブまたは MaxClients ディレクティブで設定）より後ろに指定してください。サーバプロセス数よりも前に指定すると，サーバが起動できない場合があります。
- HTTP/2 プロトコル通信機能では流量制限機能は使用できません。

4.10　ヘッダカスタマイズ機能

HTTP 通信では，Web ブラウザと Web サーバの間でさまざまな HTTP ヘッダが用いられます。Web ブラウザおよび Web サーバは，受信した HTTP ヘッダから，その後の動作を決定する場合があります。Web ブラウザが HTTP リクエストの送信時に付加する HTTP ヘッダをリクエストヘッダ，Web サーバが応答時に付加する HTTP ヘッダをレスポンスヘッダといいます。Web サーバが受信したリクエストヘッダや送信するレスポンスヘッダを追加，変更または削除して，Web サーバまたは Web ブラウザに特定の動作をさせるための機能をヘッダカスタマイズ機能といいます。

HTTP Server に mod_headers モジュールを組み込むことで，ヘッダカスタマイズ機能を使用できます。

4.10.1　mod_headers モジュールの組み込み

ヘッダカスタマイズ機能を使用するためには mod_headers モジュールの組み込みが必要です。mod_headers モジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule headers_module libexec/mod_headers.so
```

- Windows 版

```
LoadModule headers_module modules/mod_headers.so
```

4.10.2　ディレクティブの設定方法

ヘッダカスタマイズ機能は，Header ディレクティブおよび RequestHeader ディレクティブで指定します。ヘッダカスタマイズ機能を使用するための，ディレクティブの設定例を次に示します。

(1)　レスポンスヘッダを設定する場合

Header ディレクティブの set 指示子によって，レスポンスヘッダを設定できます。ほかのモジュールですでに同じ名前のレスポンスヘッダが設定されている場合は，ヘッダ値を上書きします。

レスポンスヘッダに Expires: Sat, 1 Jan 2000 00:00:00 GMT を設定する例を次に示します。ただし，有効期限を動的に設定する場合は，有効期限設定機能を使用してください。

```
Header set Expires ”Sat, 1 Jan 2000 00:00:00 GMT”
```

(2) レスポンスヘッダを追加する場合

Header ディレクティブの add 指示子によって，レスポンスヘッダを追加できます。ほかのモジュールですでに同じ名前のレスポンスヘッダが設定されていても，別のヘッダとして設定されます。同じ名前のレスポンスヘッダを複数行設定する場合に使用します。

レスポンスヘッダに Set-Cookie: HOSTNAME=HOST1; path=/; domain=www.example.com; secure を追加する例を次に示します。

```
Header add Set-Cookie ”HOSTNAME=HOST1; path=/; domain=www.example.com; secure”
```

(3) バックエンドのレスポンスヘッダ値を書き換える場合

Header ディレクティブの edit 指示子によって，レスポンスヘッダ値の書き換えができます。バックエンドからのレスポンスヘッダ値に secure 属性などを付加したい場合に使用します。

バックエンドが設定した Set-Cookie レスポンスヘッダのヘッダ値に secure 属性を付加する例を次に示します。

```
Header edit Set-Cookie ^(.+)$ ”$1; secure”
```

4.10.3 注意事項

- レスポンスヘッダのうち，Date，Server，Content-Type，Content-Length，Last-Modified ヘッダなどは，カスタマイズできない場合があります。また，LoadModule ディレクティブでほかのモジュールを組み込んだ場合，これら以外のヘッダでもカスタマイズできない場合があります。

4.11　有効期限設定機能

Web サーバ上のコンテンツに有効期間を設定すると，その期間中，キャッシュ機能をサポートしているクライアントやプロキシサーバは，Web サーバにアクセスしないで，自身のキャッシュにアクセスするようになるため，効率的です。

HTTP Server に mod_expires モジュールを組み込むことで，有効期限設定機能を使用できます。有効期限設定機能を使用すると，次に示すことができます。

- 有効期限設定機能を使用すると，レスポンスに Expires ヘッダおよび Cache-Control ヘッダが追加されます。
- Expires ヘッダでは有効期限がグリニッジ標準時（GMT）で設定され，Cache-Control ヘッダでは max-age 指示子に，有効期限までの時間が秒単位で設定されます。

設定された Expires ヘッダおよび Cache-Control ヘッダの扱いは，クライアントやプロキシサーバに依存します。

4.11.1　mod_expires モジュールの組み込み

有効期限設定機能を使用するためには，mod_expires モジュールの組み込みが必要です。mod_expires モジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule expires_module libexec/mod_expires.so
```

- Windows 版

```
LoadModule expires_module modules/mod_expires.so
```

4.11.2　ディレクティブの設定方法

有効期限設定機能を使用するための，ディレクティブの設定例を次に示します。

(1)　デフォルトの有効期限の設定

Web サーバ上のすべてのコンテンツを対象に，ExpiresDefault ディレクティブでデフォルトの有効期限を設定します。有効期限は，ファイルの更新時刻またはクライアントがアクセスした時刻を基準にして設定します。

次のように指定すると，クライアントがアクセスした時刻から 60 秒後を有効期限として，Expires ヘッダおよび Cache-Control ヘッダがレスポンスに追加されます。

```
ExpiresActive On
ExpiresDefault A60
```

ExpiresDefault の"A"指定は，クライアントがアクセスした時刻を基準時刻としていることを示します。

(2) MIME タイプ別の有効期限の設定

ExpiresByType ディレクティブで MIME タイプ別に有効期限を設定します。ExpiresDefault ディレクティブで設定されたデフォルトの有効期限は，この設定によって MIME タイプ別に上書きされます。有効期限は，ファイルの更新時刻またはクライアントがアクセスした時刻を基準にして設定します。
ExpiresByType ディレクティブに指定する MIME タイプは，TypesConfig ディレクティブに指定したファイル（デフォルト値：conf/mime.types ファイル）内にファイル拡張子とコンテントタイプ（MIME タイプ）の関係を定義するか，または AddType ディレクティブにファイル拡張子とコンテントタイプ（MIME タイプ）の関係を定義する必要があります。

次のように指定すると，MIME タイプが text/html の場合にだけ，ファイルの更新時刻から 1 時間後を有効期限として，Expires ヘッダおよび Cache-Control ヘッダがレスポンスに追加されます。

```
ExpiresActive On
ExpiresByType text/html M3600
```

ExpiresByType の"M"指定は，ファイルの更新時刻を基準時刻としていることを示します。

4.11.3 注意事項

- ファイルの更新時刻を基準時刻として設定する場合，ディスク上のファイルにアクセスしないリクエスト（ステータス情報を表示するリクエストなど）では，更新時刻が存在しないため，Expires ヘッダおよび Cache-Control ヘッダは追加されません。
- HTTP Server で標準提供されていないモジュールを，LoadModule ディレクティブで組み込んだ場合，Expires ヘッダおよび Cache-Control ヘッダが操作されるおそれがあります。
- ヘッダカスタマイズ機能を同時に使用する場合，ヘッダカスタマイズ機能では Expires ヘッダおよび Cache-Control ヘッダを操作しないでください。
- レスポンスデータにすでに Expires ヘッダが設定済みの場合には，この機能では Expires ヘッダおよび Cache-Control ヘッダを付与しません。

4.12　複数の Web サーバ環境を生成する

4.12.1　複数の Web サーバ環境の生成（hwsserveredit コマンド）

バーチャルホストでなく，1 台のサーバマシンで複数の Web サーバを運用する場合，各 Web サーバで httpsd.conf の準備などの環境設定が必要です。この環境設定を補助するのが hwsserveredit コマンドです。

(1)　形式

```
hwsserveredit {-add|-add_woker|-add_prefork|-delete|-check} サーバ名
```

(2)　オペランド

- -add

  サーバ環境を新規作成する場合に指定します。コマンドは要求を受け付けると，servers ディレクトリの下に，指定されたサーバ名と同名のディレクトリを作成し，httpsd.conf と logs ディレクトリを作成します。また，Windows 版の場合は，サーバ名を用いて HTTP Server をサービスとして登録します。UNIX 版の場合は，httpsd.conf に worker MPM が設定されています。

- -add_worker

  UNIX 版の場合に使用できます。サーバ環境を新規作成する場合に指定します。コマンドは要求を受け付けると，servers ディレクトリの下に，指定されたサーバ名と同名のディレクトリを作成し，httpsd.conf と logs ディレクトリを作成します。また，httpsd.conf には worker MPM が設定されています。

- -add_prefork

  UNIX 版の場合に使用できます。サーバ環境を新規作成する場合に指定します。コマンドは要求を受け付けると，servers ディレクトリの下に，指定されたサーバ名と同名のディレクトリを作成し，httpsd.conf と logs ディレクトリを作成します。また，httpsd.conf には prefork MPM が設定されています。

- -delete

  サーバ環境を削除する場合に指定します。コマンドは要求を受け付けると，servers ディレクトリ下のサーバ名と同名のディレクトリを削除します。また，Windows 版の場合は，サーバ名のサービスを削除します。

- -check

  サーバ環境が構築済みかどうかを確認する場合に指定します。コマンドは要求を受け付けると，-add オペランド要求時に作成したリソースがある場合は，サーバ起動環境構築済みと判断します。

- **サーバ名**　～((1－(220－HTTP Server インストールディレクトリのパス長)，かつ 128 バイト以下))

サーバ単位にユニークな文字列を指定します。ただし，"Cosminexus HTTP Server"は指定できません。また，文字列中の空白を取り除くと，"CosminexusHTTPServer"となる文字列（"Cosminexus HTTPServer"など）は指定できません。

(3) 使用方法

(a) リソースの作成

各サーバのサーバ名を決定してから，hwsserveredit コマンドを実行します。"HWS1"というサーバ名の場合は，次を指定します。

```
hwsserveredit -add HWS1
```

hwsserveredit コマンドは，インストール時に作成されている servers ディレクトリの下に，ディレクトリとファイルを作成します。また，Windows 版の場合，ディレクトリとファイルの作成と同時に，サーバ名を使用してサービスを登録します。

次にディレクトリとファイルの構成を示します。

```
+httpsd
    └servers
        └HWS1
            ├conf
            │  └httpsd.conf
            └logs
```

(b) httpsd.conf の編集

作成されたリソースのうち，httpsd.conf のディレクティブ値を変更してください。

複数環境を生成する場合には，次のディレクティブの設定値を，ほかの環境と競合しないように変更してください。

- ServerName ディレクティブ
- Port ディレクティブまたは Listen ディレクティブ
- PidFile ディレクティブ
- HWSTraceIdFile ディレクティブ
- HWSTraceLogFile ディレクティブ

UNIX 版の場合は，さらに次のディレクティブの設定値を，環境に合わせて変更してください。

- User ディレクティブ
- Group ディレクティブ

そのほかのディレクティブは，hwsserveredit コマンドによって，変更しなくても起動できるように設定されています。運用に応じて必要があれば変更してください。

なお，各ディレクティブの詳細については，「6.2　ディレクティブの詳細」を参照してください。

(c)　サーバの起動

次の方法でサーバを起動してください。

- UNIX 版の場合

```
/opt/hitachi/httpsd/sbin/httpsd -f servers/HWS1/conf/httpsd.conf
```

- Windows 版の場合

```
”<Application Serverのインストールディレクトリ>¥httpsd¥httpsd.exe” -n HWS1 -k start
```

  または，コントロールパネルから HWS1 サービスを起動してください。

(d)　複数サーバ環境の設定

複数サーバ環境を設定する場合は，(a)から(c)の操作を繰り返してください。

(4)　注意事項

- コマンドは，管理者権限を持つユーザで実行してください。また，コマンドの場所は移動しないでください。
- サーバ名には，ASCII コードで指定してください。また，次に示す文字は指定できません。
  '¥'，'/'，':'，','，';'，'*'，'?'，'"'，'<'，'>'，'|'，'$'，'%'，'^'，'`'，'!'，'('，')'，'='，'+'，'{'，'}'，'@'，'['，']'，'~'，制御コード
- サーバ名には，ピリオドだけで構成される名称は指定できません。また，名称の前後に連続した空白を指定した場合には，それらを取り除きます。
- サービスへの登録時，スタートアップの種類は手動で登録します。
- 登録したサービスの表示名を変更しないでください。

4.13　イメージマップ

画像（画像のファイル）に複数のリンクを定義できます。その指定個所をクリックすると，その画像の座標位置やイメージマップファイル名が Web サーバに Web ブラウザから送信されます。Web サーバは，そのイメージマップファイルと座標位置から対応した URL を検索して，Web ブラウザに応答します。これをイメージマップといいます。

イメージマップを使用するには，imap-file ハンドラにマップファイル拡張子を対応付ける定義が必要です。

```
AddHandler imap-file .map
```

4.13.1　イメージマップファイルの文法

イメージマップデータの指定形式には次の 3 とおりの指定があります。

```
形状名称 指定値 座標
形状名称 指定値 ”説明文” 座標
形状名称 指定値 座標 ”説明文”
```

"**説明文**"はマップファイルメニュー表示時の説明文，**座標**は画像の座標を示しています。

形状名称を表 4-15 に，**指定値**を表 4-16 に示します。

表 4-15　形状名称と座標の指定形式

形状名称	意味	座標の指定	座標の説明
base	マップファイル内の相対 URL のベースを指定。	なし	−
default	poly，circle，rect に該当しないで，point 指定もない場合のリンクを指定する。		
poly	多角形を指定する。点を 3～100 個指定する。	x1,y1 x2,y2 … xn,yn	多角形の各座標位置（3～100 点の座標）
circle	円を指定する。中心点と，円周上の 1 点を指定する。	x1,y1 x2,y2	中心座標と円周上の 1 点の座標
rect	四角形を指定する。対角の 2 点を指定する。	x1,y1 x2,y2	対角の 2 点の座標
point	点を指定する。カーソルに最も近い point が有効になる。	x1,y1	点

(凡例)

−：該当しない。

注

座標の指定で(0,0)を含んでいる場合でも，イメージマップ画像の座標(0,0)をマウスポインタでポイントすると，マップファイルメニューが表示されます。

表 4-16　指定値

指定値	意味
URL	リンク先を指定する。相対ディレクトリの場合，base，ImapBase ディレクティブが有効になる。
map	マップファイルメニューを表示する。
menu	
referer	ステータスコード 302 Found を応答する。
nocontent	ステータスコード 204 No Content を応答する。base 以外に有効。
error	ステータスコード 500 Server Error を応答する。base 以外に有効。

4.13.2　イメージマップの定義例と注意事項

(1)　イメージマップの定義例

イメージマップを利用するための操作を次に示します。

1. httpsd.conf ファイルに，次に示すディレクティブを設定します。.map 拡張子名が URL で指定されたときにイメージマップを実行します。

```
AddHandler imap-file .map
```

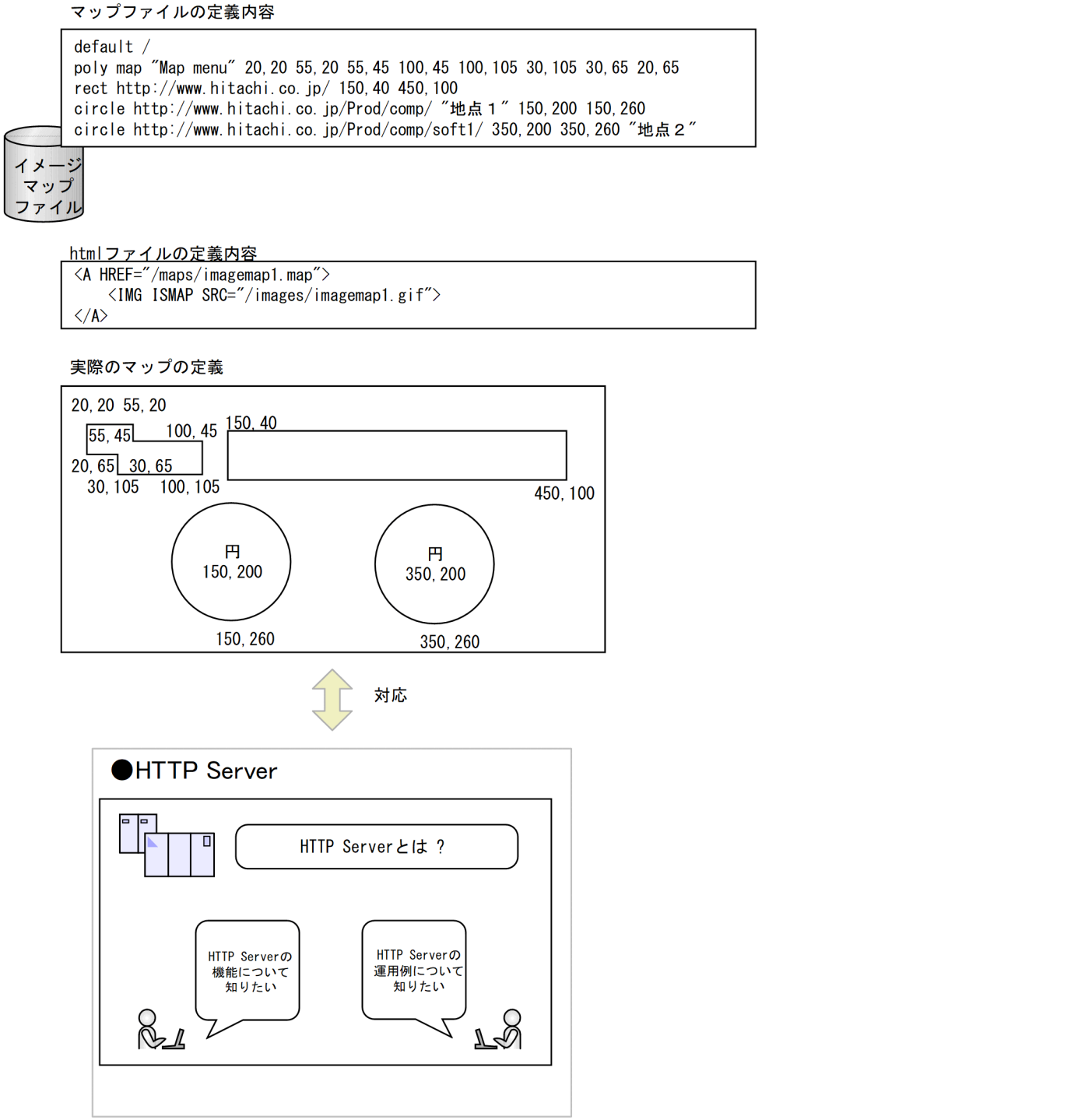
（ファイル拡張子.map に imap-file ハンドラを定義）

2. 上記で定義されたファイル拡張子のファイルにリンク先を定義します。
3. HTML 文書内に次の HTML 構文を記述します。

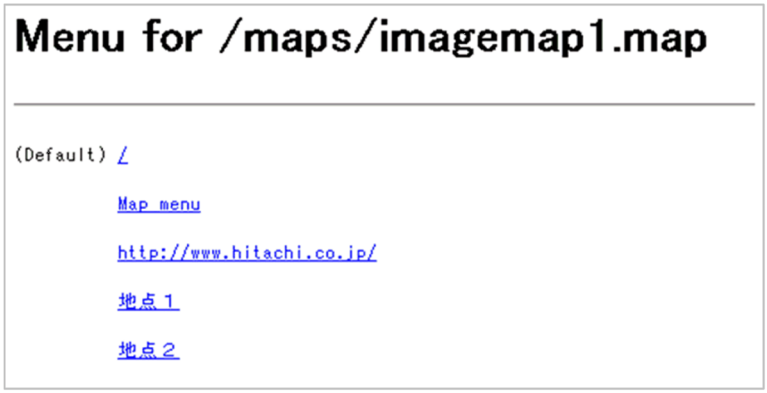
```
<A HREF="/ディレクトリ名/マップファイル名"><IMG SRC="画像データ名" ISMAP></A>
```

イメージマップファイルの定義例や，実際の表示例などを次に示します。

図 4-14　イメージマップファイルの定義例



この例で，poly で指定された部分をクリックすると，次に示すようなマップファイルメニューが表示されます。



(2) 注意事項

マップファイルメニューで使用している文字セットが，デフォルトの文字セット(ISO-8859-1)と異なる場合は，マップファイルメニューの表示において文字化けが発生します。この場合，HWSImapMenuCharsetディレクティブで，マップファイルメニューで使用している文字セットを指定してください。

4.14　IPv6 による通信

従来の IPv4 による通信だけでなく，IPv6 による通信ができます。IPv4 または IPv4 と IPv6 のデュアルスタック環境で動作させることができます。

4.14.1　サポート範囲

(1)　IPv6 に対応しているディレクティブ

Listen ディレクティブや VirtualHost ディレクティブなどのディレクティブに，IPv6 アドレスを指定することによって，IPv6 による通信や IPv6 アドレスに対応したバーチャルホストの指定などができます。

IPv6 に対応しているディレクティブを次に示します。

<VirtualHost>，AddIcon，AddIconByEncoding，AddIconByType，Allow from，CustomLog，DefaultIcon，Deny from，ErrorDocument，ExtendedStatus，HostnameLookups，HWSErrorLogClientAddr X-Forwarded-For，HWSSetEnvIfIPv6，ImapBase，ImapDefault，Listen，LogFormat，ProxyPass，ProxyPassReverse，QOSCookieDomain，QOSRedirect，Redirect，RedirectMatch，ServerAlias，ServerName，ServerSignature，SetEnvIf，SetEnvIfNoCase，TransferLog，UseCanonicalName

各ディレクティブの詳細については，「6.2　ディレクティブの詳細」を参照してください。

(2)　ディレクティブに IPv6 アドレスを指定するときの注意

ディレクティブに IPv6 アドレスを記述する場合は，「[IPv6 アドレス]」のように IPv6 アドレスを[　]で囲んで指定してください。また，ディレクティブに IPv6 アドレスとポート番号を併記する場合は，「[IPv6 アドレス]:ポート番号」のように IPv6 アドレスを[　]で囲み，「:」の後ろにポート番号を指定します。

ただし，次のディレクティブに IPv6 アドレスを記述する場合は，IPv6 アドレスを[　]で囲まないで指定してください。

- Allow from ディレクティブ
- Deny from ディレクティブ
- HWSSetEnvIfIPv6 ディレクティブ

なお，IPv6 アドレスを指定する場合は，グローバルユニキャストアドレスを指定してください。

(3)　制限事項

次の機能については，IPv6 には対応していません。

アドレス制限

BindAddress ディレクティブには，IPv6 アドレスは指定できません。

クライアントの確認

IdentityCheck ディレクティブに On を指定しても，IPv6 ソケットを使用している場合は機能しません。

環境変数の設定

SetEnvIf ディレクティブと SetEnvIfNoCase ディレクティブの正規表現には，IPv6 アドレスを指定できません。IPv6 アドレスを指定する場合は，HWSSetEnvIfIPv6 ディレクティブを使用してください。

4.14.2 IPv6 による通信の準備（httpsd.conf ファイルの編集）

IPv6 アドレスの指定

httpsd.conf ファイルに Port ディレクティブまたはポート番号だけの Listen ディレクティブを指定した場合は，IPv4 アドレスを使用したリクエストだけを受け付けます。IPv6 アドレスを使用する場合は，IPv6 アドレスを指定した Listen ディレクティブの設定が必要です。

例えば，次のように設定すると，IPv4 アドレスまたは IPv6 アドレスを使用したリクエストを受け付けるようになります。

```
Listen 80
Listen [::]:80
```

4.15　WebSocket による通信

4.15.1　mod_proxy_wstunnel モジュール

クライアントとバックエンドサーバが WebSocket 通信をするために，Web サーバはクライアントとバックエンドサーバの間で WebSocket 通信を中継します。WebSocket 通信の中継には mod_proxy モジュール，mod_proxy_http モジュール，および mod_proxy_wstunnel モジュールの組み込みが必要です。モジュールを組み込むには，コンフィグファイル（httpsd.conf）に，次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule proxy_module libexec/mod_proxy.so
LoadModule proxy_http_module libexec/mod_proxy_http.so
LoadModule proxy_wstunnel_module libexec/mod_proxy_wstunnel.so
```

- Windows 版

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

4.15.2　ディレクティブの設定方法

(1)　WebSocket 通信の転送

ここでは各アドレスを次のように仮定します。

Web サーバ：www.example.com

バックエンドサーバ：backend.example.com

ProxyPass の転送先 URL には，ws://ホスト名〔:ポート番号〕/を含む形で指定します。次のように ProxyPass ディレクティブを設定すると，クライアントからの”ws://www.example.com/ws/”というリクエストは”ws://backend.example.com/”というリクエストとしてバックエンドサーバに転送されます。

```
ProxyPass /ws/ ws://backend.example.com/
```

(2)　WebSocket ログ

WebSocket ログについては，「4.2.2　ログの採取方法」の「(5)　WebSocket ログ」を参照してください。

(3) サーバスレッド数の上限値変更

WebSocket 通信中は，クライアントまたはバックエンドサーバのどちらかのコネクションが切断されるまで，一つのサーバプロセスまたはスレッドが占有されます。そのため，WebSocket を利用しない場合に比べて多くのサーバスレッドを動作させておく必要があります。

(a) UNIX 版の場合

WebSocket を利用する場合，多くのサーバスレッドを必要とするため，worker MPM の使用を推奨します。worker MPM で一つのサーバプロセスで動作するサーバスレッド数は，ThreadsPerChild および ThreadLimit ディレクティブで指定します。サーバプロセス数の上限は ServerLimit ディレクティブで指定します。最大のサーバスレッド数は MaxClients ディレクティブで指定します。

(b) Windows 版の場合

動作させるサーバスレッドは，ThreadsPerChild および ThreadLimit ディレクティブで指定します。

4.15.3 注意事項

(1) サーバプロセス数およびスレッド数

Web サーバの起動には，サーバプロセス数およびスレッド数に応じたリソースを必要とします。大きな値を設定すると，リソース不足によって起動できない場合があります。

(2) Web サーバ停止方法

WebSocket 通信では，クライアントまたはバックエンドサーバのどちらかがコネクションを切断するまで，リクエスト処理が終了しません。そのため，計画停止を要求しても WebSocket 通信が終了しないかぎり，Web サーバを終了できません。Web サーバを停止させるには，次のどれかの方法で停止させます。

- HWSGracefulStopTimeout ディレクティブで指定したタイムアウトを使用して，時間経過後に計画停止させる。
- 計画停止要求後，通常停止をすることで，Web サーバを強制停止させる。
- 通常停止で Web サーバを停止させる。この場合，サーバプロセスの終了を一定秒数待ってから終了する（待ち時間はプラットフォームや設定によって異なる）。

(3) ProxyPass に指定するプロトコル

WebSocket 通信を転送する場合，ProxyPass ディレクティブの転送先 URL には ws://ホスト名〔:ポート番号〕/を含む形で指定してください。転送先 URL のプロトコルに wss は指定できません。

4.16　アプリケーションサーバとの連携

アプリケーションサーバとの連携については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」を参照してください。

4.17　HTTP/2 プロトコル通信機能

Web サーバでは，クライアントから送信された HTTP/2 プロトコルのリクエストを処理することができ，平文と暗号の両方の通信ができます。HTTP/2 の接続方法には次の 3 種類があります。

- TLS の ALPN 拡張による接続（TLS 使用時だけ）
- ダイレクト接続
- Upgrade ヘッダによる接続

また，リバースプロキシ機能でバックエンドサーバと通信する際，HTTP/2 プロトコルを使用できます。ただし，バックエンドサーバとの通信では暗号通信で HTTP/2 プロトコルは使用できません。

4.17.1　mod_http2 モジュールの組み込み

クライアントと Web サーバが HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。

mod_http2 モジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule http2_module libexec/mod_http2.so
```

- Windows 版

```
LoadModule http2_module modules/mod_http2.so
```

4.17.2　mod_proxy_http2 モジュールの組み込み

リバースプロキシ機能でバックエンドサーバと HTTP/2 プロトコルを使用するためには，mod_proxy モジュール，mod_proxy_http2 モジュールの組み込みが必要です。モジュールを組み込むには，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定します。

- UNIX 版

```
LoadModule proxy_module libexec/mod_proxy.so
LoadModule proxy_http2_module libexec/mod_proxy_http2.so
```

- Windows 版

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http2_module modules/mod_proxy_http2.so
```

リバースプロキシ機能を使って HTTP/2 プロトコルでバックエンドサーバにリクエストを転送する設定例を次に示します。

ここでは各アドレスを次のように仮定しています。

- リバースプロキシ：www.example.com
- バックエンドサーバ：backend.example.com

```
ProxyPass /news/ h2c://backend.example.com/
ProxyPassReverse /news/ http://backend.example.com/
```

このように ProxyPass ディレクティブを設定すると，クライアントからの"http://www.example.com/news/oct-2001"というリクエストは"http://backend.example.com/oct-2001"というリクエストに変更されます。変更されたリクエストは HTTP/2 プロトコルを使用して，バックエンドサーバに送信されます。

バックエンドサーバでリダイレクトが指示された場合，バックエンドサーバからのレスポンスの Location ヘッダにバックエンドサーバのアドレスが記載されます。これをそのままクライアントに応答すると，クライアントはリダイレクトをリバースプロキシではなく，直接バックエンドサーバにリクエストします。そこで，ProxyPassReverse ディレクティブを上記のように指定することで，リダイレクトリクエストもリバースプロキシを通るリクエストになります。

リバースプロキシ機能を使用してバックエンドサーバに転送する場合は，バックエンドサーバ側でも HTTP/2 リクエストを受け付けるための設定が必要です。バックエンドサーバに Cosminexus の NIO HTTP サーバを使用する場合には，NIO HTTP サーバの webserver.http2.enabled キーに true の指定が必要です。詳細はマニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

4.17.3 ワーカスレッド

サーバスレッドがリクエストを受け付けたあと，そのリクエストが HTTP/2 プロトコルである場合は，ワーカスレッドと呼ばれるスレッドで HTTP/2 プロトコルのストリームに相当するリクエスト処理をします。

HTTP/2 通信を使用する場合，HTTP/2 リクエストを処理したサーバスレッド，ワーカスレッド，およびストリームを把握するために，アクセスログに HTTP/2 に関連する情報を出力することを推奨します。その設定例を次に示します。

- UNIX 版

```
LogFormat "%h %l %u %t ¥"%r¥" %>s %b %T %P %{tid}P ¥"%{hws_ap_root}n¥" H2_STREAM_TAG=%{H2_STREAM_TAG}e HWS_H2_WORKER_THREAD_ID=%{HWS_H2_WORKER_THREAD_ID}n" hws_std
```

- Windows 版

```
LogFormat "%h %l %u %t ¥"%r¥" %>s %b %T %{hws_thread_id}P ¥"%{hws_ap_root}n¥" H2_STREAM_TAG=%{H2_STREAM_TAG}e HWS_H2_WORKER_THREAD_ID=%{HWS_H2_WORKER_THREAD_ID}n" hws_std
```

ワーカスレッドは，サーバプロセス内に存在し，コンフィグファイルの設定に従って増減します。ワーカスレッドは，サーバプロセスが生成される際に H2MinWorkers ディレクティブで指定した数だけ生成されます。このとき生成されたワーカスレッドはサーバプロセス終了時まで生存します。運用時にワーカスレッドが不足した場合は，H2MaxWorkers ディレクティブで指定した数を上限としてワーカスレッドが生成されます。リクエスト処理が完了したあと，H2MaxWorkerIdleSeconds ディレクティブで指定した時間が経過するとワーカスレッドは終了しますが，サーバプロセス生成時に生成されたワーカスレッドは終了しません。

ワーカスレッドはサーバプロセスごとに生成されるため，最小数は H2MinWorkers ディレクティブ指定値×サーバプロセス数，最大数は H2MaxWorkers ディレクティブ指定値×サーバプロセス数となります。

4.17.4 制限事項

次の機能は，HTTP/2 プロトコル通信機能では使用できません。

- サーバープッシュ機能は使用できません。クライアントがリクエストしたコンテンツだけを返します。
- UNIX 版の prefork MPM を使用する場合は，HTTP/2 プロトコル通信機能は使用できません。worker MPM を使用してください。
- mod_hws_qos モジュールによる流量制限機能は使用できません。
- HWSNotModifiedResponseHeaders ディレクティブによってレスポンスヘッダを付加できません。
- HWSGracefulStopLog ディレクティブによって，計画停止時に強制停止させた HTTP/2 通信のリクエスト情報は，エラーログファイルに出力できません。
- HTTP/2 通信時に WebSocket 通信は開始できません。
- KeepAlive ディレクティブによって，持続型接続（KeepAlive）は無効にできません。KeepAlive ディレクティブの設定に関係なく，コネクションが持続します。コネクションが持続する時間は，KeepAliveTimeout ディレクティブ値に従います。

4.17.5 注意事項

HTTP/2 プロトコル通信機能を使用する場合，コンフィグファイル（httpsd.conf）に次に示すディレクティブを指定することを推奨します。

```
HWSSuppressModuleTrace mod_http2.c
```

5 SSL による認証，暗号化

この章では，SSL による認証，暗号化について説明します。

5.1　SSL で認証，暗号化する

HTTP Server は SSL（Secure Sockets Layer）プロトコルを使用すれば，送受信する情報を保全できます。この製品は TLS（Transport Layer Security）バージョン 1.0，1.1，1.2，および 1.3 に対応しており，SSL サーバ認証，SSL クライアント認証ができます。SSL の機能を次に示します。

- 通信相手を確認し，特定するために認証します。
- サーバとクライアントの間で転送するデータを暗号化します。
- 転送中に改ざんされたデータを検出します。

これらの機能は SSL 関連コマンドで作成された秘密鍵と，認証局（CA）が発行した証明書を Web サーバにインストールすれば，利用できます。証明書には，SHA-2 アルゴリズムで署名された証明書，ワイルドカード証明書，マルチドメイン証明書も利用できます。また，RSA 暗号および楕円曲線暗号に対応しています。

5.1.1　SSL 通信のための準備

SSL による認証や，データの暗号化を使用するには，Web サーバに秘密鍵と認証局（CA）が発行した証明書をインストールする必要があります。

手順を次に示します。

1. 秘密鍵の作成

   openssl.bat genrsa コマンドまたは openssl.sh genrsa コマンドを使用して，Web サーバの秘密鍵を作成します。

2. CSR（証明書発行要求）の作成

   openssl.bat reqgen コマンドまたは openssl.sh reqgen コマンドを使用して，CSR を作成します。

3. CA へ CSR を送付

   2.で作成した CSR を CA に送付します。

4. 証明書の入手

   PEM 形式の証明書を CA から入手します。

5. httpsd.conf ファイルの編集（ディレクティブの定義）

   SSL を有効にするために，SSLEngine ディレクティブに On を指定します。CA から入手した PEM 形式の証明書は SSLCertificateFile ディレクティブ，Web サーバの秘密鍵は SSLCertificateKeyFile ディレクティブに指定します。

   （例）SSL を有効にして，PEM 形式の証明書および Web サーバの秘密鍵を定義

   - UNIX 版の場合

```
SSLEngine On
SSLCertificateFile /opt/hitachi/httpsd/conf/ssl/server/httpsd.pem
SSLCertificateKeyFile /opt/hitachi/httpsd/conf/ssl/server/httpsdkey.pem
```

- Windows 版の場合

```
SSLEngine On
SSLCertificateFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server/httpsd.pem"
SSLCertificateKeyFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server/httpsdkey.pem"
```

SSL を使用して通信する場合，Web ブラウザからは，https://でリクエストします。ポート番号を省略した場合，SSL の標準では 443 ポートを使用します。したがって，Listen ディレクティブで 443 ポートを指定するのが一般的です。

6. Web サーバの再起動

   httpsd.conf ファイルの定義を有効にするには，Web サーバを再起動する必要があります。ただし，SSLCertificateKeyFile ディレクティブの設定を変更した場合は，いったん，Web サーバを停止後，起動し直してください。

SSL を無効にする場合には，5.の指定を無効化して再起動します。

5.1.2 SSL 通信の手順

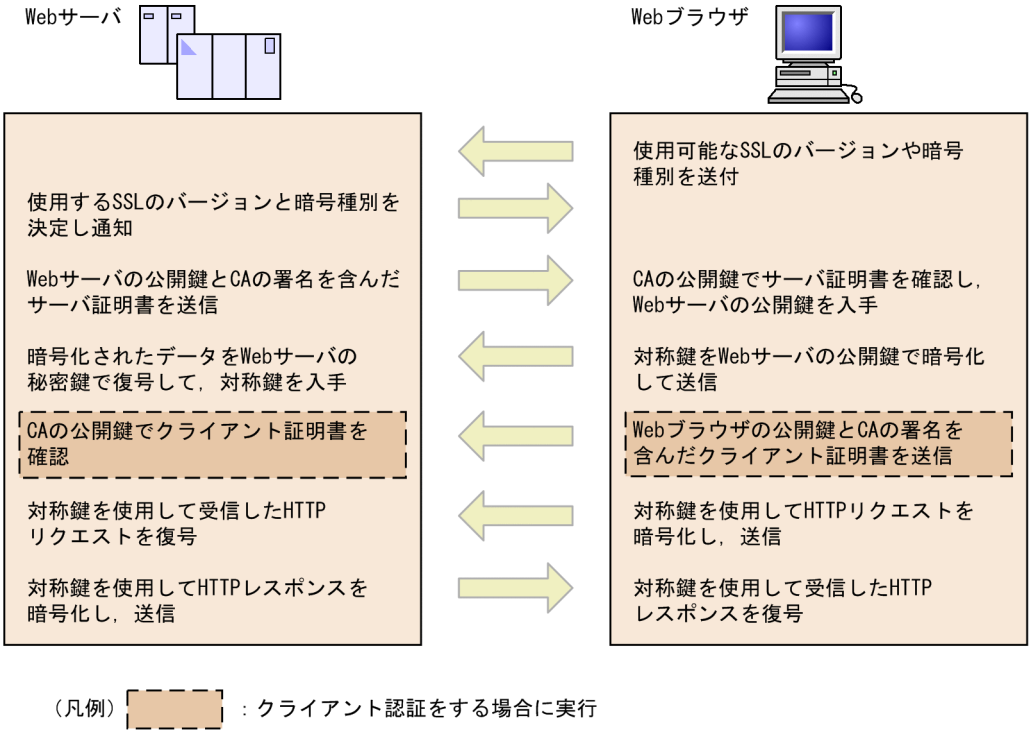
SSL 通信の手順を次に示します。2～6 の手順を SSL での**ハンドシェイク**と呼びます。

1. Web ブラウザから，https://へのリクエストを実行します。
2. Web ブラウザは，使用できる SSL のバージョンや暗号種別を示すデータを Web サーバに送付します。
3. Web サーバは，使用する SSL のバージョンと暗号種別を決定し，Web ブラウザに通知します。また，公開鍵と CA の署名が入った証明書を Web ブラウザに送付します。
4. Web ブラウザは Web ブラウザが持っている CA の公開鍵を使用して，送付された証明書が改ざんされていないことを確認して，Web サーバの公開鍵を入手します。
5. Web ブラウザは，通信で Web サーバと共有する対称鍵を作成し，Web サーバの公開鍵で暗号化して送信します。また，Web ブラウザが持っている証明書をクライアント認証のために Web サーバに提示する場合，証明書を送信します。

   Web サーバの公開鍵で暗号化したデータは，対になる秘密鍵がないと復号できません。つまり，データ送信先の Web サーバだけがデータの内容を解読できます。
6. Web サーバは受信した対称鍵を Web サーバの秘密鍵で復号し，入手します。Web ブラウザからの証明書を受信した場合は，証明書を確認します。
7. Web ブラウザと，Web サーバの間で共有された対称鍵を使用して，HTTP リクエストまたはレスポンスを暗号化し，送受信します。

SSL 通信でのリクエスト処理を次に示します。

図 5-1　SSL 通信でのリクエスト処理



ハンドシェイクの際に，クライアント側と Web サーバ側の両方に有効であり，最も強い強度の暗号が選択されます。Web サーバ側の暗号種別は，SSLCipherSuite ディレクティブで指定します。この指定で，常にすべての暗号種別を有効にしておけば，クライアントが持つ最も強い強度の暗号で通信できることになります。

5.1.3　SSL クライアント認証の準備

SSL クライアント認証をする場合は，「5.1.1　SSL 通信のための準備」の 1～5 に加えて，次の操作をして，再起動してください。

1. クライアント証明書を Web ブラウザにインストール

   証明書の発行に使用する CA の指示に従い，Web ブラウザにクライアント証明書をインストールします。

2. CA の証明書の入手

   クライアント証明書を発行した CA の証明書（PEM 形式）を入手します。

3. httpsd.conf ファイルの編集（ディレクティブの定義）

   SSL クライアント認証を有効にするために，SSLVerifyClient ディレクティブに require を指定し，SSLVerifyDepth ディレクティブに 1 以上を指定します。CA から入手した PEM 形式の証明書は，SSLCACertificateFile または SSLCACertificatePath ディレクティブに指定します。

   注

   SSLCACertificatePath ディレクティブは Windows 版では指定できません。

5.1.4　証明書の有効性の検証

SSL クライアント認証のときに，クライアント証明書の検証だけでなく，その時点のクライアント証明書の有効性を CRL（Certificate Revocation List）を使用して検証できます。CRL は，検証したいクライアント証明書を発行した CA から入手します。

(1)　CRL ファイルの形式

CRL は PEM 形式のファイルを使用します。

（例）PEM 形式の CRL

```
<Application Serverのインストールディレクトリ>¥httpsd¥conf¥ssl¥crl>type crl.pem
-----BEGIN X509 CRL-----
MIIBGDCBwwIBATANBgkqhkiG9w0BAQQFADB0MQswCQYDVQQGEwJKUDERMA8GA1UE
CBMIS2FuYWdhd2ExFTATBgNVBAcTDFlva29oYW1hLXNoaTERMA8GA1UEChMITE9D
QUwtQ0ExDDAKBgNVBAsTA2NhMTEaMBgGA1UEAxMRY2ExLmhpdGFjaGkuY28uanAX
DTAxMDgyOTA0NDIzMFoXDTAxMDgzMDA1NTIzMFowGzAZAghx2Sa8AAAAARcNMDEw
ODI4MDQ1MTI5WjANBgkqhkiG9w0BAQQFAANBAJorY7DUJ91uthNlAA+PT6zw6rVo
uZLFeYZPNVXgF217YOCtJtKDT+16bR5kgk0p/1xIbgReshjMNTmXPqARNjE=
-----END X509 CRL-----
```

注

旧バージョンの HTTP Server で DER 形式の CRL を使用していた場合は，PEM 形式に変換してください。

(2)　HTTP Server への CRL 適用方法

CRL を使用してクライアント証明書の有効性を検証する場合は，「5.1.3　SSL クライアント認証の準備」に加えて，次の操作をして，Web サーバを再起動してください。

1. CRL の入手
   各 CA の CRL 配布点から CRL ファイルを入手し，適切なディレクトリに格納します。
2. httpsd.conf の編集（ディレクティブの定義）
   CRL を有効にするために，SSLCARevocationCheck ディレクティブに leaf を指定して，SSLCARevocationFile ディレクティブに CRL ファイルを設定します。
3. Web サーバを起動または再起動します。
4. 既存の CRL を更新する場合には，古い CRL を新規 CRL で上書きしたあと，Web サーバを再起動します。

(3)　CRL を使用したクライアント証明書検証

CRL を使用したクライアント証明書検証では次の項目を確認します。

- CRL 自体が有効であるかどうか。
- 次回発行日より前かどうか。

- クライアント証明書のシリアル番号が記載されていないかどうか。

(a) CRL クライアント証明書検証でクライアント証明書が有効と判定される条件

CRL クライアント証明書検証でクライアント証明書が有効と判定される条件には次に示すものがあります。

- 証明書を発行した CA が発行した CRL を読み込んでいない場合。
- 現在時刻が CRL の次回発行日より前であり，かつ該当する接続のクライアント証明書のシリアル番号が CRL に記載されていない場合。
- 現在時刻が CRL 発行日よりあとで，次回発行日が指定されてなく，かつ CRL に該当する接続のクライアント証明書のシリアル番号が記載されていない場合。

(b) CRL クライアント証明書検証でクライアント証明書が無効と判定される条件

CRL クライアント証明書検証でクライアント証明書が無効と判定される条件には次に示すものがあります。

- CRL が有効でない場合。
- 該当する接続のクライアント証明書のシリアル番号が CRL に記載されている場合。

5.2　証明書の取得

RSA 暗号，および楕円曲線暗号を利用した SSL 通信をするためには，利用する暗号種別に応じて，秘密鍵と認証局（CA）が発行した証明書を用意する必要があります。

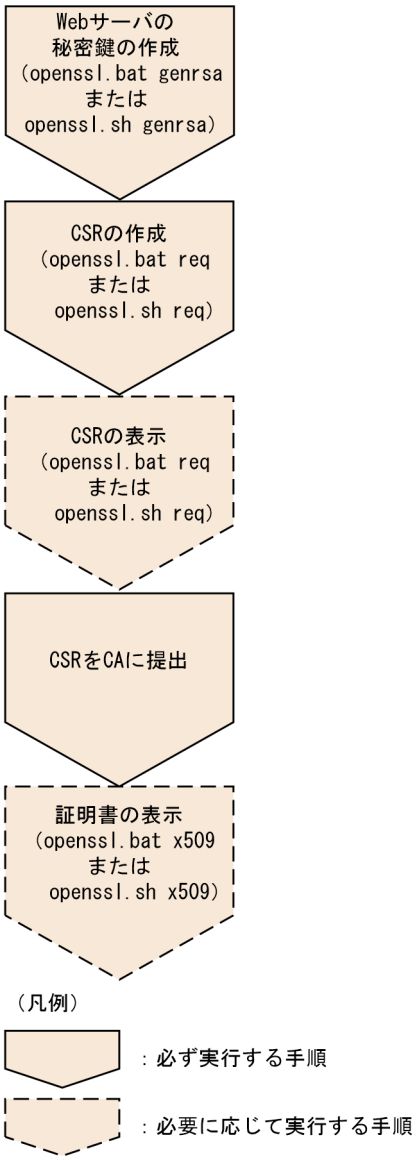
5.2.1　証明書取得手順

暗号種別ごとに証明書を取得するまでの手順を次に示します。

(1)　RSA 暗号を利用する場合

RSA 暗号を利用する場合に，証明書を取得するまでの手順を次に示します。証明書取得に必要な Web サーバの秘密鍵や証明書発行要求（CSR）は，Windows の場合は openssl.bat コマンド，UNIX の場合は openssl.sh コマンドで作成します。

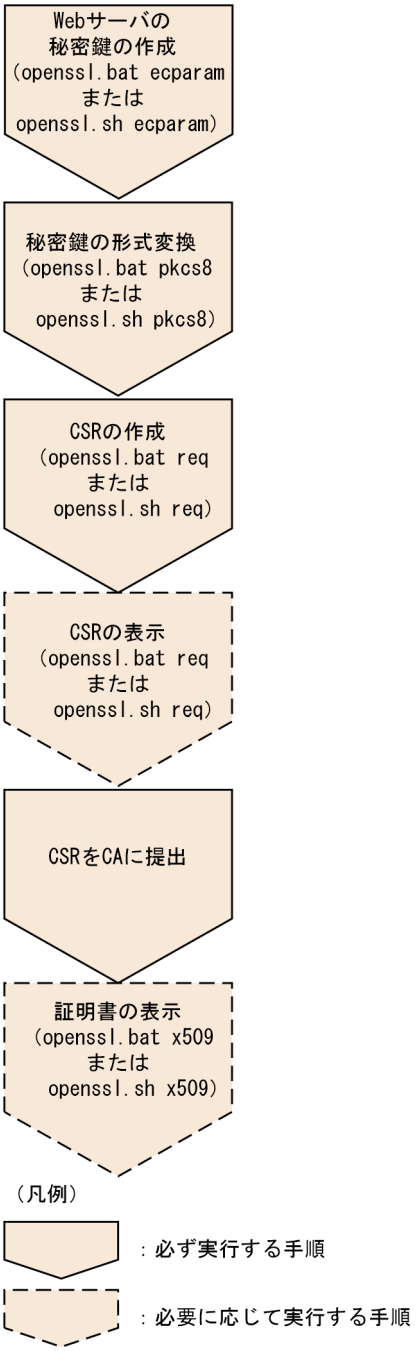
図 5-2　証明書取得手順



上の図の手順で CA の署名済みの証明書ファイルを取得したあと，証明書ファイルの"-----BEGIN CERTIFICATE-----"から，"-----END CERTIFICATE----"の部分を，ディレクティブに指定する証明書ファイルとして保存します（標準提供の httpsd.conf では httpsd.pem）。秘密鍵のファイルを SSLCertificateKeyFile ディレクティブに，保存した証明書ファイルを SSLCertificateFile ディレクティブに定義することで，SSL が利用できるようになります。なお，SSLCertificateKeyFile ディレクティブには，PKCS#1 または PKCS#8 に準拠した形式の秘密鍵のファイルを定義してください。

(2)　楕円曲線暗号を利用する場合

楕円曲線暗号を利用する場合に，証明書を取得するまでの手順を次に示します。証明書取得に必要な Web サーバの秘密鍵や証明書発行要求（CSR）は，Windows の場合は openssl.bat コマンド，UNIX の場合は openssl.sh コマンドで作成します。



上の図の手順で CA の署名済みの証明書ファイルを取得したあと，証明書ファイルの"-----BEGIN CERTIFICATE-----"から，"-----END CERTIFICATE----"の部分を，ディレクティブに指定する証明書ファイルとして保存します（標準提供の httpsd.conf では httpsd.pem）。秘密鍵のファイルを SSLCertificateKeyFile ディレクティブに，保存した証明書ファイルを SSLCertificateFile ディレクティブに定義することで，SSL が利用できるようになります。なお，SSLCertificateKeyFile ディレクティブには，PKCS#8 に準拠した形式の秘密鍵のファイルを定義してください。

(3) 注意事項

RSA 暗号と楕円曲線暗号の両方の暗号スイートを同時に利用する場合は，RSA 暗号と楕円曲線暗号の両方の秘密鍵と証明書をディレクティブに定義してください。

5.2.2　Web サーバの秘密鍵の作成

openssl.bat コマンドまたは openssl.sh コマンドを使用して，Web サーバの秘密鍵を作成します。暗号種別ごとに Web サーバの秘密鍵を作成する方法を示します。

(1)　RSA 暗号を利用する場合（openssl.bat genrsa コマンドまたは openssl.sh genrsa コマンド）

openssl.bat genrsa コマンドまたは openssl.sh genrsa コマンドを使用して，Web サーバの秘密鍵を作成します。作成した Web サーバの秘密鍵のファイルは，SSLCertificateKeyFile ディレクティブに指定します。

秘密鍵は，PKCS#1 に準拠した形式で作成されます。

(a)　形式

Windows の場合

```
openssl.bat genrsa -rand ファイル名 -out 鍵ファイル 〔1024｜2048｜4096〕
```

UNIX の場合

```
openssl.sh genrsa -rand ファイル名〔:ファイル名…〕 -out 鍵ファイル 〔1024｜2048｜4096〕
```

(b)　オペランド

- -rand ファイル名（Windows の場合）
- -rand ファイル名〔:ファイル名…〕（UNIX の場合）
  乱数生成に利用する任意のファイルを指定します。乱数生成用のファイルは，十分大きい適当なファイルを指定してください。Windows 版では，ファイル名は一つだけ指定できます。複数指定はできません。
- -out　鍵ファイル
  Web サーバの秘密鍵を出力するファイルを指定します。
- 〔1024 ｜ 2048 ｜ 4096〕
  作成する Web サーバの秘密鍵のビット長を指定します。

(c)　使用例

Web サーバの秘密鍵 httpsdkey.pem を PKCS#1 に準拠した形式で作成します。

Windows の場合

```
openssl.bat genrsa -rand file1 -out httpsdkey.pem 2048
```

file1：任意のファイル

UNIX の場合

```
openssl.sh genrsa -rand file1:file2:file3:file4:file5 -out httpsdkey.pem 2048
```

file1，file2，file3，file4，file5：任意のファイル

(2) 楕円曲線暗号を利用する場合（openssl.bat ecparam コマンドまたは openssl.sh ecparam コマンド）

openssl.bat ecparam コマンドまたは openssl.sh ecparam コマンドを使用して，Web サーバの秘密鍵を作成します。作成した Web サーバの秘密鍵のファイルは，SSLCertificateKeyFile ディレクティブに指定します。

(a) 形式

Windows の場合

```
openssl.bat ecparam -genkey -noout -rand ファイル名 -name 楕円曲線名 -out 鍵ファイル
```

UNIX の場合

```
openssl.sh ecparam -genkey -noout -rand ファイル名〔:ファイル名 …〕 -name 楕円曲線名 -out
鍵ファイル
```

(b) オペランド

- -rand ファイル名（Windows の場合）
- -rand ファイル名〔:ファイル名…〕（UNIX の場合）

  乱数生成に利用する任意のファイルを指定します。乱数生成用のファイルは，十分大きい適当なファイルを指定してください。Windows 版では，ファイル名は一つだけ指定できます。複数指定はできません。
- -name 楕円曲線名

  秘密鍵の生成に利用する楕円曲線名を指定します。次の楕円曲線暗号のうちのどれか一つを指定してください。
  - secp384r1
  - secp521r1
  - prime256v1
  - P-256
  - P-384

- P-521

- -out　鍵ファイル
  Web サーバの秘密鍵を出力するファイルを指定します。

(c)　使用例

楕円曲線暗号を利用した秘密鍵 httpsdkey.pem を作成します。Web サーバで秘密鍵を利用する場合は，httpsdkey.pem を PKCS#8 形式に変換してください。

Windows の場合

```
openssl.bat ecparam -genkey -noout -rand file1 -name P-256 -out httpsdkey.pem
```

file1：任意のファイル

UNIX の場合

```
openssl.sh ecparam -genkey -noout -rand file1:file2:file3:file4:file5 -name P-256 -out https
dkey.pem
```

file1，file2，file3，file4，file5：任意のファイル

5.2.3　Web サーバの秘密鍵の形式変換（openssl.bat pkcs8 コマンドまたは openssl.sh pkcs8 コマンド）（楕円曲線暗号使用時）

openssl.bat pkcs8 コマンドまたは openssl.sh pkcs8 コマンドを使用して，Web サーバの秘密鍵の形式を変換します。

(1)　形式

Windows の場合

```
openssl.bat pkcs8 -topk8 -in 入力ファイル -out 出力ファイル -nocrypt
```

UNIX の場合

```
openssl.sh pkcs8 -topk8 -in 入力ファイル -out 出力ファイル -nocrypt
```

(2)　オペランド

- -in 入力ファイル
  変換前の秘密鍵ファイルを指定します。
- -out 出力ファイル

変換後の秘密鍵ファイルを指定します。

- -nocrypt
  変換後の秘密鍵を暗号化しません。

(3) 使用例

PKCS#1 形式の Web サーバの秘密鍵 httpsdkey.pem を，PKCS#8 形式の Web サーバの秘密鍵 httpsdkey2.pem に変換します。

Windows の場合

```
openssl.bat pkcs8 -topk8 -in httpsdkey.pem -out httpsdkey2.pem -nocrypt
```

UNIX の場合

```
openssl.sh pkcs8 -topk8 -in httpsdkey.pem -out httpsdkey2.pem -nocrypt
```

5.2.4 証明書発行要求（CSR）の作成（openssl.bat req コマンドまたは openssl.sh req コマンド）

openssl.bat req コマンドまたは openssl.sh req コマンドを使用して，証明書発行要求（CSR）を作成します。ここで作成した CSR ファイルを CA に提出して，署名済みの証明書を発行してもらいます。CSR は，PKCS#10 に準拠した形式で作成されます。

(1) 形式

Windows の場合

```
openssl.bat req -new 〔-sha1 | -sha224 | -sha256 | -sha384 | -sha512〕 -key 鍵ファイル -out CSR ファイル
```

UNIX の場合

```
openssl.sh req -new 〔-sha1 | -sha224 | -sha256 | -sha384 | -sha512〕 -key 鍵ファイル -out CSR ファイル
```

(2) オペランド

- 〔-sha1 | -sha224 | -sha256 | -sha384 | -sha512〕
  CSR 作成時に使用する署名アルゴリズムを指定します。
  -sha1：sha1WithRSAEncryption を使用します。
  -sha224：sha224WithRSAEncryption を使用します。

-sha256：sha256WithRSAEncryption を使用します。
-sha384：sha384WithRSAEncryption を使用します。
-sha512：sha512WithRSAEncryption を使用します。

- -key 鍵ファイル
  Web サーバの秘密鍵のファイルを指定します。
- -out CSR ファイル
  作成した CSR を出力するファイルを指定します。

(3) 使用例

証明書発行要求（CSR）httpsd.csr を作成します。

Windows の場合

```
openssl.bat req -new -sha1 -key httpsdkey.pem -out httpsd.csr
```

UNIX の場合

```
openssl.sh req -new -sha1 -key httpsdkey.pem -out httpsd.csr
```

httpsdkey.pem：鍵ファイル

httpsd.csr：CSR ファイル

5.2.5 証明書発行要求（CSR）の内容表示（openssl.bat req コマンドまたは openssl.sh req コマンド）

証明書発行要求（CSR）の内容を表示します。

(1) 形式

Windows の場合

```
openssl.bat req -in CSRファイル -text
```

UNIX の場合

```
openssl.sh req -in CSRファイル -text
```

(2) オペランド

- -in CSR ファイル

表示する CSR ファイルを指定します。

(3) 使用例

Windows の場合

```
openssl.bat req -in httpsd.csr -text
```

UNIX の場合

```
openssl.sh req -in httpsd.csr -text
```

httpsd.csr：表示する CSR ファイル

5.2.6 証明書の内容表示（openssl.bat x509 コマンドまたは openssl.sh x509 コマンド）

証明書ファイルの内容を表示します。

"-----BEGIN CERTIFICATE-----"から，"-----END CERTIFICATE----"の証明書ファイルの内容を表示します。

(1) 形式

Windows の場合

```
openssl.bat x509 -in 証明書ファイル -text
```

UNIX の場合

```
openssl.sh x509 -in 証明書ファイル -text
```

(2) オペランド

- -in 証明書ファイル
  表示する証明書ファイルを指定します。

(3) 使用例

Windows の場合

```
openssl.bat x509 -in httpsd.pem -text
```

UNIX の場合

```
openssl.sh x509 -in httpsd.pem -text
```

httpsd.pem：表示する証明書ファイル

5.2.7 証明書の形式変換（openssl.bat x509 コマンドまたは openssl.sh x509 コマンド）

証明書の形式を変換します。必要に応じて使用します。

(1) 形式

Windows の場合

```
openssl.bat x509 -inform 入力形式 -outform 出力形式 -in 入力ファイル -out 出力ファイル
```

UNIX の場合

```
openssl.sh x509 -inform 入力形式 -outform 出力形式 -in 入力ファイル -out 出力ファイル
```

(2) オペランド

- -inform 入力形式
  入力形式：{DER | PEM}
- -outform 出力形式
  出力形式：{DER | PEM}
- -in 入力ファイル
  変換前の証明書ファイルを指定します。
- -out 出力ファイル
  変換後の証明書ファイルを指定します。

5.2.8 ハッシュリンクの作成（UNIX 版）（openssl.sh x509 コマンド）

証明書の妥当性チェックのために，証明書を発行した CA の証明書を SSLCACertificateFile ディレクティブまたは SSLCACertificatePath ディレクティブで指定します。SSLCACertificatePath ディレクティブには，証明書発行元の CA の証明書をポイントするハッシュ値を使用したシンボリックリンク（ハッシュリンク）を格納したディレクトリを指定します。ハッシュ値は openssl.sh x509 コマンドで作成します。

SSLCACertificatePath ディレクティブを指定すると，Web サーバでの証明書の検索はハッシュ値を用いて効率良く実行できます。したがって，CA の証明書が多い場合は，SSLCACertificateFile ディレクティブよりも SSLCACertificatePath ディレクティブを推奨します。なお，ハッシュ値は一つの証明書に一つである必要があるので，ハッシュリンク作成時には，複数の証明書が混在したファイルは指定できません。

SSLCACertificatePath ディレクティブで指定するハッシュリンクディレクトリ内のシンボリックリンク生成時には，ハッシュ値に.0 を付ける必要があります。また，SSLCACertificatePath ディレクティブで指定するディレクトリは，User，Group ディレクティブで指定したユーザでアクセスできるように，ディレクトリに読み込み権限，実行権限を設定してください。

(1) 形式

```
openssl.sh x509 -noout -hash -in CAの証明書ファイル
```

(2) オペランド

- -in CA の証明書ファイル

  ハッシュリンク値を作成する CA の証明書ファイルを指定します。

(3) 使用例

ハッシュリンクのディレクトリおよび CA の証明書が次に示すディレクトリ，ファイルの場合の例を示します。

/opt/hitachi/httpsd/conf/ssl/cacerts：ハッシュリンクディレクトリ

/opt/hitachi/httpsd/conf/ssl/cacert/cacert.pem：CAの証明書

```
cd /opt/hitachi/httpsd/conf/ssl/cacerts
ln -s /opt/hitachi/httpsd/conf/ssl/cacert/cacert.pem `
openssl.sh x509 -noout -hash  -in /opt/hitachi/httpsd/conf/ssl/cacert/cacert.pem`.0
```

これによって，/opt/hitachi/httpsd/conf/ssl/cacert/cacert.pem についてのハッシュリンク xxxxxxxx.0 が作成されます。

5.2.9 openssl.bat コマンドおよび openssl.sh コマンドの使用例

openssl.bat コマンドおよび openssl.sh コマンドの使用例を示します。なお，この例で使用している Common Name などは架空のものです。

(1) 秘密鍵の作成（openssl.bat コマンドまたは openssl.sh コマンド）

(a) RSA 暗号を利用する場合

RSA 暗号の秘密鍵を生成する場合のコマンドの使用例を次に示します。

使用例（Windows の場合）

```
# openssl.bat genrsa -rand file -out httpsdkey.pem 1024
Generating RSA private key, 1024 bit long modulus
..............++++++
..............................++++++
e is 65537 (0x10001)
#
```

注

11-00-10 以降，11-10-10 以降，11-20-10 以降では，メッセージ（Generating…）は出力されません。

使用例（UNIX の場合）

```
# openssl.sh genrsa -rand file -out httpsdkey.pem 1024
Generating RSA private key, 1024 bit long modulus
..............++++++
..............................++++++
e is 65537 (0x10001)
#
```

注

11-00-10 以降，11-10-10 以降，11-20-10 以降では，メッセージ（Generating…）は出力されません。

RSA 暗号の**秘密鍵の内容**

RSA 暗号の秘密鍵の内容を次に示します。

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCwFMAIeJw8jhLyV3wog/0cjs2KQCRCumwgRSCAaDuKGgjlCsY9
/7z4evgK91sKvSVBcNFP/CemK7e8GyorwMT1CbJR7HD6D+LJ7ksr9zxl7vrUohCu
C/EW2ut0ZSVe9X4chfQc4RLvmkcmiMZuUKa5zP2kiOL9Ug5u3VksS/hWGwIDAQAB
AoGBAJysCeY/svyqia86Ko4+StPDOJ2/zsPU7mqUN4Qpunh6C9oIiTYXPER33gab
61UV0XV19bhOq9TOZ3CnVxGRN206PnXWA8E2M1g+yFnHSTmrF/noXYYL88L57ZKP
+eE0H5otxJC2E5wdDTlNJEtfv2PxLkNQe0czgFkzeVJX/hqZAkEA33UiTURMdi5r
iEL8l741dQQ0mXO7Iek+U4B9rkZXxobxL6+G/Txsv+5/NI3ULjt/NGn6yIqCgwJM
37igriqejwJBAMm5V5ZRLSsN0upq0cO0rNQ79T+XwypUNALjFEL/NgsbplL1emjW
y7DJwjd9Wmu0MHlserDJ9NrFXHsYDJQjlbUCQBVyVpJ35abKGcQAOeIOfW73slyw
ANvmWPcGtAlP8wi41tkuzZPsgruBFnBi1GSDjVfofAtXT+NnCx3FyJYuvP0CQQC9
egARS1J33FY+pfM+NlkYSPFFuFEzU0A/bfg8LegfautBhR5jl05gUkLBSFdET04w
33om0KvTSgph/ObjxsD5AkAiA0i0DpwL477ffxs96K7uA9T6VEwrQGg1N5X6Elm9
mPrrOtvGP+Qbzl2ujsr8V6qPIbRabzR28MBFNK+O7iPd
-----END RSA PRIVATE KEY-----
```

注

11-00-10 以降，11-10-10 以降，11-20-10 以降では，「RSA PRIVATE KEY」部分が「PRIVATE KEY」となります。

(b) 楕円曲線暗号を利用する場合

楕円曲線暗号の秘密鍵を生成する場合のコマンドの使用例を次に示します。

使用例（Windows の場合）

```
# openssl.bat ecparam -genkey -noout -rand file -name P-256 -out httpsdkey-tmp.pem
#
```

使用例（UNIX の場合）

```
# openssl.sh ecparam -genkey -noout -rand file -name P-256 -out httpsdkey-tmp.pem
#
```

生成された秘密鍵を PKCS #8 形式に変換する場合のコマンドの使用例を次に示します。

使用例（Windows の場合）

```
# openssl.bat pkcs8 -topk8 -in httpsdkey-tmp.pem -out httpsdkey-ecc.pem -nocrypt
#
```

使用例（UNIX の場合）

```
# openssl.sh pkcs8 -topk8 -in httpsdkey-tmp.pem -out httpsdkey-ecc.pem -nocrypt
#
```

楕円曲線暗号の秘密鍵の内容

楕円曲線暗号の秘密鍵の内容を次に示します。

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQg5s6WeJmSoxeX+rw3
5cYub8aXBI4YdczVkpW10kbTtdShRANCAAQXh6w0loXxP2NZ2/wqvL5PZUEyJB1o
ZZc3zWVE9BTkx6sC46euFBrZ0ha5A+P9WwcdsC4IjaY09mf+rTeAmpgG
-----END PRIVATE KEY-----
```

(2) 証明書発行要求（CSR）の作成（openssl.bat req コマンドまたは openssl.sh req コマンド）

証明書発行要求（CSR）を作成する場合のコマンドの使用例を次に示します。ここで作成した CSR ファイルを CA に提出して，署名済みの証明書を発行してもらいます。

設定する項目および内容は，CSR を提出する CA の指示に従ってください。

使用例（Windows の場合）

```
# openssl.bat req -new -sha1 -key httpsdkey.pem -out httpsd.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Kanagawa
Locality Name (eg, city) []:Yokohama-shi
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HITACHI
Organizational Unit Name (eg, section) []:WebSite
Common Name (e.g. server FQDN or YOUR name) []:www.hws.hitachi.co.jp
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
#
```

使用例（UNIX の場合）

```
# openssl.sh req -new -sha1 -key httpsdkey.pem -out httpsd.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Kanagawa
Locality Name (eg, city) []:Yokohama-shi
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HITACHI
Organizational Unit Name (eg, section) []:WebSite
Common Name (e.g. server FQDN or YOUR name) []:www.hws.hitachi.co.jp
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
#
```

CSR の形式

CSR の形式を次に示します。

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBuzCCASQCAQAwezEeMBwGA1UEAxMVd3d3Lmh3cy5oaXRhY2hpLmNvLmpwMRAw
DgYDVQQLEwdXZWJTaXRlMRAwDgYDVQQKEwdISVRBQ0hJMRUwEwYDVQQHEwxZb2tv
aGFtYS1zaGkxETAPBgNVBAgTCEthbmFnYXdhMQswCQYDVQQGEwJKUDCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwgYkCgYEArZZyYumQcY8h4AppAz447H9R+Srzrt08eSzr
yZT8HYrDXz9I8XH6bMMahO4M6u2YI9iVzepQU1uI0f8bCwkFageBWwVQmDwcyJYf
1kY5X+2OgFEYV8CTu7I+A70VlYHobpM/FlBkzUVWD9/fTob0ALYNF9eTbFAL0c6U
sJBZfSsCAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GBAEiq+yGSVblaOuljyrAei9r3
n5mXtE5KXzQRz0cy6N5BaEV0l9KOtUaTCallsZmQdZ/6dZRSaE27xf/2UF3UxlCC
0+qrG10iQgDe5huSsqBnGGghJB2OPVUJh5S7YC6Ub6HRdOzq7H+D0qvsBC2C0dA/
cCkp8UsRzIjlDW8SVBZO
-----END CERTIFICATE REQUEST-----
```

6 ディレクティブ

この章では，httpsd.conf ファイルおよびアクセスコントロールファイルに定義するディレクティブについて説明します。

6.1　ディレクティブ一覧

6.1.1　ディレクティブ一覧

HTTP Server を起動するために，最低限設定が必要なディレクティブは次のとおりです。

- 最低限設定が必要なディレクティブ
  User（UNIX 版）
  Group（UNIX 版）
  ServerName
- SSL を利用する場合にさらに最低限必要なディレクティブ
  SSLEngine On
  SSLCertificateFile
  SSLCertificateKeyFile

コンフィグファイルに指定できるディレクティブの一覧を次の表に示します。なお，次の表およびディレクティブの説明では，次の記号を使用しています。

- U：UNIX 版だけに有効なディレクティブ
- W：Windows 版だけに有効なディレクティブ

表 6-1　ディレクティブ一覧

設定内容	ディレクティブ	複数指定
httpsd.conf ファイル内のブロックの定義	<Directory>	○
	<DirectoryMatch>	○
	<Files>	○
	<FilesMatch>	○
	<IfModule>	○
	<Limit>	○
	<Location>	○
	<LocationMatch>	○
	<VirtualHost>	○
サーバの基本的な定義	ServerName	×
	Port	×
	User U	×

設定内容	ディレクティブ	複数指定
	Group U	×
	ServerAdmin	×
	ServerRoot	×
	ServerSignature	×
	Listen	○
	BindAddress	×
	LoadModule	○
	LoadFile	○
	Include	○
	ExtendedStatus	×
	ServerTokens	×
	CoreDumpDirectory U	×
	FileETag	○
コンテンツを管理するための定義	UserDir	○
	DocumentRoot	×
	ErrorDocument	○
Web ブラウザからのリクエストについての定義（Alias）	Alias	○
	AliasMatch	○
	Redirect	○
	RedirectMatch	○
Web ブラウザへのレスポンスについての定義	HWSNotModifiedResponseHeaders	○
MIME タイプについての定義	TypesConfig	×
	AddCharset	○
	AddDefaultCharset	×
	AddType	○
	ForceType	×
	HWSErrorDocumentMETACharset W	×
コンテントネゴシエーションについての定義	LanguagePriority	○
	AddEncoding	○
	AddLanguage	○

設定内容	ディレクティブ	複数指定
	DefaultLanguage	×
	CacheNegotiatedDocs	×
	MultiviewsMatch	×
ハンドラについての定義	AddHandler	○
	SetHandler	×
Web サーバの性能についての定義	ServerLimit U	×
	StartServers U	×
	MinSpareServers U	×
	MaxSpareServers U	×
	MinSpareThreads U	×
	MaxSpareThreads U	×
	MaxClients U	×
	MaxRequestsPerChild U	×
	Timeout	×
	ListenBacklog	×
	ThreadsPerChild W	×
	ThreadsPerChild U	×
	HWSMaxQueueSize W	×
	HWSKeepStartServers U	×
	RequestReadTimeout	×
	SendBufferSize	×
	ThreadLimit W	×
	ThreadLimit U	×
KeepAlive の定義	KeepAlive	×
	MaxKeepAliveRequests	×
	KeepAliveTimeout	×
リクエストを制限する定義	LimitRequestBody	×

設定内容	ディレクティブ	複数指定
	LimitRequestFields	×
	LimitRequestFieldsize	×
	LimitRequestLine	×
CGI，環境変数の定義	ScriptAlias	○
	ScriptAliasMatch	○
	UseCanonicalName	×
	BrowserMatch	○
	BrowserMatchNoCase	○
	PassEnv	○
	SetEnv	○
	UnsetEnv	○
	SetEnvIf	○
	SetEnvIfNoCase	○
	Action	○
	Script	○
	ScriptInterpreterSource 	×
	HWSSetEnvIfIPv6	○
ディレクトリインデクスの表示内容の定義	DirectoryIndex	×
	FancyIndexing	×
	AddIconByEncoding	○
	AddIconByType	○
	AddIcon	○
	DefaultIcon	×
	ReadmeName	×
	HeaderName	×
	IndexIgnore	○
	IndexOrderDefault	×
	AddAltByEncoding	○
	AddAltByType	○
	AddAlt	○

設定内容	ディレクティブ	複数指定
	AddDescription	○
	IndexOptions	○
Web サーバへのアクセスを制御する定義	AccessFileName	×
	AllowEncodedSlashes	×
	AllowOverride	×
	AuthName	×
	AuthType	×
	AuthGroupFile	×
	AuthUserFile	×
	AuthAuthoritative	×
	Require	○
	Options	×
	Order	×
	Allow from	○
	Deny from	○
	Satisfy	×
	TraceEnable	×
	IdentityCheck U	×
SSL による暗号化および認証の定義	SSLRequireSSL	×
	SSLEngine	×
	SSLCertificateFile	×※2
	SSLCertificateKeyFile	×※2
	SSLCACertificatePath U	×
	SSLCACertificateFile	×
	SSLVerifyClient	×
	SSLVerifyDepth	×
	SSLCipherSuite	×※1
	SSLOptions	○
	SSLBanCipher	○
	SSLProtocol	×

設定内容	ディレクティブ	複数指定
	SSLCARevocationFile	×
	SSLCARevocationCheck	×
Web サーバを運用形態に合わせて複数ホストに見せる定義	ServerAlias	○
	ServerPath	×
イメージマップファイルについての定義	ImapDefault	×
	ImapBase	×
	ImapMenu	×
	HWSImapMenuCharset	×
採取するログの定義	HostnameLookups	×
	ErrorLog	×
	LogLevel	×
	LogFormat	○
	HWSWebSocketLog	×
	CustomLog	○
	TransferLog	○
	PidFile	×
	ScriptLog	×
	ScriptLogBuffer	×
	ScriptLogLength	×
	HWSLogSSLVerbose	×
	HWSLogTimeVerbose	×
	HWSRequestLog	×
	HWSRequestLogType	×
	HWSSuppressModuleTrace	○
	HWSErrorLogClientAddr	×
採取するトレースの定義	HWSTraceIdFile	×
	HWSTraceLogFile	×
	HWSPrfId	×
リバースプロキシについての定義	ProxyPass	○
	ProxyPassReverse	○
	ProxyVia	×

設定内容	ディレクティブ	複数指定
	ProxyErrorOverride	×※3
	ProxyPreserveHost	×
	HWSProxyPassReverseCookie	○
	Proxy100Continue	×
流量制限機能についての定義	QOSCookieDomain	×
	QOSCookieExpires	×
	QOSCookieName	○
	QOSCookieSecure	×
	QOSCookieServers	×
	QOSRedirect	○
	QOSRejectionServers	×
	QOSResponse	×
ヘッダカスタマイズ機能についての定義	Header	○
	RequestHeader	○
HTTP/2 プロトコル通信機能についての定義	H2Direct	×
	H2MaxWorkerIdleSeconds	×
	H2MaxWorkers	×
	H2MinWorkers	×
	H2Padding	×
	H2SerializeHeaders	×
	H2StreamMaxMemSize	×
	H2Upgrade	×
	H2WindowSize	×
	HWSH2SendGoaway	×
	Protocols	○
有効期限設定機能についての定義	ExpiresActive	×
	ExpiresByType	○
	ExpiresDefault	×
計画停止についての定義	HWSGracefulStopLog	×
	HWSGracefulStopTimeout	×

（凡例）

○：指定できる。

×：指定できない。

注

特に記述がないかぎり，ファイル名はディレクトリ名を含めた形式（パス情報付き）で記述できます。

注※1

TLSv1.3 用の SSLCipherSuite と TLSv1.2 以前用の SSLCipherSuite は，それぞれ一つずつ指定できます。

注※2

RSA 暗号用の SSLCertificateFile と楕円曲線暗号用の SSLCertificateFile は，それぞれ一つずつ指定できます。

RSA 暗号用の SSLCertificateKeyFile と楕円曲線暗号用の SSLCertificateKeyFile は，それぞれ一つずつ指定できます。

注※3

Off の指定を含まず，On だけを指定する場合は，複数指定できます。

6.1.2 ディレクティブの記述規則

(1) 正規表現

ディレクティブの指定に使用できる正規表現を次に示します。

表 6-2 正規表現

記号	機能	使用例	使用例の意味
.	任意の 1 文字。	a...c	a のあとに任意の 3 文字と c が続く。abcdc は適合する。
*	直前の 1 文字の 0 個以上の繰り返し。	ab*cd*	ac，abbbbc，abbbbcd は適合する。
+	直前の 1 文字の 1 個以上の繰り返し。	ab*c+	abbbc は適合する。abbb は適合しない。
?	直前の 1 文字があるかないか。	abbbc?	abbbc，abbb は適合する。
|	選択肢の区切り。	a|bc|d	a，bc または d。
¥	直後の特殊文字（. ^$*+?|¥[]()｛｝）の 1 文字。ただし，¥を表す場合は¥¥¥。	¥.	.と適合する。
		¥¥¥	1 文字の¥と適合する。
^	行の先頭に適合する。	^ab	abcde は適合する。
$	行の末尾に適合する。	abc$	aaabc は適合する。
{m}	直前の正規表現の m 個の繰り返し。	a{5}	aaaaa が適合する。
{m,}	直前の正規表現の m 個以上の繰り返し。	a{3,}	aaa，aaaa は適合する。aa は適合しない。
{m,n}	直前の正規表現の m 個以上 n 個以下の繰り返し。	a{3,5}	aaa，aaaa，aaaaa が適合する。aa，aaaaaa は適合しない。

記号	機能	使用例	使用例の意味
[文字列]	文字列にある任意の 1 文字※。	[abc]*または[a-c]*	aaa，bbb，ccc，cba，aab は適合する。
[^文字列]	文字列にない任意の 1 文字。	[^0-9]	数字以外の 1 字が適合する。
(文字列)	文字列をグループ化する。	(ab)+	ababab が適合する。ababb は適合しない。
		aa(xx|yy)bb	aaxxbb，aayybb が適合する。

注※

次の 3 文字は，[文字列]内で特殊な意味を持ちます。

- ^：[の次に指定して，文字列に含まれないものを示すために用います。
-]：文字列の最後を示すために用います。
- -：範囲を指定するために用います。

また，これら特殊文字の前の¥は省略されます。

[文字列]内で特殊な意味を持つ文字を通常の文字として指定するには，次のようにします。なお，^] - ¥ 以外の特殊文字は，通常の文字として扱われます。

- ^ ：文字列の先頭以外で指定します。（例）[ab^yz]
-]：文字列の先頭に指定します。（例）[]abxy]
- -：最後に指定します。（例）[abxy-]
- ¥：¥¥¥と指定します。（例）[¥¥¥abxy]

(2) ディレクティブに指定するパス情報

ディレクトリ名，ファイル名またはパス名を指定するディレクティブの場合，ディレクティブの種類によって，指定できるパス情報が異なります。

パスの種類には，次のものがあります。各ディレクティブのパス情報は，各ディレクティブで説明します。

- 絶対パスしか指定できない（Windows 版の場合，ドライブ名を付けた指定も絶対パスに含まれる）。
- ServerRoot ディレクティブの指定値からの相対パスで指定できる（ただし，ServerRoot ディレクティブの指定が先に必要）。

また，パス情報にネットワーク上のディレクトリやファイルを指定することはできません。ネットワークを使用したファイルシステム上のディレクトリやファイルを指定することもできません。

(3) コメント行

コンフィグファイル中，行の最初に#を付けると，コメント行になります。ただし，ディレクティブを指定したあとに#から始まる文字列を記述しても，#以降をコメントとして扱いません。コメント行を指定する場合の記述例を次に示します。

正しい例

```
#Deny from all
```

#行はコメント行として扱われます。

誤った例

```
Deny from all    #comment
```

#comment はディレクティブ指定値として扱われます。コメントとしては扱われません。

(4) IPv6 アドレスを指定するときの注意

ディレクティブに IPv6 アドレスを記述する場合は，「[IPv6 アドレス]」のように IPv6 アドレスを[ ]で囲んで指定してください。また，ディレクティブに IPv6 アドレスとポート番号を併記する場合は，「[IPv6 アドレス]:ポート番号」のように IPv6 アドレスを[ ]で囲み，「:」の後ろにポート番号を指定します。

ただし，次のディレクティブに IPv6 アドレスを記述する場合は，IPv6 アドレスを[ ]で囲まないで指定してください。

- Allow from ディレクティブ
- Deny from ディレクティブ
- HWSSetEnvIfIPv6 ディレクティブ

IPv6 アドレスを指定する場合は，グローバルユニキャストアドレスを指定してください。

(5) リダイレクタの定義について

リダイレクタ定義（JkMount キーなど）は，リダイレクタ定義ファイル（mod_jk.conf）に設定してください。

ワーカ定義（worker.ワーカ名.host など）は，ワーカ定義ファイル（workers.properties）に設定してください。

6.1.3 ディレクティブの説明形式

この項では，「6.2 ディレクティブの詳細」で記載する項目について，注意が必要な事項を説明します。

(1) ディレクティブ名

ディレクティブ名を記載しています。また，各ディレクティブの指定形式を次の記号で説明しています。

記号	意味
〔 〕	〔 〕内の項目を省略できます。

記号	意味
	（例）A〔,B〕〔,C〕のとき，次の４通りの指定ができます。 A A,B A,B,C A,C
{ }	{ } 内のどれか一つを選んで指定します。 （例）A {,B | ,C} のとき，次の２通りの指定ができます。 A,B A,C
|	選択肢の区切りを表しています。
__（アンダーライン）	項目の指定を省略したときに，システムが仮定する値を表しています。なお，この値は推奨値を示すものではありません。
…	この記号の直前に位置する項目を繰り返し指定できます。
～	この記号の直前に位置する項目をこの記号以降の文法規則に従って，記述することを示します。
《 》	項目の指定を省略したときに，システムが仮定する値を表しています。なお，この値は推奨値を示すものではありません。
(( ))	指定できる値の範囲を表しています。

(2) 「記述できる場所」について

ディレクティブによっては，記述できる場所が制限されているものがあります。「6.2 ディレクティブの詳細」では，各ディレクティブの記述できる場所を次のような形式で記述します。

記述できる場所	説明
httpsd.conf	VirtualHost ブロック，Directory ブロック以外の httpsd.conf ファイル
<VirtualHost>	httpsd.conf ファイルの VirtualHost ブロック
<Directory>	httpsd.conf ファイルの Directory ブロック，Location ブロック，Files ブロック
.htaccess	AccessFileName ディレクティブで指定したアクセスコントロールファイル
<Location>	httpsd.conf ファイルの Location ブロック

また，ディレクティブは次に示す順に参照されます。

1. VirtualHost ブロック，Directory ブロック以外の httpsd.conf ファイル
2. httpsd.conf ファイルの VirtualHost ブロック
3. httpsd.conf ファイルの Directory ブロック
4. アクセスコントロールファイル
5. httpsd.conf ファイルの Files ブロック

6. httpsd.conf ファイルの Location ブロック

Directory ブロックの AllowOverride ディレクティブの定義（上書き許可レベル）によって，アクセスコントロールファイルで定義しているディレクティブを有効または無効にできます。

(3) 「上書き許可」について

AllowOverride ディレクティブで上書きを許可する場合の許可レベルを定義します。各ディレクティブの上書き許可レベルは，各ディレクティブで説明します。許可レベルは複数あります。詳細は，AllowOverride ディレクティブを参照してください。なお，各ディレクティブの説明で.htaccess が指定でき，かつ上書き許可レベルの記述がない場合には，許可レベルは All になります。

6.2　ディレクティブの詳細

6.2.1　<で始まるディレクティブ

ブロック定義のディレクティブを参照順に示します。

1. <Directory>ディレクティブ，<DirectoryMatch>ディレクティブ，アクセスコントロールファイル
2. <Files>ディレクティブ，<FilesMatch>ディレクティブ
3. <Location>ディレクティブ

(1)　<Directory　ディレクトリ名>・・・</Directory>

(a)　内容

特定のディレクトリに対してディレクティブを定義する場合に指定します。**ディレクトリ名**にディレクトリ名を指定し，そのディレクトリとサブディレクトリだけに有効なディレクティブを定義するブロックを指定できます。

ディレクトリ名は，絶対パスで指定してください。

ただし，ディレクトリ名に，*および正規表現は指定できません。

また，J2EE サーバなどのアプリケーションが管理するディレクトリは，指定できません。

(b)　記述できる場所

httpsd.conf，<VirtualHost>

(c)　指定例

```
<Directory />                                                    …1.
     Options None                                                …2.
     AllowOverride None                                          …3.
</Directory>                                                     …4.

<Directory "<Application Serverのインストールディレクトリ>/httpsd/htdocs">  …5.
     Options Indexes                                             …6.
     AllowOverride None                                          …7.
     Order allow,deny                                            …8.
     Allow from all                                              …9.
</Directory>                                                     …10.
```

1. ルートディレクトリの定義
2. 機能はすべて無効
3. すべての上書き禁止

4. 定義終わり
5. <Application Server のインストールディレクトリ>/httpsd/htdocs ディレクトリの定義
6. ディレクトリインデクス表示可
7. すべての上書き禁止
8. Allow ディレクティブの指定を Deny ディレクティブの指定より先に評価
9. すべてのホストからのアクセスを許可
10. 定義終わり

(2) <DirectoryMatch 正規表現>・・・</DirectoryMatch>

(a) 内容

正規表現で記述した条件を満たすディレクトリに対してディレクティブを定義する場合に指定します。ディレクトリ名を正規表現で指定し，そのディレクトリとサブディレクトリだけに有効なディレクティブを定義するブロックを指定できます。

正規表現のディレクトリ名は，絶対パスで指定してください。

J2EE サーバなどのアプリケーションが管理するディレクトリは，指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(3) <Files ファイル名>・・・</Files>

(a) 内容

特定のファイルに対してディレクティブを定義する場合に指定します。**ファイル名**にファイル名を指定し，そのファイルだけに有効なディレクティブを定義するブロックを指定できます。

ファイル名に，*および正規表現は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(4) <FilesMatch 正規表現>・・・</FilesMatch>

(a) 内容

正規表現で記述した条件を満たすファイルに対してディレクティブを定義する場合に指定します。ファイル名を正規表現で指定し，そのファイルだけに有効なディレクティブを定義するブロックを指定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(5) <IfModule 〔!〕モジュール名>・・・</IfModule>

(a) 内容

指定したモジュールが組み込まれているとき，ブロック内で指定したディレクティブが有効になります。モジュール名の前に!を付けた場合は，指定したモジュールが組み込まれていないとき，ブロック内で指定したディレクティブが有効になります。ブロック内で指定可能なディレクティブに制限はありません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(6) <Limit メソッド名 〔メソッド名 …〕>・・・</Limit>

(a) 内容

特定の HTTP プロトコルメソッドだけに有効な，アクセス制御のディレクティブを定義する場合に指定します。メソッド名は複数指定できます。

指定できるメソッド名：GET，POST，PUT，DELETE，CONNECT，OPTIONS

（HEAD は GET に含まれています）

ブロック内に指定できるディレクティブ：

- Allow from
- Deny from
- AuthName
- AuthType
- AuthUserFile
- AuthGroupFile
- Order
- Require
- Satisfy

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 指定例

```
<Directory />
  <Limit PUT DELETE>                    …1.
    Order deny,allow                    …2.
    Deny from all                       …3.
    Allow from .your_domain.com         …4.
  </Limit>                              …5.
</Directory>
```

1. PUT および DELETE メソッドに対する定義
2. Deny ディレクティブの指定を，Allow ディレクティブの指定よりも先に評価
3. すべてのホストからの PUT および DELETE メソッドによるアクセスは不可
4. .your_domain.com からの PUT および DELETE メソッドによるアクセスを許可
5. 定義終わり

(7) <Location URL>・・・</Location>

(a) 内容

特定の URL で示す場所へのリクエストについて，ディレクティブを定義する場合に指定します。ただし，URL に，?以降（問い合わせ文字列），*および正規表現は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
  <Location /server-status>             …1.
       SetHandler server-status         …2.
       Order deny,allow                 …3.
       Deny from all                    …4.
       Allow from .your_domain.com      …5.
  </Location>                           …6.
```

1. URL /server-status の定義
2. このディレクトリのリクエストは server-status ハンドラに関連づける
3. Deny ディレクティブの指定を Allow ディレクティブの指定よりも先に評価
4. すべてのホストからのアクセスは不可
5. .your_domain.com からのアクセスを許可
6. 定義終わり

(8) <LocationMatch 正規表現>・・・</LocationMatch>

(a) 内容

正規表現で記述した条件を満たす URL へのリクエストについてディレクティブを定義する場合に指定します。ただし，URL に，?以降（問い合わせ文字列）は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(9) <VirtualHost {ホスト名｜IP アドレス〔:ポート番号〕} 〔{ホスト名｜IP アドレス〔:ポート番号〕} …〕>・・・</VirtualHost>

(a) 内容

ホスト名または IP アドレス〔:ポート番号〕で示すホストへのリクエストについてディレクティブを定義する場合に指定します。

なお，IPv6 アドレスに対応したホスト名も指定できます。IP アドレスに IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。

IP アドレスには，すべての IP アドレスにマッチする*が指定できます。

サーバ名に基づくバーチャルホストの場合には，IP アドレスを明示的に指定してください。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
<VirtualHost 172.17.40.30:80>
    :
</VirtualHost>
<VirtualHost [2001::123:4567:89ab:cdef]:80>
    :
</VirtualHost>
```

6.2.2 A で始まるディレクティブ

(1) AccessFileName ファイル名 〔ファイル名 …〕

～《.htaccess》

(a) 内容

アクセス制御のディレクティブを定義しているファイル（アクセスコントロールファイル）のファイル名を定義します。AllowOverride ディレクティブで許可されていれば，コンテンツリクエスト時に毎回このファイルを参照しアクセス制限がチェックされます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
AccessFileName .htaccess
```

アクセスコントロールファイルのファイル名は，.htaccess

(2) AllowEncodedSlashes {On | Off}

(a) 内容

符号化されたパス文字が存在する URL の使用を許可するかどうかを設定します。許可しない場合は，クライアントにステータスコード 404 Not Found を応答します。符号化されたパス文字は，Windows 版の場合は%2F と%5C，UNIX 版の場合は%2F を指します。%2F は/（スラッシュ），%5C は¥（バックスラッシュ）に該当します。

On：符号化されたパス文字を許可します。

Off：符号化されたパス文字を許可しません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
AllowEncodedSlashes On
```

(3) Action {MIME タイプ | ハンドラ} CGI スクリプト名

(a) 内容

MIME タイプまたはハンドラで指定したコンテンツが Web ブラウザからリクエストされたとき，実行させるスクリプトを CGI スクリプト名で指定します。CGI スクリプト名は，URL で指定します。このディレクティブを複数指定する場合，同じ MIME タイプに異なる CGI スクリプトは指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
Action image/gif /cgi-bin/images.cgi
```

(4) AddAlt "文字列" 拡張子 〔拡張子 …〕

(a) 内容

ディレクトリインデクス表示時に，拡張子に指定したファイルと関連づけて文字列を表示する場合に指定します。一つの文字列に対して複数の拡張子が指定できます。テキストモードの Web ブラウザのようにアイコン表示ができない環境で，ファイルの属性を表示する場合などに利用できます。

拡張子に指定できるものを次に示します。

- ファイル拡張子
- ワイルドカード表記のファイル拡張子またはファイル名
- ファイル名

このディレクティブを複数指定する場合，同じ拡張子に異なる文字列は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddAlt ”HTML” htm html
```

拡張子が htm または html のファイルの場合，文字列"HTML"を表示します。

(5) AddAltByEncoding "文字列" MIME エンコーディング 〔MIME エンコーディング …〕

(a) 内容

ディレクトリインデクス表示時に，アイコンが表示できない環境で，MIME エンコーディング（x-compress など）と関連づけて文字列を表示する場合に指定します。一つの文字列に対して複数の MIME エンコーディングが指定できます。このディレクティブを複数指定する場合，同じ MIME タイプに異なる文字列は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddAltByEncoding ”gzip” x-gzip
```

(6) AddAltByType "文字列" MIME タイプ 〔MIME タイプ …〕

(a) 内容

ディレクトリインデクス表示時に，アイコンが表示できない環境で，MIME タイプ（text/html など）と関連づけて文字列を表示する場合に指定します。一つの文字列に対して複数の MIME タイプが指定できます。このディレクティブを複数指定する場合，同じ MIME タイプに異なる文字列は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddAltByType ”plain text” text/plain
```

(7) AddCharset 文字セット 拡張子 〔拡張子 …〕

(a) 内容

ファイル拡張子に対する文字セットを指定します。文字セットは Content-Type ヘッダに charset=の値として設定されます。クライアントに対して文字セットを明示する場合に使用します。このディレクティブを複数指定する場合，同じ拡張子に異なる文字列は指定できません。指定する拡張子は，AddType ディレクティブまたは TypesConfig ディレクティブで指定したファイルで，MIME タイプの関連づけが必要です。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

(8) AddDefaultCharset 〔On | Off | 文字セット〕

(a) 内容

ファイル拡張子に対する文字セットのデフォルト値を指定します。AddCharset ディレクティブの設定に対するデフォルト値となります。Content-Type が text/plain，text/html の場合のデフォルトとして設定されます。

On：デフォルト文字セットとして ISO-8859-1 を設定します。

Off：文字セットを設定しません。

文字セット：指定した文字セットをデフォルト文字セットとします。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddDefaultCharset ISO-2022-JP
```

(9) AddDescription "文字列" ファイル名 〔ファイル名 …〕

(a) 内容

ディレクトリインデクス整形表示時に，ファイル名で指定したファイル拡張子，ワイルドカード表記ファイル名またはパス情報なしの完全なファイル名に対して，説明文として文字列を表示する場合に指定します。なお，ファイル名にスラッシュで終わる文字列を指定した場合，*が付けられワイルドカード指定と同様に見なされます。

ファイル名に指定できるものを次に示します。

- ファイル拡張子
- ワイルドカード表記のファイル名
- ファイル名

このディレクティブを複数指定する場合，同じファイル名に異なる文字列は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddDescription ”The planet Mars” /web/pics/mars.gif
```

(10) AddEncoding 圧縮形式 拡張子

(a) 内容

Web サーバ内の圧縮データを Web ブラウザに表示させるときに必要な拡張子と圧縮形式の関連づけを指定します。Web ブラウザに圧縮ファイルの展開の情報として Content-Encoding ヘッダを Web サーバから送信する場合に設定します。このヘッダを利用した運用は，Web ブラウザ側の実装に依存します。このディレクティブを複数指定する場合，同じ拡張子に異なる圧縮形式は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddEncoding x-compress Z     拡張子がZのファイルの圧縮形式はx-compress
AddEncoding x-gzip gz        拡張子がgzのファイルの圧縮形式はx-gzip
```

(11) AddHandler ハンドラ名 拡張子 〔拡張子 …〕

(a) 内容

ハンドラで処理するファイル拡張子を対応付ける場合に定義します。

指定できるハンドラ名を次に示します。このディレクティブを複数指定する場合，同じ拡張子に異なるハンドラ名は指定できません。

cgi-script：CGI スクリプトの実行

imap-file：イメージマップ処理

server-status：ステータスの表示

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddHandler cgi-script .cgi              拡張子.cgiはcgi-scriptハンドラ
AddHandler imap-file map                拡張子mapはimap-fileハンドラ
```

(12) AddIcon {(文字列，URL)｜URL} 拡張子 〔拡張子 …〕

(a) 内容

拡張子などにディレクトリインデクスのアイコンを対応付けて表示する場合に指定します。文字列には画像表示ができない Web ブラウザの場合に表示する文字を指定します。URL にアイコンの画像ファイルの URL を指定します。自ホスト内の画像ファイルを指定する場合，URL の「http：//IP アドレス」は省略できます。なお，URL の「http：//IP アドレス」を省略しないで IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。

拡張子として指定できるものを次に示します。

- ファイル拡張子
- ワイルドカード表記のファイル拡張子またはファイル名
- ファイル名

拡張子として^^DIRECTORY^^を記述すると，ディレクトリに対するアイコンを設定できます。また，^^BLANKICON^^と指定すると，ディレクトリインデクスを表示した場合の，表示内容のヘッダのインデントを合わせるためのアイコンを設定できます。

このディレクティブを複数指定する場合，同じ拡張子に異なる文字列や URL は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddIcon /icons/tar.gif .tar
```

拡張子が.tar の場合のアイコン定義

```
AddIcon /icons/layout.gif .html .shtml .htm .pdf
```

拡張子が.html，.shtml，.htm，.pdf の場合のアイコン定義

```
AddIcon /icons/text.gif .txt
```

拡張子が.txt の場合のアイコン定義

```
AddIcon /icons/back.gif ..
```

親ディレクトリのアイコン定義

```
AddIcon /icons/hand.right.gif README
```

README ファイルのアイコン定義

```
AddIcon /icons/folder.gif ^^DIRECTORY^^
```

ディレクトリの場合のアイコン定義

```
AddIcon /icons/blank.gif ^^BLANKICON^^
```

ディレクトリインデクスのヘッダのインデントアイコン定義

```
AddIcon http://[2001::123:4567:89ab:cdef]/icons/text.gif .txt
```

IPv6 アドレスを指定する場合のアイコン定義

(13) AddIconByEncoding {(文字列，URL)｜URL} MIME エンコーディング 〔MIME エンコーディング …〕

(a) 内容

ディレクトリインデクスの整形表示時のアイコンを MIME エンコーディングと対応付けて表示する場合に指定します。文字列には画像表示ができない Web ブラウザの場合に表示する文字を指定します。URL にアイコンの画像ファイルの URL を指定します。自ホスト内の画像ファイルを指定する場合，URL の「http：//IP アドレス」は省略できます。なお，URL の「http：//IP アドレス」を省略しないで IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。

このディレクティブを複数指定する場合，同じ MIME タイプに異なる文字列や URL は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddIconByEncoding (CMP, /icons/compressed.gif) x-compress x-gzip
```

MIME エンコーディング x-compress および x-gzip の場合のアイコン定義

(14) AddIconByType {(文字列，URL)｜URL} MIME タイプ 〔MIME タイプ …〕

(a) 内容

ディレクトリインデクスの整形表示時のアイコンを MIME タイプと対応付けて表示する場合に指定します。画像表示ができない Web ブラウザの場合に表示する文字は文字列で指定できます。また，URL で表示するアイコンの画像ファイル名の場所を指定できます。なお，URL の「http：//IP アドレス」を省略しないで IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。

このディレクティブを複数指定する場合，同じ MIME タイプに異なるファイル名は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
AddIconByType (TXT, /icons/text.gif) text/*
```

MIME タイプ　text/*の場合のアイコン定義

```
AddIconByType (IMG, /icons/image2.gif) image/*
```

MIME タイプ　image/*の場合のアイコン定義

```
AddIconByType (SND, /icons/sound2.gif) audio/*
```

MIME タイプ　audio/*の場合のアイコン定義

```
AddIconByType (VID, /icons/movie.gif) video/*
```

MIME タイプ　video/*の場合のアイコン定義

(15) AddLanguage　言語コード　拡張子

(a) 内容

ドキュメントで使用する言語を指定します。言語コードは Content-Language レスポンスヘッダに設定されます。このディレクティブを指定すると，Web ブラウザの言語設定で言語コードの優先順位（Accept-Language ヘッダ）がリクエストに設定されている場合，Web サーバから送信するコンテンツを選択するコンテントネゴシエーションができます。言語コードは Web ブラウザが送信するヘッダ情報に依存します。基本的には，ISO639 に定義されている言語コードに従って指定します。なお，コンテントネゴシエーションを有効にするためには，Options ディレクティブで MultiViews オプションを設定しなければなりません。このディレクティブを複数指定する場合，同じ拡張子に異なる言語コードは指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddLanguage ja .ja                日本語
AddLanguage en .en                英語
AddLanguage fr .fr                フランス語
AddLanguage de .de                ドイツ語
AddLanguage da .da                デンマーク語
AddLanguage el .el                ギリシャ語
AddLanguage it .it                イタリア語
```

(16) AddType MIMEタイプ 拡張子 〔拡張子 …〕

(a) 内容

TypesConfig ディレクティブで指定したファイルに未定義のコンテンツの拡張子と MIME タイプを関連づけたい場合に指定します。このディレクティブを複数指定する場合，同じ拡張子に異なる MIME タイプは指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
AddType text/html .shtml
```

MIME タイプ text/html と拡張子.shtml を関連づけます。

(17) Alias URL ディレクトリ名

(a) 内容

Web ブラウザからリクエストされた特定の URL を別名に置き換える場合に指定します。ただし，URL には，?以降（問い合わせ文字列）を指定できません。URL で指定されたディレクトリを，ディレクトリ名で指定したディレクトリに置き換えて，Web ブラウザに表示します。

次のディレクティブ指定値と重複する URL は指定できません。

- AliasMatch の正規表現

- ProxyPass のパス名
- Redirect の旧パス
- RedirectMatch の正規表現
- ScriptAlias の URL
- ScriptAliasMatch の正規表現
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
Alias /aaa/bbb/ C:/alias/
ProxyPass /aaa/ http://aaa.example.com/
```

ディレクトリ名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
Alias /icons/ "<Application Serverのインストールディレクトリ>/httpsd/icons/"
```

/icons/を<Application Server のインストールディレクトリ>/httpsd/icons/に置き換えます。

(18) AliasMatch 正規表現 新パス

(a) 内容

Web ブラウザからリクエストされた URL を別名に置き換える場合に指定します。ただし，URL には，？以降（問い合わせ文字列）を指定できません。

正規表現で記述した条件を満たす URL が Web ブラウザからリクエストされた場合，指定した新パスのコンテンツを Web ブラウザに表示します。正規表現で括弧（）を使用してグループ化している場合，その i 番目のグループの表現にマッチした文字列を，新パスで $i を使用して参照できます。i には 1 から 9 までの数字を指定します。

次のディレクティブ指定値と重複する正規表現は指定できません。

- Alias の正規表現
- ProxyPass のパス名
- Redirect の旧パス
- RedirectMatch の正規表現
- ScriptAlias の URL

- ScriptAliasMatch の正規表現
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
AliasMatch ^/aaa/bbb/(.*) C:/alias/$1
ProxyPass /aaa/ http://aaa.example.com/
```

新パスは，絶対パスで指定してください。また，新パスの文字として，'$'または'&'を含める場合は，その文字の前に'¥'を付加してください。なお，$i を指定する際には，'$'の前に'¥'を付加する必要はありません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
AliasMatch ^/html/(.*) "C:/htdocs/html/$1"
```

"/html/"で始まるリクエストのとき，/html/部分を C:/htdocs/html/に置き換えます。例えば，/html/index.html へのアクセスの場合，C:/htdocs/html/index.html に置き換えます。

(19) Allow from {ホスト | all | env=環境変数} 〔{ホスト | env=環境変数} …〕

(a) 内容

Web サーバへアクセスできるクライアントを制限する場合に指定します。ホストにはアクセスを許可するホストのドメイン名，IP アドレス，サブネット，ネットマスクを指定できます。すべてのホストからのアクセスを許可する場合は all を指定します。

また，ホストには，IPv6 アドレスに関するドメイン名，アドレスおよびプリフィックス長も指定できます。IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲まないでください。プリフィックス長は，「IPv6 アドレス/プリフィックス長」の形式で指定します。プリフィックス長は 10 進数で指定してください。

env=環境変数を指定すると，サーバへのアクセスを環境変数で制御できます。BrowserMatch，BrowserMatchNoCase，SetEnvIf，SetEnvIfNoCase ディレクティブと併せて使用すれば，HTTP リクエストヘッダフィールドに基づいてアクセスを制限できます。

Allow ディレクティブ（アクセス許可）と Deny ディレクティブ（アクセス制限）は，Order ディレクティブで評価の順序を設定できます。

ホスト	意味
ドメイン名	ドメイン名で指定したホストからのアクセスを許可する。

ホスト	意味
IP アドレス	IP アドレスで指定したホストからのアクセスを許可する。
サブネット	サブネット（IP アドレスの最初の 1 から 3 バイト）で指定したホストからのアクセスを許可する。
ネットマスク	ネットマスク表記（例:10.1.0.0/255.255.0.0）で指定したホストからのアクセスを許可する。 10.1.0.0/16 形式で表記した場合，10.1.0.0/255.255.0.0 と同じ意味である。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

Limit レベル

(d) 指定例

(例 1)

```
SetEnvIf User-Agent Mozilla.* access_ok
<Directory /docroot>
    Order deny,allow
    Deny from all
    Allow from env=access_ok
</Directory>
```

この場合，User-Agent の文字列が Mozilla を含むブラウザからのリクエストだけがアクセスを許可されて，ほかのアクセスは拒否されます。

(例 2)

ホストに IPv6 アドレスを指定する場合は，次のように指定します。

```
allow from 2001::123:4567:89ab:cdef
```

また，プリフィックス長を指定するとき，次の指定はどれも同じ意味となります。

```
allow from 2001:0:0:89ab::/64
allow from 2001:0:0:89AB::/64
allow from 2001::89ab:0:0:0:0/64
allow from 2001:0000:0000:89ab:0000:0000:0000:0000/64
```

(20) AllowOverride 指示子 〔指示子 …〕

～《None》

(a) 内容

AccessFileName ディレクティブで指定したファイルでアクセス情報定義の上書きを許可するかどうかを設定します。各指示子によって制御できるディレクティブは，各ディレクティブの上書き許可の記述を参照してください。

指示子	内容
AuthConfig	AuthGroupFile，AuthName，AuthType，AuthUserFile，Require ディレクティブなど サーバへのアクセス制御関連のディレクティブの上書きを許可
FileInfo	AddType，AddEncoding，AddLanguage ディレクティブなど コンテンツ管理，MIME タイプのディレクティブの上書きを許可
Indexes	FancyIndexing，AddIcon，AddDescription ディレクティブなど ディレクトリインデクス関連のディレクティブの上書きを許可
Limit	Allow from，Deny from，Order ディレクティブ ホスト名または IP アドレスを用いたアクセス制御の上書きを許可
Options	Options ディレクトリの使用を許可
All	すべての上書きを許可する
None	すべての上書きを禁止する

(b) 記述できる場所

<Directory>

Location ブロック，Files ブロックに指定した場合，AllowOverride ディレクティブの指定は有効になりません。

(21) AuthAuthoritative {On | Off}

(a) 内容

ユーザ認証をする場合の制御方法を指定します。

On：AuthUserFile，AuthGroupFile，Require ディレクティブの設定によるユーザ認証をします。ユーザ未登録またはパスワード不整合の場合は 401 エラーステータスを Web ブラウザに表示します。

Off：AuthUserFile，AuthGroupFile，Require ディレクティブの設定によるユーザ認証をします。そのとき，パスワード不整合の場合は 401 エラーステータスを Web ブラウザに表示します。さらに，ユーザ未登録の場合には他製品のモジュール（機能）でユーザ認証をします。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(22) AuthGroupFile　ファイル名

(a) 内容

グループでユーザ認証をする場合，認証するグループのリストを格納しているファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

グループファイルはテキストエディタを使用して次に示すようなフォーマットで作成してください。

グループ名：ユーザ名〔 ユーザ名　…〕

任意のグループ名に，ユーザ認証のためのパスワードファイルに登録しているユーザ名を定義します。1 行につき 1 グループで指定します。グループファイルには複数グループを定義できます。同じグループ名の行を複数指定した場合には，同じグループ名に登録されているすべてのユーザ名を含んだ一つのグループが定義されます。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(23) AuthName　realm 名

(a) 内容

ユーザ認証する場合の realm 名（Web ブラウザのユーザ認証画面に表示される）を指定します。このディレクティブを指定する場合は AuthType，Require，AuthUserFile（または AuthGroupFile）ディレクティブを必ず指定しなければなりません。ただし，ディレクトリサービスを利用したユーザ認証を行う場合は，AuthUserFile（または AuthGroupFile）ディレクティブの指定は必要ありません。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(24) AuthType　認証タイプ名

(a) 内容

ユーザ認証する場合の認証制御のタイプを指定します。認証タイプ名として"Basic"が指定できます。このディレクティブを指定する場合は AuthName，Require，AuthUserFile（または AuthGroupFile）ディ

レクティブを必ず指定しなければなりません。ただし，ディレクトリサービスを利用したユーザ認証を行う場合は，AuthUserFile（または AuthGroupFile）ディレクティブの指定は必要ありません。

Basic：Base64 コード変換をします。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(25) AuthUserFile ファイル名

(a) 内容

ユーザ名でユーザ認証をする場合，認証するユーザ名とパスワードのリストを格納しているファイル名を指定します。

ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

6.2.3 B，C，D で始まるディレクティブ

(1) BindAddress {IP アドレス|*}

～《*》

(a) 内容

Web サーバをインストールしたサーバ機に割り当てられた IP アドレスのうち，どの IP アドレスから Web サーバに接続できるようにするかを指定します。IP アドレスに IPv6 アドレスは指定できません。どの IPv4 アドレスからも接続できるようにする場合には，*を指定します。Listen ディレクティブを指定した場合は，BindAddress ディレクティブの指定は無視されます。

(b) 記述できる場所

httpsd.conf

(2) BrowserMatch "ブラウザ名" 環境変数〔=値〕〔環境変数〔=値〕 …〕

(a) 内容

Web ブラウザごとに環境変数を設定する場合に指定します。設定する値のデフォルト値は 1 です。環境変数の前に!が付いたときは，その環境変数の設定を解除します。ブラウザ名は正規表現で指定でき，大文字，小文字を区別します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 指定例

```
BrowserMatch ”Mozilla/2” nokeepalive
BrowserMatch ”MSIE 4¥.0b2;” nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch ”RealPlayer 4¥.0” force-response-1.0
BrowserMatch ”Java/1¥.0” force-response-1.0
BrowserMatch ”JDK/1¥.0” force-response-1.0
BrowserMatch ”Microsoft Data Access Internet Publishing Provider” redirect-carefully
BrowserMatch ”^WebDrive” redirect-carefully
BrowserMatch ”^WebDAVFS/1.[012]” redirect-carefully
BrowserMatch ”^gnome-vfs” redirect-carefully
```

指定例で示した環境変数の意味を次に示します。

環境変数	内容
nokeepalive	KeepAlive 接続を無効にします。Via ヘッダがリクエストに付加されている場合または HTTP/2 通信の場合は，KeepAlive 接続を無効にできません。
downgrade-1.0	HTTP/1.1 以上のリクエストを，HTTP/1.0 のリクエストとして扱います。
force-response-1.0	HTTP/1.0 のリクエストに対して，常に HTTP/1.0 のレスポンスを応答します。
redirect-carefully	ディレクトリへのアクセスで URL の最後に'/'を付加していなく，かつそれが GET メソッド以外を使用していたとき，クライアントにリダイレクトを要求しません。

(3) BrowserMatchNoCase "ブラウザ名" 環境変数〔=値〕〔環境変数〔=値〕 …〕

(a) 内容

Web ブラウザごとに環境変数を設定する場合に指定します。設定する値のデフォルト値は 1 です。環境変数の前に!が付いたときは，その環境変数の設定を解除します。ブラウザ名は正規表現で指定でき，大文字，小文字を区別しません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(4) CacheNegotiatedDocs 〔{On | Off}〕

(a) 内容

コンテントネゴシエーションをするリクエストで，クライアント側のキャッシュを有効にするかどうかを指定します。ディレクティブの引数を省略した場合は，On を指定した場合と同様の動作をします。ディレクティブを設定しない場合は，Off を指定した場合と同様の動作をします。このディレクティブの指定は，HTTP/1.1 のリクエストに対しては無効です。

On：キャッシュされるようになります。

Off：Expires ヘッダが付けられてキャッシュされなくなります。

(b) 記述できる場所

httpsd.conf

(5) CoreDumpDirectory ディレクトリ名 U

～《ServerRoot ディレクティブ指定値》

(a) 内容

コアをダンプするディレクトリを指定します。絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。なお，指定したディレクトリには，User，Group ディレクティブで指定したユーザ，グループからの書き込み権限を付与する必要があります。Linux 版では，ディレクティブをコンフィグファイルに指定した場合だけ有効となります。

(b) 記述できる場所

httpsd.conf

(6) CustomLog {ファイル名|パイプ} {"フォーマット"|ラベル名} 〔env=〔!〕環境変数〕

(a) 内容

任意のフォーマットのログをファイルに出力させる場合に指定します。フォーマットは LogFormat ディレクティブで指定するフォーマットと同様です。

このディレクティブを複数指定する場合，同じファイル名は複数指定できません。

ファイル名：ログの出力先ファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

パイプ：標準入力からログ情報を受け取るプログラムを"|プログラム名"の形式で指定します。Web サーバはログ情報に含める改行コードを CRLF にして渡します。

（Windows 版での注意事項）

パイプで指定されたプログラムは，制御プロセスと Web サーバプロセス用にログ情報を受け取るそれぞれ別のプロセスとして生成されます。これをパイププロセスと呼びます。サービスとして Web サーバを起動する場合には次の点に注意してください。

- 制御プロセスのログ情報取得不可

  サービスとして Web サーバを起動した場合には，制御プロセスからのログ情報を受け取るための標準入力は NUL デバイスに関連づけられているため，制御プロセス用のパイププロセスは，制御プロセスからのログ情報を受け取ることはできません。制御プロセスからのログ情報とは，Web サーバ起動，停止時のエラーログ情報であり，これらの情報は採取できないことになります。Web サーバ起動後のエラーログ，アクセスログの情報は Web サーバプロセスからのログ情報となりますので，Web サーバプロセス用のパイププロセスで受け取れます。

- プログラム作成時の留意点

  制御プロセス用のパイププロセスは，NUL デバイスからのデータ読み込み処理で，read()のバッファが小さいと入力データ待ち状態が解除されないことがあります。read()のバッファを十分大きい値を取るなどしてパイププロセスが入力データ待ち状態にならないようにしてください。

- プログラムに引数を指定する場合の注意

  プログラム，引数に空白を含む場合には，¥"で囲んでください。

  プログラム，引数を¥"で囲む場合には，全体も¥"で囲んでください。

  （例）

  ```
  CustomLog ”|¥”¥”<Application Server のインストールディレクトリ>/httpsd/sbin/
  rotatelogs.exe¥” ¥”<Application Server のインストールディレクトリ>/httpsd/logs/access¥”
  86400 -diff 540¥”” common
  ```

- プログラムの引数指定に誤りがある場合の注意

  プログラムの引数指定に誤りがある場合，プログラムの起動に失敗しますが，Web サーバは起動します。この場合はログが出力されません。プログラムを指定した場合は，ログファイルが作成され，意図した分割が実施されることを確認してください。

"フォーマット"：ログフォーマットを指定します。指定できるフォーマット名を表 6-3，表 6-4 に示します。

ラベル名：LogFormat ディレクティブで定義したラベル名を指定します。

env=環境変数：指定した環境変数が設定されている場合に，ログを採取します。

env=!環境変数：指定した環境変数が設定されていない場合に，ログを採取します。

表 6-3　フォーマット一覧

フォーマット	意味
%A※1	Web サーバの IP アドレス。
%a※1	クライアントの IP アドレス。
%B	送信バイト数（HTTP ヘッダおよび chunked エンコーディングによって追加されたデータを除く）。
%b	送信バイト数（HTTP ヘッダおよび chunked エンコーディングによって追加されたデータを除く）。ただし，0 の場合は-（ハイフン）。
%{cookie_name}C	Cookie ヘッダ値に含まれるクッキー名 cookie_name の値。
%D	リクエスト処理時間をマイクロ秒で表示。
%{env_name}e	env_name に指定した環境変数の値。
%f	クライアントが要求したディレクトリまたはファイル名。
%H	リクエストプロトコル(HTTP/1.0 など)。
%h※2	クライアントのホスト名。
%I	リクエストとヘッダを含む，全受信バイト数。
%{header_name}i	header_name に指定したリクエストヘッダの値。
%l	クライアントの識別情報（IdentityCheck ディレクティブが On，かつクライアント上で identd が動作している場合）。
%m	リクエストメソッド(GET，POST など)。
%{note_name}n	note_name に指定した Web サーバ内モジュールの注記の値。 hws_ap_root：ルートアプリケーション情報※3 hws_ap_client：クライアントアプリケーション情報※3 HWS_H2_WORKER_THREAD_ID：HTTP/2 リクエストを処理しているワーカスレッドのスレッド ID
%O	ヘッダを含む，全送信バイト数。
%{header_name}o	header_name に指定したレスポンスヘッダの値。
%P	HTTP 通信のリクエストを処理するプロセス ID。
%{hws_thread_id}P	HTTP 通信のリクエストを処理するスレッド ID。Windows 版で有効。
%{tid}P	HTTP 通信のリクエストを処理するスレッド ID。UNIX 版で有効。
%p	ポート番号。
%q	問い合わせ文字列。
%r	HTTP 通信のリクエストの先頭行。
%s	ステータス（内部リダイレクトされた場合はオリジナルを示す）。

フォーマット	意味
%T	リクエスト処理に掛かった時間（秒）。HWSLogTimeVerbose ディレクティブで On を指定すると，ミリ秒単位まで表示。
%t	リクエスト処理を開始した時刻。HWSLogTimeVerbose ディレクティブで On を指定すると，ミリ秒単位まで表示。
%{format}t	リクエスト処理を開始した時刻。strftime()で定義されているフォーマットを format に指定する。
%U	URL。
%u	クライアントのユーザ名（ユーザ認証をした場合）。
%V※2	UseCanonicalName ディレクティブの指定に従い，ServerName ディレクティブ指定値，サーバ名または IP アドレス。
%v	サーバ名。
%X	レスポンス完了時の接続ステータス。 ＋：レスポンス送信後も接続を維持する。 －：レスポンス送信後に接続を切断する。 X：レスポンス完了前に接続を切断する。
%>s	最終ステータス。

注

フォーマットで示す{　}は選択を意味するものではありません。{　}内の太字はログを採取する変数名を，細字は文字列そのままを記述します。

注※1

フォーマットに%A または%a を指定した場合，IPv6 アドレスも出力できます。

注※2

フォーマットに%h または%V を指定した場合，IPv6 アドレスに対応したホスト名または IPv6 アドレスも出力できます。

注※3

ルートアプリケーション情報とクライアントアプリケーション情報は，クライアントから AP 情報が設定されたリクエストを受信した場合，およびリバースプロキシを使用してリクエストを転送する場合に出力します（リダイレクタを使用した場合は出力しません）。

表 6-4　SSL 関連のログフォーマット一覧

フォーマット	意味
%{version}c	SSL のバージョン
%{cipher}c	現在の通信で使用している暗号種別
%{clientcert}c	SSL クライアント証明書の subject の Distinguished Name

フォーマットの%の後ろにステータスコードを記述できます。

(例) エラーステータスコード 400 および 501 の場合，User-Agent リクエストヘッダ値のログを採取する。

```
%400,501{User-Agent}i
```

（例）エラーステータスコード 200，304 および 302 の 3 種類以外の場合，Referer リクエストヘッダ値のログを採取する。

```
%!200,304,302{Referer}i
```

また，env=は，指定した環境変数の設定によって，ログの採取を分ける場合に指定します。

（例）gif へのアクセスは gif.log に，gif 以外へのアクセスは nongif.log にログを採取する。

```
SetEnvIf Request_URI ¥.gif$ gif-image
CustomLog logs/gif.log common env=gif-image
CustomLog logs/nongif.log common env=!gif-image
```

(b) 注意事項

11-20 以降では，CustomLog ディレクティブで指定するログのフォーマットで%{header_name}i を指定した場合，リバースプロキシで付加するリクエストヘッダはログに出力されません。詳細は「付録 C 旧バージョンからの移行に関する注意点」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

```
CustomLog logs/access.log common
CustomLog logs/ssl.log "%t %{version}c %{cipher}c %{clientcert}c"
```

(7) DefaultIcon URL

(a) 内容

ディレクトリインデクスで表示するアイコンを指定します。AddIcon，AddIconByType および AddIconByEncoding ディレクティブのどれにも該当しない場合に表示するアイコンの URL を指定します。なお，URL の「http：//IP アドレス」を省略しないで IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
DefaultIcon /icons/unknown.gif
```

(8) DefaultLanguage 言語コード

(a) 内容

ドキュメントで使用するデフォルトの言語を指定します。指定した言語コードは Content-Language レスポンスヘッダに設定されます。AddLanguage ディレクティブの設定に対するデフォルト値となります。デフォルト値が設定されていない場合，Content-Language レスポンスヘッダは送信しません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(9) Deny from {ホスト | all | env=環境変数} 〔{ホスト | env=環境変数} …〕

(a) 内容

Web サーバへアクセスできるクライアントを制限する場合に指定します。ホストにはアクセスを禁止するホストのドメイン名，IP アドレス，サブネット，ネットマスクを指定できます。すべてのホストからアクセスを禁止する場合は，all を指定します。

また，ホストには，IPv6 アドレスに関するドメイン名，アドレスおよびプリフィックス長も指定できます。IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲まないでください。プリフィックス長は，「IPv6 アドレス/プリフィックス長」の形式で指定します。プリフィックス長は 10 進数で指定してください。

env=環境変数を指定すると，サーバへのアクセスを環境変数で制御できます。BrowserMatch，BrowserMatchNoCase，SetEnvIf，SetEnvIfNoCase ディレクティブと併せて使用すれば，HTTP リクエストヘッダフィールドに基づいてアクセスを制限できます。

Allow ディレクティブ（アクセス許可）と Deny ディレクティブ（アクセス制限）は，Order ディレクティブで評価の順序を設定できます。

アクセスを拒否した場合，ステータスコード 403 を返します。

ホスト	意味
ドメイン名	ドメイン名で示すホストからのアクセスを禁止する。
IP アドレス	IP アドレスで示すホストからのアクセスを禁止する。
サブネット	サブネット（IP アドレスの最初の 1 から 3 バイト）で指定したホストからのアクセスを禁止する。
ネットマスク	ネットマスク表記（例：10.1.0.0/255.255.0.0）で指定したホストからのアクセスを禁止する。

ホスト	意味
	10.1.0.0/16 形式で表記した場合 10.1.0.0/255.255.0.0 と同じ意味である。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

Limit レベル

(10) DirectoryIndex ファイル名 〔ファイル名 …〕

～《index.html》

(a) 内容

Web ブラウザからのリクエストが特定のコンテンツを指定していない場合に，デフォルトとしてクライアントに送信するコンテンツのファイル名を指定します。ファイル名を複数指定した場合は，先に指定したファイル名を優先して送信します。

ここで指定したファイルがリクエストされたディレクトリにない場合，Options ディレクティブの指定によって Web ブラウザの表示が変わります。

- Indexes が有効の場合
  Web ブラウザに Web サーバで作成したディレクトリのインデクスを表示します。
- Indexes が無効の場合
  ステータスコード 403 Forbidden を応答します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
DirectoryIndex index.html
```

ファイル名の指定がないリクエストの場合，ディレクトリに index.html があれば表示させます。

(11) DocumentRoot ディレクトリ名

～《/opt/hitachi/httpsd/htdocs》 (UNIX 版)

～《ServerRoot ディレクティブのデフォルト値/htdocs》 (Windows 版)

(a) 内容

コンテンツを格納するドキュメントルートディレクトリを絶対パスで指定します。ディレクトリ名の終端には/（スラッシュ）を記述しないでください。

ディレクトリ名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
DocumentRoot ”<Application Serverのインストールディレクトリ>/httpsd/htdocs”
```

6.2.4 E，F，G，H，I で始まるディレクティブ

(1) ErrorDocument エラーステータス番号 {テキスト | ローカル URL | フル URL}

(a) 内容

エラーが発生したときに，Web ブラウザへ表示するメッセージをカスタマイズする場合に指定します。

テキスト：文字列を"で囲み指定します。

ローカル URL：先頭に/を記述して，自サイト内のコンテンツを指定します。

フル URL：http://または https://で始まる URL を記述し，他サイトのコンテンツを指定します。

このディレクティブに指定できるエラーステータス番号と，テキスト，ローカル URL，フル URL の指定可否について，次に示します。

エラーステータス番号（意味）	テキスト	ローカル URL	フル URL
400 (Bad Request)	○	△	△
401 (Authorization Required)	○	○	×
402 (Payment Required)	○	○	○
403 (Forbidden)	○	○	○
404 (Not Found)	○	○	○
405 (Method Not Allowed)	○	○	○

エラーステータス番号（意味）	テキスト	ローカル URL	フル URL
406 (Not Acceptable)	○	○	○
407 (Proxy Authentication Required)	○	○	○
408 (Request Time-out)	△	△	△
409 (Conflict)	○	○	○
410 (Gone)	○	○	○
411 (Length Required)	○	△	△
412 (Precondition Failed)	○	○	○
413 (Request Entity Too Large)	○	○	○
414 (Request-URI Too Large)	○	△	△
415 (Unsupported Media Type)	○	○	○
416 (Requested Range Not Satisfiable)	○	○	○
417 (Expectation Failed)	○	△	△
422 (Unprocessable Entity)	○	○	○
423 (Locked)	○	○	○
424 (Failed Dependency)	○	○	○
500 (Internal Server Error)	○	○	○
501 (Method Not Implemented)	○	○	○
502 (Bad Gateway)	○	○	△
503 (Service Temporarily Unavailable)	○※	○※	○※
504 (Gateway Timeout)	○	○	○
505 (HTTP Version Not Supported)	○	○	○
506 (Variant Also Negotiates)	○	○	○
507 (Insufficient Storage)	○	○	○
510 (Not Extended)	○	○	○

(凡例)

○：指定できる。

△：リバースプロキシ機能またはリダイレクタを使用する場合，バックエンドサーバまたは J2EE サーバからのエラーメッセージをカスタマイズするときに指定できる。

×：指定できない。

注※

流量制限機能が返すメッセージをカスタマイズする場合は，QOSResponse ディレクティブまたは QOSRedirect ディレクティブを使用してください。

このディレクティブ指定時には，次の点に留意してください。

- このディレクティブを複数指定する場合，同じエラー番号に異なる指定はできません。
- CGI プログラム内で設定されたエラーステータスに対しては，メッセージをカスタマイズできません。
- ローカル URL，フル URL の指定先でエラーとなる場合は，カスタマイズできません。
- ローカル URL の指定先でコンテントネゴシエーションが発生する場合は，エラーとなりカスタマイズできないことがあります。
- LoadModule ディレクティブによって動的に接続したモジュール内で設定されたエラーステータスに対しても，そのモジュールの実装方法によってメッセージをカスタマイズできない場合があります。
- フル URL の指定時には，ステータスコード 302 Found および Location ヘッダに新パスを設定した応答を返します。通常，ステータスコード 302 を受けた Web ブラウザは，Location ヘッダに指定されたアドレスに対して自動的にリダイレクトします。
- フル URL の指定時には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。
- リバースプロキシ機能を使用する場合，ProxyErrorOverride On を指定する必要があります。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
ErrorDocument 500 ”Server Error.”
ErrorDocument 404 /missing.html
ErrorDocument 403 http://some.other_server.com/subscription_info.html
ErrorDocument 404 http://[2001::123:4567:89ab:cdef]/missing.html
```

(2) ErrorLog {ファイル名|パイプ}

～《logs/error_log》 (UNIX 版)

～《logs/error.log》 (Windows 版)

(a) 内容

エラーログを出力するファイル名を指定します。出力するログの内容は，LogLevel ディレクティブで選択できます。

ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

ファイル名：エラーログを格納するファイル名を指定します。ServerRoot ディレクティブ指定値からの相対パスで指定できます。

パイプ：標準入力からエラーログ情報を受け取るプログラムを"|プログラム名"の形式で指定します。Windows 版での注意事項は，CustomLog ディレクティブを参照してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
ErrorLog logs/error.log
```

(3) ExtendedStatus {On | Off}

(a) 内容

server-status ハンドラによるステータス表示形式で，それぞれのリクエストの拡張ステータス情報を表示するかどうかを指定します。

On：拡張ステータス情報を表示します。この場合，クライアントの IP アドレスが IPv6 アドレスでも表示します。ただし，最大表示数は 31 バイトです。

Off：拡張ステータス情報を表示しません。

(b) 記述できる場所

httpsd.conf

(4) ExpiresActive {On | Off}

(a) 内容

レスポンスに Expires ヘッダおよび Cache-Control ヘッダを追加するかどうかを指定します。

On：Expires ヘッダおよび Cache-Control ヘッダを追加します。

Off：Expires ヘッダおよび Cache-Control ヘッダを追加しません。

(b) 注意事項

- 有効期限設定機能を使用するためには mod_expires モジュールの組み込みが必要です。有効期限設定機能の詳細は，「4.11 有効期限設定機能」を参照してください。
- ExpiresDefault ディレクティブまたは ExpiresByType ディレクティブを指定していない場合は，ExpiresActive ディレクティブに On が指定されていても，レスポンスに Expires ヘッダおよび Cache-Control ヘッダは追加されません。

(c)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d)　上書き許可

Indexes レベル

(5)　ExpiresByType　MIME タイプ　{A ｜ M}　時間

～((0－2147483647))（単位：秒）

(a)　内容

レスポンスに Expires ヘッダおよび Cache-Control ヘッダを追加する場合に，指定する MIME タイプのドキュメントに対する有効期限を指定します。このディレクティブは ExpiresActive ディレクティブで On を指定している場合に有効になります。ExpiresDefault ディレクティブで設定されたデフォルトの有効期限は，この設定によって MIME タイプ別に上書きされます。

基準時刻を A または M で指定し，基準時刻から有効期限までの時間を秒単位で指定します。A または M と，時間との間に空白は入りません。

A：クライアントがアクセスした時刻を基準時刻とします。

M：ファイルを最後に修正した時刻を基準時刻とします。

(b)　注意事項

有効期限設定機能を使用するためには mod_expires モジュールの組み込みが必要です。有効期限設定機能の詳細は，「4.11　有効期限設定機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d)　上書き許可

Indexes レベル

(e)　指定例

```
ExpiresByType text/html A604800
```

(6)　ExpiresDefault　{A ｜ M}　時間

～((0－2147483647))（単位：秒）

(a) 内容

レスポンスに Expires ヘッダおよび Cache-Control ヘッダを追加する場合に，デフォルトの有効期限を指定します。このディレクティブは ExpiresActive ディレクティブで On を指定している場合に有効になります。この設定は ExpiresByType ディレクティブによって MIME タイプごとに上書きされます。

基準時刻を A または M で指定し，基準時刻から有効期限までの時間を秒単位で指定します。A または M と，時間との間に空白は入りません。

A：クライアントがアクセスした時刻を基準時刻とします。

M：ファイルを最後に修正した時刻を基準時刻とします。

(b) 注意事項

有効期限設定機能を使用するためには mod_expires モジュールの組み込みが必要です。有効期限設定機能の詳細は，「4.11 有効期限設定機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d) 上書き許可

Indexes レベル

(e) 指定例

```
ExpiresDefault A604800
```

(7) FancyIndexing {On | Off}

(a) 内容

ディレクトリインデクスを表示する場合に，整形表示（ファンシーインデクス）をするかどうかを指定します。

On：整形表示をします。

Off：整形表示をしません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
FancyIndexing On
```

整形表示機能を使用します。

(8) FileETag 〔{+|-}〕オプション 〔〔{+|-}〕オプション …〕

～《MTime Size》

(a) 内容

ETag レスポンスヘッダフィールドを作成するために使用されるファイル属性値を指定します。このディレクティブが指定されていない場合，ETag レスポンスヘッダフィールドにはファイルの最終更新時刻およびバイト数が設定されます。

オプションに＋－を指定しない場合は，オプションで指定した属性値が使用されます。

オプションに＋－を指定する場合は，FileETag ディレクティブによって設定された属性値を変更できます。

＋：設定されている属性値にオプションで指定した属性値が追加されます。

－：設定されている属性値からオプションで指定した属性値が削除されます。

指定できるオプションの一覧を次に示します。

オプション	意味
Inode	ファイルに割り振られた一意な ID が含まれます。
Mtime	ファイルの最終更新時刻が含まれます。
Size	ファイルのバイト数が含まれます。
All	Inode，Mtime，Size のオプションがすべて有効になります。
None	Etag ヘッダが付きません。

(b) 注意事項

- FileETag ディレクティブの Inode オプションを有効にした場合，負荷分散をしている Web サーバ環境などで，同一のコンテンツを要求するごとに，異なる ID が Etag ヘッダに含まれることがあります。このため，同一コンテンツでありながらその Etag ヘッダの内容が異なり，ブラウザやプロキシでのキャッシングにとって不都合となることがあります。この場合，FileETag ディレクティブによって，Inode オプションを無効にするように指定することで回避できます。
- ＋－を使用しないでこのディレクティブを複数指定すると，最後に指定したディレクティブだけが有効になります。
- －を付加した属性値だけを指定した場合は，このディレクティブを指定していない場合と同じ状態になります。

- All オプションと None オプションには，＋－を指定できません。
- オプションに'-Inode -Mtime -Size'と指定した場合は，このディレクティブを指定していない場合と同じ状態になります。ETag レスポンスヘッダフィールドにはファイルの最終更新時刻およびバイト数が設定されます。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d) 上書き許可

FileInfo レベル

(e) 指定例

(例 1)

```
FileETag Inode Mtime Size
FileETag -Inode
```

この指定では，ファイルの最終更新時刻およびバイト数が属性値として使用されます。

(例 2)

```
FileETag Inode Mtime
FileETag Size
```

この指定では，ファイルのバイト数が属性値として使用されます。

(例 3)

```
FileETag All
FileETag -Inode -Mtime -Size
```

この指定では，ファイルの最終更新時刻およびバイト数が属性値として使用されます。

(9) ForceType MIME タイプ

(a) 内容

<Directory>ブロックまたはアクセスコントロールファイルに定義し，特定のディレクトリ下のすべてのコンテンツに対して使用する MIME タイプを指定します。none を指定すると，それまでの ForceType ディレクティブの指定が無効になります。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(10) Group グループ名 U

～《#-1》

(a) 内容

サーバプロセスが動作するときのグループ名を指定します。

(b) 記述できる場所

httpsd.conf

(c) 指定例

Group nogroup	グループ名nogroupを定義

(11) H2Direct {On | Off}

《h2 は off，h2c は on》

(a) 内容

HTTP/2 ダイレクトモードを有効にするかどうかを指定します。HTTP/2 プロトコルの通信を有効にしたいバーチャルホストの<VirtualHost>ブロック内で指定する必要があります。

HTTP/2 ダイレクトモードとは，接続でサーバが受信した最初のバイトが HTTP/2 プリアンブルと一致する場合に，HTTP/2 プロトコルに切り替えます。これは h2c の場合に有効です。

h2c をサポートするサーバについて帯域外の知識を持っているクライアントの場合，HTTP/2 ダイレクトモードを有効にすることで，クライアントは HTTP/1.1 アップグレードを実行する必要がなくなり，パフォーマンスが向上し，リクエストボディに対するアップグレード制限が回避されます。

On：HTTP/2 ダイレクトモードを有効にします。

Off：HTTP/2 ダイレクトモードを無効にします。

(b) 注意事項

Protocols ディレクティブで h2 または h2c が有効になっていない場合，HTTP/2 プリアンブルをチェックしません。この場合，H2Direct ディレクティブの設定は意味がありません。

デフォルトでは h2 は off，h2c は on であり，h2 と h2c で設定が異なります。H2Direct ディレクティブを設定した場合，h2 と h2c の両方で同じ指定値が設定されます。

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17 HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(12)　H2MaxWorkerIdleSeconds　時間

～((1－2147483647))《600》（単位：秒）

(a)　内容

HTTP/2 プロトコル処理専用のワーカスレッドを終了させるまでの最大秒数を指定します。H2MinWorkers ディレクティブ指定値を超えているワーカスレッドだけに適用されます。

(b)　注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf

(13)　H2MaxWorkers　スレッド数

～((1－150000))　W

～((1－10000))　U

《H2MinWorkers ディレクティブ指定値×3/2》

(a)　内容

サーバプロセスごとに生成する HTTP/2 プロトコル処理に使用するワーカスレッドの最大数を指定します。

(b)　注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf

(14)　H2MinWorkers　スレッド数

～((1－15000))　W

～((1－1000))　U

《ThreadsPerChild ディレクティブ指定値》

(a) 内容

サーバプロセスごとに生成する HTTP/2 プロトコル処理に使用するワーカスレッドの最小数を指定します。

(b) 注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17 HTTP/2 プロトコル通信機能」を参照してください。

(c) 記述できる場所

httpsd.conf

(15) H2Padding 数

～((0－8))《0》

(a) 内容

デフォルトの 0 では，パディングバイトはペイロードフレーム（HEADERS，DATA，PUSH_PROMISE など）に追加されません。

このディレクティブで 1～8 の数値を設定すると，各フレームに[0, 2^指定値－1]の範囲の乱数が追加されます。

パディングするバイト数が多いほど，メッセージ長がわかりにくくなりますが，トラフィックも増えます。

(b) 注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17 HTTP/2 プロトコル通信機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(16) H2SerializeHeaders {On | Off}

(a) 内容

HTTP/2 リクエストを HTTP/1.1 でシリアル化して処理するかどうかを指定します。HTTP/1.1 でシリアル化することでパフォーマンスは低下しますが，下位互換性が高まります。ただし，このディレクティブの機能は今後無効となるため，On を指定することは推奨しません。

On：HTTP/2 リクエストを HTTP/1.1 でシリアル化して処理します。

Off：HTTP/2 リクエストを HTTP/1.1 でシリアル化しません。

(b)　注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(17)　H2StreamMaxMemSize　最大メモリサイズ

～((1024－2147483647))《32768》

(a)　内容

ストリームのメモリにバッファリングされる送信データの最大サイズを指定します。ストリーム処理は，制限に達するとフリーズして，バッファに入れられたデータがクライアントに送信された場合に続行されます。

(b)　注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(18)　H2Upgrade　{On ｜ Off}

《h2 は off，h2c は on》

(a)　内容

HTTP/1.1 から HTTP/2 へのアップグレードを有効にするかどうかを指定します。

On：HTTP/1.1 から HTTP/2 へのアップグレードを有効にします。

Off：HTTP/1.1 から HTTP/2 へのアップグレードを無効にします。

プロトコルを切り替えるこの方法は HTTP/1.1 で定義されており，クライアントが Upgrade ヘッダを使用して，別のプロトコルを使用する意思があることを通知します。これは，HTTP/1.1 接続の任意のリクエストで発生する可能性があります。

このディレクティブでの HTTP/2 へのアップグレードは，Protocols ディレクティブで h2 または h2c が有効になっている場合にだけ効果があります。

(b)　注意事項

リクエストボディを含む POST と PUT は，HTTP/2 へのアップグレードができません。この場合にアップグレードが必要なときは，H2Direct ディレクティブを設定してください。

デフォルトは h2 は off，h2c は on であり，h2 と h2c で設定が異なります。H2Upgrade ディレクティブを設定した場合，h2 と h2c の両方で同じ指定値が設定されます。

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(19)　H2WindowSize　サイズ

～((65535－2147483647))《65535》

(a)　内容

クライアントからサーバへのフロー制御に使用されるウィンドウサイズをバイト単位で指定します。クライアントは，サーバが通知した使用可能なサイズの制限に到達するとストリームでの送信を停止します。

(b)　注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17　HTTP/2 プロトコル通信機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(20)　Header　〔always〕｛set｜append｜add｝ヘッダ　ヘッダ値｜edit ヘッダ　正規表現 置換文字列｜unset　ヘッダ｝〔env=〔!〕環境変数〕

(a)　内容

レスポンスヘッダをカスタマイズする場合に指定します。200 番台のステータスコード応答時に有効となります。ただし，リバースプロキシ機能またはリダイレクタ機能を使用する場合は，バックエンドサーバまたは J2EE サーバが返したステータスコードの値にかかわらず，有効となります。

always：エラーのステータスコード応答時にも有効となります。ただし，リバースプロキシ機能またはリダイレクタ機能を使用する場合は，バックエンドサーバまたは J2EE サーバが返したレスポンスヘッダをカスタマイズしません。

set：ヘッダを設定します。ヘッダがある場合は，指定したヘッダ値に書き換えます。

append：存在するヘッダにヘッダ値を追加します。存在するヘッダ値との間は，コンマで区切られます。ヘッダがない場合は，ヘッダを設定します。

add：ヘッダがあっても，別の行にヘッダを設定します。同じヘッダを複数行設定する場合に使用します。

edit：指定したヘッダがある場合，ヘッダ値の正規表現に合致した部分を置換文字列に置き換えます。正規表現で括弧（）を使用してグループ化している場合，その i 番目のグループの表現にマッチした文字列を，置換文字列で$i を使用して参照できます。i には 1 から 9 までの数字を指定します。

unset：指定したヘッダがある場合，そのヘッダをすべて削除します。

env=環境変数：指定した環境変数が設定されている場合に，Header ディレクティブで指定した内容を実行します。

env=!環境変数：指定した環境変数が設定されていない場合に，Header ディレクティブで指定した内容を実行します。

ヘッダ値に空白がある場合は，"（引用符）で囲む必要があります。ヘッダ値は文字だけから成る文字列，フォーマット指示子を含む文字列または両方から成る文字列を指定できます。フォーマット指示子を次に示します。

フォーマット	意味
%t	リクエストを受け取った時刻を，1970 年 1 月 1 日 0 時 0 分 0 秒(GMT：Greenwich Mean Time)から経過した時間で表示する。単位はマイクロ秒。先頭には"t="が付けられる。
%D	リクエスト処理に掛かった時間を表示する。単位はマイクロ秒。先頭には"D="が付けられる。
%{env_name}e	環境変数 env_name の値。

(b)　注意事項

ヘッダカスタマイズ機能を使用するためには mod_headers モジュールの組み込みが必要です。ヘッダカスタマイズ機能については，「4.10　ヘッダカスタマイズ機能」を参照してください。

CGI スクリプトからのレスポンスヘッダをカスタマイズする場合は，always を指定してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d)　上書き許可

FileInfo レベル

(e) 指定例

```
Header set Cache-Control no-cache
```

(21) HeaderName ファイル名

(a) 内容

ディレクトリインデクス表示時のヘッダに付けるコメントを記述したファイルのファイル名（パス情報なし）を指定します。HTML またはプレーンテキストで記述できます。ただし，AddType ディレクティブまたは TypesConfig ディレクティブで指定したファイルで，MIME タイプが正しく定義されている必要があります。プレーンテキストでコメントを作成した場合，ディレクトリインデクス表示の HTML には<PRE>タグが追加されます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
HeaderName HEADER.html
```

各ディレクトリ下の HEADER.html の内容をヘッダに付けます。

(22) HostnameLookups {On | Off | double}

(a) 内容

CGI の REMOTE_HOST 環境変数の IP アドレスおよびログファイルに出力するクライアントの IP アドレスをホスト名に変換するために，ホスト名のルックアップの逆引きをするかどうかを指定します。なお，逆引きを使用する場合，レスポンスが遅くなります。

On：IP アドレスをホスト名に変換します。

Off：IP アドレスをホスト名に変換しません。

double：IP アドレスをホスト名に変換します。その後，再変換し，IP アドレスが正しいかどうかを確認します。

このディレクティブは，IPv6 アドレスにも対応しています。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>

(c) 指定例

```
HostnameLookups Off
```

IP アドレスをホスト名に変換しません。

(23) HWSErrorDocumentMETACharset {On | Off | 文字セット}

W

(a) 内容

エラーが発生したときに Web ブラウザへ表示するメッセージ（以降，エラードキュメントと呼びます）についての文字セットを設定します。文字セットは，エラードキュメント中に META タグで charset=の値として設定されます。ErrorDocument ディレクティブで，カスタマイズされたエラードキュメントは，このディレクティブの META タグによる文字セットの設定対象とはなりません。

On：文字セット ISO-8859-1 を設定します。

Off：文字セットを設定しません。

文字セット：指定した文字セットを設定します。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
HWSErrorDocumentMETACharset ISO-2022-JP
```

(24) HWSErrorLogClientAddr X-Forwarded-For

(a) 内容

バックエンドサーバで，エラーログに出力するメッセージテキストの"[client クライアントアドレス]"を"[X-Forwarded-For X-Forwarded-For ヘッダ値]"に変更するよう設定します。

負荷分散装置やプロキシサーバを介してバックエンドサーバがリクエストを受信すると，バックエンドサーバがエラーログに出力する「クライアント IP アドレス」がリクエストを送信したクライアントの IP アドレスではなく，負荷分散装置やプロキシサーバの IP アドレスになる場合があります。ただし，負荷分散装置やプロキシサーバの中には接続元のクライアント IP アドレスを X-Forwarded-For ヘッダに付加する場

合があるため，出力内容を X-Forwarded-For X-Forwarded-For ヘッダの値へ変更することで，接続元のクライアント IP アドレスが出力されるようにします。

X-Forwarded-For：エラーログに出力する"[client クライアントアドレス]"を"[X-Forwarded-For X-Forwarded-For ヘッダ値]"に変更します。

(b) 注意事項

X-Forwarded-For ヘッダを受信する前にエラーが発生した場合や，一部のメッセージでは変更できません。

(c) 記述できる場所

httpsd.conf

(25) HWSGracefulStopLog {On | Off}

(a) 内容

計画停止時に，強制停止待ち時間を経過したあとに強制停止させたリクエスト情報を，エラーログファイルに出力するかどうかを指定します。

On：強制停止させたリクエスト情報をエラーログファイルに出力します。

Off：強制停止させたリクエスト情報をエラーログファイルに出力しません。

(b) 注意事項

計画停止時に強制停止させた HTTP/2 通信のリクエスト情報は，エラーログファイルに出力できません。

(c) 記述できる場所

httpsd.conf

(d) 指定例

```
HWSGracefulStopLog On
```

(26) HWSGracefulStopTimeout 強制停止時間

～((0－3600))《300》（単位：秒）

(a) 内容

計画停止時に，実行中のリクエストを直ちに終了するまでの強制停止待ち時間を秒単位で指定します。なお，0 を指定すると，強制停止待ち時間の上限は設定されません。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
HWSGracefulStopTimeout 600
```

(27) HWSH2SendGoaway リクエスト処理数

～((0-10000))《10000》

(a) 内容

コネクションでの，HTTP/2 通信の新たなリクエストの受付を終了させる，リクエスト処理完了数を指定します。リクエスト処理完了数がディレクティブ指定値に到達した場合，クライアントに GOAWAY フレームを送信します。GOAWAY フレームを送信までに受け付けたリクエストは処理を中断せずに実行します。

HTTP/2 通信を有効とする設定でメモリ使用量が多い傾向にある場合は，このディレクティブに小さな値を指定すると軽減されることがあります。

なお，0 を指定すると，リクエスト処理完了数によるクライアントへの GOAWAY フレーム送信はしません。

(b) 注意事項

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17 HTTP/2 プロトコル通信機能」を参照してください。

(c) 記述できる場所

httpsd.conf

(28) HWSImapMenuCharset 文字セット

～《ISO-8859-1》

(a) 内容

次の場合のメニュー表示に対する文字セットを指定します。

- イメージマップファイルの指定値に map を指定した場合
- イメージマップ画像の座標(0,0)をマウスでポイントした場合
- 座標指定のない形でイメージマップファイルがリクエストされた場合

文字セットは，レスポンスの Content-Type ヘッダで charset=の値として設定されます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
HWSImapMenuCharset SHIFT_JIS
```

(29) HWSKeepStartServers {On | Off}

(a) 内容

サーバプロセスの稼働数を StartServers ディレクティブに指定した数だけ維持するかどうかを指定します。

On：StartServers ディレクティブに指定した数だけ，稼働しているサーバプロセスが維持されます。サーバプロセス数が StartServers ディレクティブ指定値より小さくなった場合，新しいプロセスが生成されます。

prefork MPM

この機能は，プロセス数に関する各ディレクティブの指定値が，次の関係にある場合に有効です。

MinSpareServers ＜ StartServers≦MaxClients

かつ

MinSpareServers ＜ MaxSpareServers≦MaxClients

StartServers ディレクティブ設定値が，MinSpareServers ディレクティブ設定値より小さい場合は，MinSpareServers ディレクティブの値でサーバプロセス数が維持されます。

worker MPM

この機能は，プロセス数およびスレッド数に関する各ディレクティブの指定値が，次の関係にある場合に有効です。

MinSpareThreads ＜ StartServers × ThreadsPerChild ≦ MaxClients

かつ

MinSpareThreads ＜ MaxSpareThreads ≦ MaxClients

StartServers ディレクティブ × ThreadsPerChild ディレクティブの値が，MinSpareThreads ディレクティブ設定値より小さい場合は，MinSpareThreads ディレクティブ設定値に従ってサーバプロセス数が維持されます。

Off：StartServers ディレクティブに指定した数の稼働しているサーバプロセスは維持されません。

(b) 記述できる場所

httpsd.conf

(30) HWSLogSSLVerbose {On | Off}

(a) 内容

クライアントとサーバ間の SSL ハンドシェイク処理中に，ログに出力されるエラーのうち info レベルおよび error レベルのエラーについて，詳細情報を表示するかどうかを指定します。SSL を有効にする場合には，このディレクティブを On に設定することを推奨します。

On：詳細情報を表示します。

Off：詳細情報を表示しません。

(b) 記述できる場所

httpsd.conf

(31) HWSLogTimeVerbose {On | Off}

(a) 内容

エラーログ※とリクエストログの時刻，アクセスログのアクセス時刻，リクエスト処理に掛かった時間(%T)，およびリクエスト処理を開始した時刻(%t)をミリ秒まで表示するかどうかを指定します。なお，起動および再起動中に出力されるメッセージでは，このディレクティブで On を指定してもミリ秒まで表示されない場合があります。

注※ ErrorLog ディレクティブで指定するエラーログが対象になります。ScriptLog ディレクティブで指定する CGI スクリプトのエラーログは対象になりません。

On：時刻および時間をミリ秒まで表示します。

Off：時刻および時間を秒まで表示します。

(b) 記述できる場所

httpsd.conf

(32) HWSMaxQueueSize リクエストキューサイズ W

～((0－2147483647))《8192》

(a) 内容

クライアントからのリクエストについての最大の待ちリクエスト数を指定します。0 を指定した場合は，無制限となります。このディレクティブで指定したリクエストキューサイズを超えたクライアントからのリクエストは，サーバ側で切断されます。

(b) 記述できる場所

httpsd.conf

(33) HWSNotModifiedResponseHeaders ヘッダ名〔ヘッダ名 …〕

(a) 内容

ステータスコード 304 Not Modified をクライアントへ送信する際に付加するレスポンスヘッダを指定します。

なお，次のヘッダについては，このディレクティブに指定がなくてもレスポンスに付加します。ただし，必ず付加するのではなく，外部モジュールまたはサーバ内部などで設定された場合にだけ付加します。

- Date
- Server
- Connection
- Keep-Alive
- ETag
- Content-Location
- Expires
- Cache-Control
- Vary
- Warning
- WWW-Authenticate
- Proxy-Authenticate

(b) 記述できる場所

httpsd.conf

(c) 指定例

HWSNotModifiedResponseHeaders Set-Cookie Set-Cookie2

(d) 注意事項

HTTP/2 通信の場合，このディレクティブは機能しません。

(34) HWSPrfId 文字列

《PRF_ID》

(a) 内容

PRF デーモン起動時に PRF 識別子に指定した文字列を指定します。

リダイレクタのモジュールを LoadModule で組み込んだ場合，このディレクティブの指定は無効になります。

(b) 記述できる場所

httpsd.conf

(35) HWSProxyPassReverseCookie パス名

(a) 内容

リバースプロキシを使用する場合，リバースプロキシはバックエンドサーバから受信した Set-Cookie ヘッダを変換します。これは，Web ブラウザが Set-Cookie ヘッダを受信したあとに，リバースプロキシを経由するバックエンドサーバへのリクエストに対して，クッキーを送信させるために必要になります。

パス名：ProxyPass ディレクティブと同じパス名を指定します。*および正規表現は使用できません。

(b) 注意事項

リバースプロキシを使用するためには mod_proxy モジュールおよび mod_proxy_http モジュールの組み込みが必要です。リバースプロキシの詳細は，「4.7 リバースプロキシの設定」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(36) HWSRequestLog {ファイル名 | パイプ}

(a) 内容

リクエストログを出力するファイル名を指定します。リクエストログとは，モジュールトレース，リクエストトレース，I/O フィルタトレースおよびプロキシトレースの総称です。出力するリクエストログの種別は，HWSRequestLogType ディレクティブで選択できます。

ファイル名：リクエストログを出力するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスを指定できます。

パイプ：標準入力からリクエストログ情報を受け取るプログラムを「|プログラム名」の形式で指定します。Windows 版での注意事項は，CustomLog ディレクティブを参照してください。

(b) 注意事項

- このディレクティブを省略した場合のモジュールトレース出力先は，ErrorLog ディレクティブで指定したファイルになります。モジュールトレースの採取レベルは，LogLevel ディレクティブで指定してください。モジュールトレースの詳細は，「4.2.6　モジュールトレースの採取」を参照してください。
- リクエストトレース，I/O フィルタトレースおよびプロキシトレースの出力先を，ErrorLog ディレクティブで指定したファイルにすることはできません。

(c) 記述できる場所

httpsd.conf

(37) HWSRequestLogType　トレース種別 〔トレース種別 …〕

～《module-info request proxy》

(a) 内容

HWSRequestLog ディレクティブで設定するリクエストログに出力するトレース種別を指定します。トレース種別を次に示します。

トレース種別	内容
module-debug	内部モジュールに対するモジュールトレースおよび module-info 相当のトレースを出力します。出力量が多いため，デバッグ目的以外では指定しないでください。
module-info	外部モジュールと CGI プログラム実行時のモジュールトレースを出力します。
request	リクエスト処理開始時およびリクエスト処理完了時にトレースを出力します。また，HTTP/1.1 通信の KeepAlive 接続の場合は，次のリクエストライン受信完了時にもトレースを出力します。これらのトレースをリクエストトレースと呼びます。
filter	モジュールが実装している入出力フィルタ関数の実行契機を示す I/O フィルタトレースを出力します。出力量が多いため，デバッグ目的以外では指定しないでください。
proxy	リバースプロキシ機能使用時にリバースプロキシに関するトレースを出力します。
none	リクエストログを採取しません。

(b) 注意事項

指定したトレース種別に none が含まれている場合，リクエストログを一切採取しません。

(c) 記述できる場所

httpsd.conf

(38) HWSSetEnvIfIPv6 リクエスト値 IPv6アドレス 環境変数〔=値〕〔環境変数〔=値〕…〕

(a) 内容

クライアントまたはサーバのIPv6アドレスを基に環境変数を定義します。リクエスト値がIPv6アドレスで表した条件を満たす場合，指定した環境変数を設定します。設定する値のデフォルト値は1です。環境変数の前に「!」が付いたときは，その環境変数の設定を解除します。

リクエスト値として，次に示す値を指定できます。

リクエスト値	意味
Remote_Addr	クライアントのIPv6アドレス
Server_Addr	リクエストを受信したサーバのIPv6アドレス

IPv6アドレスは，[ ]で囲まないで指定してください。なお，IPv6アドレスの後ろに，10進数でプリフィックス長も指定できます。プリフィックス長は，「IPv6アドレス/プリフィックス長」の形式で指定します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfoレベル

(d) 指定例

```
HWSSetEnvIfIPv6 Remote_Addr 2001:0:0:1230::/64 IPV6_CLIENT
```

クライアントのIPv6アドレスが2001:0:0:1230から始まる場合，環境変数IPV6_CLIENTを設定します。

(39) HWSSuppressModuleTrace モジュールファイル名 〔all | hook | handler〕

(a) 内容

モジュールトレースの出力を抑止するモジュールファイル名および抑止する関数種別を指定します。

all：指定したモジュールが出力するモジュールトレースをすべて抑止します。

hook：指定したモジュールが出力するモジュールトレースのうち，handler関数以外のモジュールトレースを抑止します。関数の種別については，「4.2.6 モジュールトレースの採取」の表4-4を参照してください。

handler：指定したモジュールが出力するモジュールトレースのうち，handler 関数のモジュールトレースを抑止します。関数の種別については，「4.2.6　モジュールトレースの採取」の表 4-4 を参照してください。

モジュールファイル名には，エラーログまたはリクエストログに出力されるモジュールファイル名称を指定します。次の例のモジュールトレースを抑止する場合は，モジュールファイル名に"mod_example.c"を指定します。

(例)

```
[Mon Dec 18 14:57:14 2006] [info] hws : module --> (mod_example.c[12])(1896)
[Mon Dec 18 14:57:14 2006] [info] hws : module <-- (mod_example.c[12])(1896)(-1)
```

HTTP Server が標準提供している外部モジュールとモジュールファイル名の対応を次に示します。

表 6-5　HTTP Server が標準提供している外部モジュールとモジュールファイル名の対応

モジュール名	モジュールファイル名
mod_expires.so	mod_expires.c
mod_headers.so	mod_headers.c
mod_hws_qos.so	mod_hws_qos.c
mod_proxy.so	mod_proxy.c
mod_proxy_http.so	モジュールトレースは出力されません。

HTTP Server が標準提供している外部モジュール以外を使用する場合は，そのモジュールのトレースが出力される可能性があります。また，LogLevel ディレクティブに debug を設定または HWSRequestLogType ディレクティブに module-debug を設定している場合は，内部モジュールに対するトレースも出力されます。

なお，このディレクティブは，複数指定できます。同じモジュールファイル名を指定した場合は，あとに指定したものが有効となります。

(b)　注意事項

CGI プログラム実行時のモジュールトレースは抑止できません。

(c)　記述できる場所

httpsd.conf

(d)　指定例

(例 1)

```
HWSSuppressModuleTrace mod_proxy.c all
```

この指定では，プロキシモジュール内のすべての関数に対するモジュールトレースを抑止します。

(例 2)

```
HWSSuppressModuleTrace mod_proxy.c hook
```

この指定では，プロキシモジュール内の handler 以外の関数に対するモジュールトレースを抑止します。

(40) HWSTraceIdFile　ファイル名

～《logs/hws.trcid》

(a) 内容

トレース採取のための共有メモリ ID を格納するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

このファイルは複数の Web サーバでは共有できません。同一 ServerRoot ディレクティブ指定で複数の Web サーバを起動する場合は，このディレクティブで異なるファイル名を指定する必要があります。

(b) 記述できる場所

httpsd.conf

(41) HWSTraceLogFile　ファイル名

～《logs/hws.trclog》

(a) 内容

サーバプロセスが異常終了した場合に共有メモリに採取されたトレースを出力するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

このファイルは複数の Web サーバでは共有できません。同一 ServerRoot ディレクティブ指定で複数の Web サーバを起動する場合は，このディレクティブで異なるファイル名を指定する必要があります。

トレースは複数のファイルにラップアラウンドして出力します。

UNIX 版では，最大 5 ファイル出力します。出力するファイルは「指定したファイル名.nn」のファイル名となります。nn は 01 から 05 までです。HTTP Server の起動時には，「指定したファイル名.01」がカレントの出力ファイル名となります。カレントの出力ファイル名が「指定したファイル名.nn」であった場合にトレースをファイルに出力すると，次のカレントのファイル名は「指定したファイル名.nn+1」になります。なお，「指定したファイル名.nn」が.05 の場合には，次のカレントのファイル名は「指定したファイル名.01」になります。

Windows 版では，最大 2 ファイル出力します。出力するファイルは「指定したファイル名.01」または「指定したファイル名.02」のファイル名になります。HTTP Server の起動時には，「指定したファイル名.01」がカレントの出力ファイル名となります。カレントの出力ファイル名が「指定したファイル名.01」

のときにトレースをファイルに出力すると，次のカレントのファイル名は「指定したファイル名.02」になります。なお，カレントの出力ファイル名が「指定したファイル名.02」のときにトレースをファイルに出力すると，次のカレントのファイル名は「指定したファイル名.01」になります。

(b) 記述できる場所

httpsd.conf

(42) HWSWebSocketLog {ファイル名 | パイプ}

(a) 内容

WebSocket 通信に関するログを出力するファイル名を指定します。

ファイル名：WebSocket 通信に関するログを出力するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスを指定できます。

パイプ：標準入力から WebSocket 通信に関するログ情報を受け取るプログラムを「|プログラム名」の形式で指定します。

(b) 注意事項

WebSocket プロキシ機能を使用するためには，mod_proxy モジュール，mod_proxy_http モジュール，および mod_proxy_wstunnel モジュールの組み込みが必要です。

UNIX 版の場合

```
LoadModule proxy_module libexec/mod_proxy.so
LoadModule proxy_http_module libexec/mod_proxy_http.so
LoadModule proxy_wstunnel_module libexec/mod_proxy_wstunnel.so
```

Windows 版の場合

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

(c) 記述できる場所

httpsd.conf

(43) IdentityCheck {On | Off} U

(a) 内容

クライアントホストの identd デーモンを使用してクライアントの確認をするかどうかを指定します。 ident については，RFC1413 を参照してください。

ただし，クライアントホストが IPv6 アドレスの場合は，On を指定しても identd デーモンを使用してクライアントの確認をしません。また，ログフォーマットに%l を指定している場合，CGI 環境変数 REMOTE_IDENT には「unknown」を出力します。

On：identd デーモンを使用してクライアントの確認をします。

Off：identd デーモンを使用してクライアントの確認をしません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>

(44) ImapBase {map | referer | URL}

(a) 内容

イメージマップファイルの base 行のデフォルトを指定します。

map：マップファイルの場所

referer：ドキュメントの場所（イメージマップを表示した HTML ファイルの場所）

URL：指定した URL

URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(45) ImapDefault {error | nocontent | map | referer | URL}

(a) 内容

イメージマップファイルの default 行のデフォルトを指定します。

error：標準のエラーメッセージを表示します（ステータスコード 500 Server Error を応答します）。

nocontent：リクエストを無視します（ステータスコード 204 No Content を応答します）。

map：マップファイル中の URL をメニュー表示します。

referer：ステータスコード 302 Found を応答します。

URL：指定した URL のコンテンツを表示します。

URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(46) ImapMenu {none | formatted | semiformatted | unformatted}

(a) 内容

イメージマップファイルの指定値に map を与えた場合またはイメージマップ画像の(0,0)座標をマウスでポイントした場合のメニュー表示を指定します。座標指定のない形でイメージマップファイルがリクエストされた場合の動作もこの設定に従います。

none：メニューは生成しません。このときの動作は，マップファイル中の default 行の指定に従います。

formatted：ヘッダおよびリンク一覧を表示します。マップファイル中のコメントは無視されます。

semiformatted：リンク一覧を表示します。マップファイル中のコメントも表示します。

unformatted：マップファイル中に HTML を記述することで，メニューの形式を自由に設定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(47) Include ファイル名

(a) 内容

ファイル名で指定したファイルをコンフィグファイルとして利用できるようにします。

ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。このディレクティブを複数指定する場合，マージされた内容が使用されます。ファイル内に同じディレクティブがある場合，後ろに指定した方で上書きされます。

(b) 記述できる場所

httpsd.conf

(48) IndexIgnore ファイル名 〔ファイル名 …〕

(a) 内容

ディレクトリインデクス表示時に，Web ブラウザに表示させないファイル名を指定します。正規表現でも指定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```

(49) IndexOptions 〔{+|-}〕オプション 〔〔{+|-}〕オプション …〕

(a) 内容

ディレクトリインデクスの整形表示機能のオプション設定をします。オプションの前に+を指定するかまたは+-を省略するとそのオプションが有効になります。デフォルトではすべてのオプションが無効です。指定できるオプションの一覧を次に示します。

表 6-6 オプション一覧

オプション	意味
Charset=文字セット ～《ISO-8859-1》 U ～《UTF-8》 W	インデクス表示するページの文字セットを指定します。HeaderName ディレクティブまたは ReadmeName ディレクティブで指定したファイルで使用している文字セットが，デフォルトの文字セット（UNIX 版：ISO-8859-1，Windows 版：UTF-8）と異なる場合は，このオプションで HeaderName ディレクティブまたは ReadmeName ディレクティブで指定したファイルと同じ文字セットを指定してください。このオプションでは，=文字セットを省略できません。また，-Charset 指定時も+Charset 指定時と同様の動作をします。
DescriptionWidth〔=｛文字数|*｝〕 《23，30，42 または 49》	ファイル説明文エリアの幅を文字数(1 文字＝ 1 バイト)で指定します。*を指定した場合は AddDescription ディレクティブで指定したファイル説明文の最大長に合わせて表示します。このオプションを省略した場合のファイル説明文エリアの幅は，23 バイト(ただし，SuppressSize 指定時＋ 7，SuppressLastModified 指定時＋ 19)です。 -DescriptionWidth 指定時は＝｛文字数|*｝を省略できます。この場合の表示幅は 23 バイトです。
FancyIndexing	ディレクトリインデクスの整形表示機能を有効にします。

オプション	意味
FoldersFirst	ファイルよりディレクトリを先にインデクス表示する場合に指定します。ただし，FancyIndexing が有効な場合だけです。
IconsAreLinks	ディレクトリインデクス整形表示時のアイコンをファイルに対するアンカーにします。
IconHeight〔=ピクセル数〕((>0))《22》	ディレクトリインデクス整形表示時のアイコンの高さをピクセル数で指定します。IconWidth オプションと一緒に指定します。インデクスを表示する HTML の IMG タグの HEIGHT 属性になります。
IconWidth〔=ピクセル数〕((>0))《20》	ディレクトリインデクス整形表示時のアイコンの幅をピクセルで指定します。IconHeight オプションと一緒に指定します。インデクスを表示する HTML の IMG タグの WIDTH 属性になります。
IgnoreCase	ディレクトリインデクス整形表示時に，ファイル名およびディレクトリ名の大文字と小文字の区別をしないで並べ替えます。
NameWidth〔=｛文字数 | *｝〕《23》	ファイル名およびディレクトリ名エリアの幅を文字数（1 文字= 1 バイト）で指定します。*を指定した場合はファイル名およびディレクトリ名の最大長に合わせて表示します。 =｛文字数 | *｝を省略する場合は必ず-NameWidth と指定してください。
ScanHTMLTitles	AddDescription ディレクティブの指定がない場合に，HTML ファイル中の<TITLE>タグを検索し，説明文として表示します。
SuppressColumnSorting	ファイル名，ディレクトリ名，最終更新日時，ファイルサイズおよびファイルの説明文の各カラムでインデクスを並べ替える機能を抑止します。
SuppressDescription	ファイルの説明文を表示しません。
SuppressHTMLPreamble	HeaderName ディレクティブが指定されている場合，HeaderName ディレクティブで指定されたファイルの内容と，自動生成される HTML ヘッダ部（<HTML>や<TITLE>など）が共に出力されます。このオプションは，HeaderName ディレクティブで指定されたファイルが HTML で記述されている場合，自動生成される HTML ヘッダ部の出力を抑制します。
SuppressLastModified	最終更新日時を表示させません。
SuppressSize	ファイルサイズを表示させません。
TrackModified	ディレクトリ表示のためのレスポンスの HTTP レスポンスヘッダに，Last-Modified 値と Etag 値を設定します。このオプションを指定すると，クライアントは HEAD リクエストでディレクトリはファイル構成の変更を確認できるため，クライアントのキャッシュ機能を有効に活用できます。このオプションはオペレーティングシステムとファイルシステムが stat()をサポートしている場合だけ有効です。

(b) 注意事項

- このディレクティブを複数指定する場合，同じファイル名に異なる文字列は指定できません。
- IconHeight，IconWidth，NameWidth で=値を指定する場合，-の指定はできません。

- 設定されたオプションは，httpsd.conf，<VirtualHost>，<Directory>，.htaccess の順で，また，上位ディレクトリから下位ディレクトリへ継承します。継承したオプションを最終的にマージして，インデクス整形表示形式を決定します。
- httpsd.conf で+-を付けてオプションを指定しても無効になります。ただし，httpsd.conf，<VirtualHost>，<Directory>，.htaccess の順で，また，下位ディレクトリに継承されます。継承されたオプション指定はマージ処理で有効になります。参照順位が下位の指定場所でオプションの指定がある場合または次に示すディレクティブのどれかの指定がある場合，マージ処理が実行されます。
  - AddAlt
  - AddAltByEncoding
  - AddAltByType
  - AddDescription
  - AddIcon
  - AddIconByEncoding
  - AddIconByType
  - DefaultIcon
  - HeaderName
  - ReadmeName

  (例)

  httpsd.conf ファイルに IndexOptions +FancyIndexing +IconsAreLinks を指定した場合，下位の指定場所でインデクス関係のディレクティブ指定がなければ FancyIndexing，IconsAreLinks は無効になります。

  httpsd.conf ファイルに IndexOptions +FancyIndexing +IconsAreLinks，かつ下位ディレクトリのアクセスコントロールファイルに，AddDescription "テキストファイル" *.txt を指定した場合，FancyIndexing，IconsAreLinks は有効になります。
- +-指定のない Charset，IconHeight，IconWidth，NameWidth ディレクティブを指定すると，その指定場所内でそのオプションが指定されている位置より前に指定されている+-付のオプション（Charset，IconHeight，IconWidth，NameWidth を除いて）は無効になります。

  (例)

  IndexOptions FancyIndexing -IconsAreLinks IconHeight IconWidth

  この場合，FancyIndexing，IconHeight，IconWidth ディレクティブが有効になります。IconsAreLinks の-指定は継承されません。
- 指定場所間で同じディレクトリのインデクスを対象にオプション指定した場合のマージ処理は，参照順位がより後方の指定場所で+-のないオプションを指定すると，先に指定したオプションは無効になります。ただし，IconHeight，IconWidth，NameWidth は無効になりません。

  (例 1)

  - httpsd.conf ファイルの指定

IndexOptions +FancyIndexing +IconsAreLinks

- アクセスコントロールファイルの指定

  IndexOptions FancyIndexing SuppressLastModified

  これらを指定した場合，IconsAreLinks は無効になります。FancyIndexing，SuppressLastModified は有効になります。

(例 2)

- httpsd.conf ファイルの指定

  IndexOptions SuppressColumnSorting +FancyIndexing +IconsAreLinks

- アクセスコントロールファイルの指定

  IndexOptions FancyIndexing SuppressLastModified

これらを指定した場合，SuppressColumnSorting，IconsAreLinks は無効になります。また，FancyIndexing，SuppressLastModified は有効になります。

- 指定場所間で，同じディレクトリのインデクスを対象にオプション指定した場合のマージ処理は，同じオプションに対して+と-の両方を指定すると-指定が有効になります。

(例)

- httpsd.conf ファイルの指定

  IndexOptions +FancyIndexing -IconsAreLinks

- アクセスコントロールファイルの指定

  IndexOptions +IconsAreLinks

これらを指定した場合，IconsAreLinks は無効になります。

- 同じ指定場所で+-を指定しないオプションを指定すると，Charset，IconHeight，IconWidth，NameWidth ディレクティブ以外の+-で指定したオプションは無効になります。

(例 1)

- httpsd.conf ファイルの指定

  IndexOptions +IconsAreLinks FancyIndexing +SuppressLastModified

  この場合，IconsAreLinks は無効になります。

(例 2)

- <VirtualHost>ブロック，<Directory>ブロックまたはアクセスコントロールファイルの指定

  IndexOptions +IconsAreLinks FancyIndexing +SuppressLastModified

この場合，IconsAreLinks，SuppressLastModified は無効になります。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d) 上書き許可

Indexes レベル

(50) IndexOrderDefault {Ascending | Descending} {Name | Date | Size | Description}

(a) 内容

ディレクトリインデクス表示での，ファイルの並び順のデフォルトを指定します。

Ascending：昇順

Descending：降順

Name：ファイル名で並べます。

Date：ファイル更新日付で並べます。

Size：ファイルサイズで並べます。

Description：AddDescription ディレクティブで指定した説明文で並べます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

6.2.5 K，L で始まるディレクティブ

(1) KeepAlive {On | Off}

(a) 内容

KeepAlive 接続を有効にするかどうかを指定します。実際に KeepAlive が実行されるのはクライアント側も KeepAlive に対応している場合だけです。KeepAlive はサーバプロセスとクライアントとのコネクションが持続されるので，連続したリクエストのレスポンスが良くなります。反面，サーバプロセスが特定のクライアント専用になるので，Web サーバ全体としてサービス能力が低下することもあります。KeepAliveTimeout，MaxKeepAliveRequests ディレクティブを使用して調整する必要があります。なお，この指定は，リバースプロキシ機能使用時のバックエンドサーバとの接続には影響しません。

On：持続型接続（KeepAlive）を有効にします。

Off：持続型接続（KeepAlive）を無効にします。

(b) 注意事項

HTTP/2 通信の場合，KeepAlive ディレクティブに Off を指定しても，コネクションが持続します。

コネクションが持続する時間は，KeepAliveTimeout ディレクティブ値に従います。

(c) 記述できる場所

httpsd.conf

(d) 指定例

```
KeepAlive On
```

(2) KeepAliveTimeout 時間

～((0－65535))《5》（単位：秒）

(a) 内容

KeepAlive 接続時の要求待ち時間を秒単位で指定します。この時間以上経過しても，クライアントから次のリクエストが来ない場合，コネクションを切断します。KeepAlive はサーバプロセスが特定のクライアントに占有されます。ある Web ページから次の Web ページへ移る場合に必要とする標準的な時間以上は，タイムアウトにしてコネクションを切断し，サーバプロセスをほかのリクエストの処理に当てるようにします。時間に 0 を指定した場合は，KeepAlive 接続が無効になります。なお，この指定は，リバースプロキシ機能使用時のバックエンドサーバとの接続には影響しません。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
KeepAliveTimeout 15
```

KeepAlive 接続時の要求待ち時間は 15 秒

(3) LanguagePriority 言語コード 〔言語コード …〕

(a) 内容

使用言語を優先順位の高い順に指定します。コンテントネゴシエーションで，Web ブラウザからのリクエストに言語コードの優先順位（Accept-Language ヘッダ）が含まれていない場合に，ここで指定した優先順位が使用されます。ここで指定する言語コードなどについては，AddLanguage ディレクティブを参照してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
LanguagePriority ja en fr de
```

優先順位は日本語，英語，フランス語，ドイツ語の順

(4) LimitRequestBody リクエストボディサイズ

～((0－2147483647))《0》（単位：バイト）

(a) 内容

HTTP 通信によって，Web ブラウザが送信してくるリクエストをサーバが受信する場合のオブジェクトボディ（デ ータ）のサイズの上限を指定します。Web ブラウザから<FORM METHOD=POST ACTION=...>によるリクエストを送る場合などにオブジェクトボディが用いられます。上限値を設定しない場合は，0 を指定してください。

リクエストデータ内のリクエストボディサイズが上限値を超えた場合，クライアントにステータスコード 413 を返します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(5) LimitRequestFields ヘッダ数

～((0－32767))《100》

(a) 内容

HTTP 通信によって，Web ブラウザが送信してくるリクエストをサーバが受信する場合の HTTP ヘッダ数の上限を指定します。リクエストの HTTP ヘッダ数は，Web ブラウザやリクエストを中継するプロキシなどの仕様で変わります。上限値を設定しない場合は，0 を指定してください。

このディレクティブ指定値を，バーチャルホストに対するリクエストにも適用するためには，<VirtualHost>ブロックよりも前に指定する必要があります。

リクエストデータ内の HTTP ヘッダ数が上限値を越えた場合，クライアントにステータスコード 400 を返します。

(b) 注意事項

HTTP/2 通信の場合，リクエストに含まれる:method，:scheme，:authority，:path の疑似ヘッダを除いたヘッダ数と比較します。また，同じヘッダが複数ある場合は一つのヘッダとして計算します。

(c) 記述できる場所

httpsd.conf

(6) LimitRequestFieldsize　ヘッダサイズ

～((0－8190))《8190》（単位：バイト）

(a) 内容

HTTP 通信によって，Web ブラウザが送信してくるリクエストをサーバが受信する場合，一つの HTTP ヘッダの，サイズの上限を指定します。リクエストヘッダのサイズは Web ブラウザやリクエストを中継するプロキシなどの仕様で変わります。

このディレクティブ指定値を，バーチャルホストに対するリクエストにも適用するためには，<VirtualHost> ブロックよりも前に指定する必要があります。

リクエストデータ内の一つの HTTP ヘッダのサイズが上限値を越えた場合，クライアントにステータスコード 400 を返します。

(b) 注意事項

HTTP/2 通信の場合，同じヘッダが存在すると，ヘッダ値の間に“, ”を挿入して一つにマージされます（ただし，Cookie ヘッダフィールドの場合は“; ”を挿入）。マージされた場合はマージ後のサイズと比較します。

(c) 記述できる場所

httpsd.conf

(7) LimitRequestLine　リクエストライン長

～((0－2147483647))《8190》（単位：バイト）

(a) 内容

HTTP 通信によって，Web ブラウザが送信してくるリクエストをサーバが受信する場合のリクエストライン（メソッド，問い合わせ文字列などを含む URI，HTTP バージョン）の長さの上限を指定します。Web ブラウザから<FORM METHOD=GET ACTION...>によるリクエストを送る場合などに問い合わせ文字列としてリクエストラインが用いられます。なお，リクエストラインとして Web ブラウザから何バイト送れるかは，Web ブラウザやリクエストを中継するプロキシなどの仕様で変わります。

このディレクティブ指定値を，バーチャルホストに対するリクエストにも適用するためには，<VirtualHost>ブロックよりも前に指定する必要があります。

リクエストデータ内のリクエストライン長が上限値を越えた場合，クライアントにステータスコード 414 を返すか，または HTTP/2 通信を終了します。

(b) 注意事項

リクエストライン長に大きな値を指定した場合，リクエスト処理中に大量のメモリ領域を使用するおそれがあるため，このディレクティブに必要以上に大きな値を指定しないでください。

HTTP/2 通信の場合，:method, :scheme, :authority, :path の疑似ヘッダフィールドの値の長さと比較します。:method, :authority の疑似ヘッダフィールドの値の長さが上限値を超えた場合は，HTTP/2 通信を終了します。

(c) 記述できる場所

httpsd.conf

(8) Listen 〔IP アドレス:〕ポート番号

(a) 内容

リクエストを受け付ける IP アドレスおよびポート番号を指定します。Port ディレクティブと異なり，複数指定できます。Listen ディレクティブを指定すると，Port ディレクティブおよび BindAddress ディレクティブの指定は無視されます。

IP アドレスには IPv6 アドレスも指定できます。IPv6 アドレスを指定する場合は，IPv6 アドレスを[ ]で囲んでください。ただし，IP アドレスを省略してポート番号だけを指定した場合は，IPv4 アドレスを使用したリクエストだけを受け付けます。このため，IPv6 アドレスを使用する場合は，必ず Listen ディレクティブに IPv6 アドレスを指定してください。

Listen ディレクティブの IP アドレスを変更してサーバを再起動する場合，サーバをいったん停止後，起動してください。コマンドなどで再起動を選択すると，サーバの起動に失敗する場合があります。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
Listen 80
Listen [2001::123:4567:89ab:cdef]:8080
Listen [::]:80
```

(9) ListenBacklog バックログ数

～((1－2147483647))《511》

(a) 内容

クライアントからの接続要求の最大の待ち行列数を指定します。この指定値はシステムコール listen( )のバックログ数として設定されます。ただし，指定値の制限値や，実際の待ち行列数の最大値については OS によって異なるため，詳細は各 OS の listen( )についてのマニュアルや，各 OS の TCP/IP 実装の詳細を説明しているドキュメントを参照してください。

(b) 記述できる場所

httpsd.conf

(10) LoadFile ファイル名 〔ファイル名 …〕

(a) 内容

DSO によって組み込むモジュールが参照するコードがあるオブジェクトファイルまたはライブラリを指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

LoadModule ディレクティブでこのファイルを参照するモジュールを指定する場合，それらが httpsd.conf で使用される前に，このディレクティブを指定する必要があります。

(b) 記述できる場所

httpsd.conf

(11) LoadModule module 構造体名 ライブラリファイル名

(a) 内容

Web サーバに動的に組み込むモジュールを指定します。ライブラリファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

UNIX 版の場合，prefork MPM モジュールまたは worker MPM モジュールのどちらか一つを組み込む必要があります。どちらの指定もない場合は prefork MPM モジュールを組み込みます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

prefork MPM モジュールを組み込む場合

```
LoadModule mpm_prefork_module libexec/mod_mpm_prefork.so
```

worker MPM モジュールを組み込む場合

```
LoadModule mpm_worker_module libexec/mod_mpm_worker.so
```

(12) LogFormat "フォーマット" 〔ラベル名〕

～<<"%h %l %u %t ¥"%r¥" %>s %b">>

(a) 内容

ログのフォーマットにラベル名を定義します。ここで定義したラベル名を CustomLog ディレクティブで指定できます。指定できるフォーマットは CustomLog ディレクティブを参照してください。なお，フォーマットに%A または%a を指定した場合，IPv6 アドレスも出力できます。また，フォーマットに%h または%V を指定した場合，IPv6 アドレスに対応したホスト名または IPv6 アドレスも出力できます。

ラベル名を付けない場合は，このディレクティブを複数指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" combined
LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

(13) LogLevel {debug | info | notice | warn | error | crit | alert | emerg}

(a) 内容

エラーログに出力するエラーのレベルを指定します。指定したレベルの上位レベルのログを出力します。ただし，notice レベルのログはこの指定に関係なく出力されます。また，HTTP Server 起動時など，レベル指定の解析終了前に出力されるメッセージは，この指定に関係なく出力される場合があります。

次にエラーレベルを上位順に示します。

レベル	意味
emerg	緊急メッセージ

レベル	意味
alert	即時処理要求メッセージ
crit	致命的な状態のメッセージ
error	一般的エラーメッセージ
warn	警告レベルメッセージ
notice	標準的だが重要なメッセージ
info	インフォメーションメッセージ，外部モジュールと CGI プログラム実行時のモジュールトレース※
debug	デバッグレベルメッセージ，内部モジュールトレースおよび info 相当のモジュールトレース※

注※　モジュールトレースは，エラーログではなくリクエストログに出力するよう設定できます。詳細は，「4.2.2(6)　各トレースの出力先」および「4.2.6　モジュールトレースの採取」を参照してください。

(b)　記述できる場所

httpsd.conf，<VirtualHost>

(c)　指定例

```
LogLevel info
```

6.2.6　M，N，O，P，Q，R で始まるディレクティブ

(1)　MaxClients　接続数　U

prefork MPM

～((1－1024))《1024》

worker MPM

～((ThreadsPerChild - (ServerLimit × ThreadsPerChild)))《400》

(a)　内容

同時に接続できるクライアントの最大数（TCP/IP のコネクション最大数）を指定します。

HTTP Server を起動すると，StartServers ディレクティブで指定した数のプロセスが起動されリクエストを待ちます。

prefork MPM

多くのリクエストが同時に発生した場合，複数のプロセスでリクエストを処理することになります。リクエスト待ちの残りプロセス数が MinSpareServers ディレクティブで指定した数より少なくなると，徐々に新規プロセスを生成します。このとき，プロセス数がこのディレクティブで指定した数になるま

でプロセスが生成されます。その後，リクエストの処理が終了しリクエスト待ちプロセスが増加すると，MaxSpareServers ディレクティブで指定した数までプロセスを終了させます。

worker MPM

多くのリクエストが同時に発生した場合，複数のスレッドでリクエストを処理することになります。リクエスト待ちの残りスレッド数が MinSpareThreads ディレクティブで指定した数より少なくなると，新しいプロセスを生成します。このとき，スレッド数がこのディレクティブで指定した数になるまでプロセスが生成されます。その後，リクエストの処理が終了しリクエスト待ちスレッドが増加すると，MaxSpareThreads ディレクティブで指定した数以下になるまでプロセスを終了させます。

MaxClients ディレクティブの値は，ThreadsPerChild ディレクティブの値から，ServerLimit ディレクティブの値×ThreadsPerChild ディレクティブの値の範囲です。MaxClients ディレクティブの指定値を増加させるために，サーバプロセス数の上限値を変更する場合には ServerLimit ディレクティブ指定値を変更します。MaxClients ＜ ServerLimit×ThreadsPerChild の場合，サーバプロセス数の上限が MaxClients / ThreadsPerChild（小数点以下切り捨て）になります。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MaxClients 150
```

(2) MaxKeepAliveRequests 接続数

～((0－2147483647))《100》

(a) 内容

KeepAlive 連続接続回数の上限を指定します。上限値を設定しない場合は 0 を指定します。KeepAlive はサーバプロセスが特定のクライアントに占有されるので，ほかのクライアントにもサービスの機会を与えるために上限を設けます。

(b) 注意事項

HTTP/2 通信の場合，連続接続回数に上限はありません。

(c) 記述できる場所

httpsd.conf

(d) 指定例

```
MaxKeepAliveRequests 100
```

(3) MaxRequestsPerChild リクエスト処理回数 U

～((0－2147483647))《0》

(a) 内容

サーバプロセスのリクエスト処理回数を指定します。サーバプロセスは指定されたリクエスト処理回数だけ動作し，終了します。 ユーザが作成したアプリケーションなどによるメモリリークによる障害を未然に防ぐ効果があります。なお，0 を指定すると，サーバプロセスのリクエスト処理回数の上限は設定されません。サーバプロセスは終了することなく，リクエストを待ち，処理します。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MaxRequestsPerChild 10000
```

(4) MaxSpareServers プロセス数 U

～((1－1024))《10》

(a) 内容

リクエスト待ち状態で稼働させておくサーバプロセスの最大数を指定します。プロセス数に関連するほかのディレクティブについては，「4.1 HTTP Server の処理とディレクティブとの関係」を参照してください。

MinSpareServers 以下の値を設定した場合，MinSpareServers 指定値+1 の値が仮定されます。

このディレクティブは，prefork MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MaxSpareServers 10
```

(5) MaxSpareThreads スレッド数 U

～(((MinSpareThreads+ThreadsPerChild)－MaxClients))《250》

(a) 内容

リクエスト待ち状態で稼働させておくサーバスレッドの最大数を指定します。リクエスト待ち状態のサーバスレッド数がこの指定値より多くなった場合，リクエスト待ち状態のサーバスレッド数がこの指定値以下になるまでサーバプロセスを終了させます。

MaxSpareThreads ディレクティブの値は，MinSpareThreads ディレクティブの値 + ThreadsPerChild ディレクティブの値 から，MaxClients ディレクティブの値の範囲です。MinSpareThreads ディレクティブの値 + ThreadsPerChild ディレクティブの値より小さい値を指定した場合は，MinSpareThreads ディレクティブの値 + ThreadsPerChild ディレクティブの値が仮定されます。

このディレクティブは，worker MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MaxSpareThreads 75
```

(6) MinSpareServers プロセス数 U

～((1－1024))《5》

(a) 内容

リクエスト待ち状態で稼働しているサーバプロセスの最小数を指定します。サーバプロセス数がこの指定値より少なくなったら，新しいプロセスを生成します。プロセス数に関連するほかのディレクティブについては，「4.1 HTTP Server の処理とディレクティブとの関係」を参照してください。

このディレクティブは，prefork MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MinSpareServers 5
```

(7) MinSpareThreads スレッド数 U

～((1 - MaxClients))《75》

(a) 内容

リクエスト待ち状態で稼働しているサーバスレッドの最小数を指定します。リクエスト待ち状態のサーバスレッド数がこの指定値より少なくなった場合，新しいサーバプロセスを生成します。

このディレクティブは，worker MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
MinSpareThreads 25
```

(8) MultiviewsMatch {NegotiatedOnly | Handlers}

(a) 内容

コンテントネゴシエーションの対象となる拡張子の種類を指定します。

NegotiatedOnly：拡張子が文字セット，圧縮形式，言語コード，MIME タイプと関連づけられたものだけをコンテントネゴシエーションの対象にします。

Handlers：NegotiatedOnly を指定した場合の対象に加え，ハンドラと関連づけられた拡張子についてもコンテントネゴシエーションの対象にします。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
MultiviewsMatch Handlers
```

(9) Options {+ | -} オプション 〔{+ | -} オプション …〕

～《None》

(a) 内容

ユーザが利用できる機能を制限する場合に指定します。

+：オプションで指定した機能の利用を許可します。

-：オプションで指定した機能の利用を禁止します。

オプション	機能
All	MultiViews，SymLinksIfOwnerMatch を除くすべてのオプションが有効です。
ExecCGI	CGI スクリプトの実行を許可します。
FollowSymLinks	シンボリックリンクをたどります。Windows 版では指定できません。
Indexes	URL にディレクトリが指定されたとき，DirectoryIndex ディレクティブで指定したファイル（デフォルトは index.html）がない場合，ディレクトリのインデクスを表示します。
MultiViews	Content-negotiated Multiviews をサポートします。
None	すべてのオプションで指定できる機能を無効にします。
SymLinksIfOwnerMatch	ファイルまたはディレクトリの所有者がシンボリックリンクの所有者と同じ場合だけ，リンクをたどります。Windows 版では指定できません。

注意事項

+-を使用しないでこのディレクティブを複数指定すると，最後に指定したディレクティブだけが有効になります。

(例 1)

```
Options All
Options ExecCGI
```

このようにオプションに+-を指定しないディレクティブを 2 行指定した場合，ユーザは CGI スクリプトの実行機能だけが利用できます。ディレクトリインデクスなどの機能は利用できません。

(例 2)

httpsd.conf ファイルの指定

```
Options All
```

アクセスコントロールファイルの指定

```
Options ExecCGI
```

httpsd.conf ファイルのあとにアクセスコントロールファイルが参照されるので，アクセスコントロールファイルがあるディレクトリでは CGI スクリプトの実行機能だけが利用できます。

(例 3)

```
Options Indexes ExecCGI
```

このように 1 行に+-を指定しないオプションを指定した場合は，指定した機能の両方を利用できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Options レベル

(10) Order 指示子

～《deny,allow》

(a) 内容

Allow ディレクティブと Deny ディレクティブの指定の評価の順序を指定します。指示子に指定できるものを次に示します。先に評価されたものは，あとに評価されるものに上書きされます。

指示子	意味
deny,allow	Deny ディレクティブの指定を，Allow ディレクティブの指定より先に評価
allow,deny	Allow ディレクティブの指定を，Deny ディレクティブの指定より先に評価
mutual-failure	Allow ディレクティブに指定され，Deny ディレクティブに指定されていないホストだけアクセスを許可

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

Limit レベル

(11) PassEnv 環境変数 〔環境変数 …〕

(a) 内容

CGI スクリプトに渡す任意の環境変数を指定できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
PassEnv TMP
```

(12) PidFile ファイル名

～《logs/httpd.pid》

(a) 内容

制御プロセス ID を格納するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

(b) 注意事項

Windows 版の場合，再起動時には，PidFile ディレクティブ指定値の変更は反映されません。PidFile ディレクティブ指定値を変更した場合は，いったん Web サーバを停止してから，再起動してください。

UNIX 版の場合，再起動時には，PidFile ディレクティブ指定値の変更は反映されません。PidFile ディレクティブ指定値を変更した場合は，いったん Web サーバを kill コマンドで停止してから，起動してください。停止時に httpsdctl コマンドは使用できません。

複数環境を生成する場合には，ファイルパスはほかの環境と競合しないように変更してください。

(c) 記述できる場所

httpsd.conf

(d) 指定例

```
PidFile logs/httpd.pid
```

(13) Port ポート番号

～((1－65535))《80》

(a) 内容

IPv4 アドレスを使用した Web ブラウザからの要求を受け付けるサーバのポート番号を指定します。

Port ディレクティブを指定しても，IPv6 アドレスを使用した Web ブラウザからの要求は受け付けません。IPv6 アドレスを使用する場合は，Listen ディレクティブで指定してください。その場合，IPv4 アドレスと併用するときは，IPv4 アドレスについても Listen ディレクティブを指定してください。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
Port 80
```

(14) Protocols プロトコル名 〔プロトコル名 …〕

～《http/1.1》

(a) 内容

サーバで使用可能なプロトコルのリストを指定します。複数のプロトコルが指定された場合には，先に指定されたプロトコルを優先します。サーバが認識できないプロトコルが指定された場合は無視します。
<VirtualHost>で指定した場合，httpsd.conf での指定はマージしないで上書きします。

h2：TLS 暗号を利用した HTTP/2 プロトコルでの通信をする場合に指定します。

h2c：平文の HTTP/2 プロトコルでの通信をする場合に指定します。

http/1.1：HTTP/1.1 プロトコルでの通信をする場合に指定します。HTTP/1.0 プロトコルも含まれます。

(b) 注意事項

http/1.1 を含まない h2 および h2c の指定でも HTTP/2 プロトコルが選択できなかった場合は，
HTTP/1.1 プロトコルが選択されます。

HTTP/2 プロトコル通信機能を使用するためには mod_http2 モジュールの組み込みが必要です。
HTTP/2 プロトコル通信については，「4.17 HTTP/2 プロトコル通信機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

(例 1)

```
Protocols h2 h2c
```

h2 および h2c プロトコルが使用できます。

(例 2)

```
Protocols h2
Protocols h2c
```

h2 および h2c プロトコルが使用できます。

(15) Proxy100Continue {On | Off}

(a) 内容

クライアントから Expect: 100-continue ヘッダを含むリクエストを受けた場合の動作を設定します。

On：クライアントから Expect: 100-continue ヘッダを含むリクエストを受けた場合，バックエンドサーバにリクエストを転送し，バックエンドサーバからのレスポンスを待つ動作となります。HTTP Server 11-20-09 以前のバージョンでのデフォルト値です。

Off：クライアントから Expect: 100-continue ヘッダを含むリクエストを受けた場合，リバースプロキシがクライアントに 100 Continue のレスポンスを返し，クライアントからのリクエストボディを待つ動作となります。HTTP Server 11-20-10 以降のバージョンでのデフォルト値です。

(b) 注意事項

バックエンドサーバに Cosminexus の NIO HTTP サーバを使用する場合は，このディレクティブを必ず Off に設定してください。

(c) 記述できる場所

httpsd.conf

(16) ProxyErrorOverride {On〔ステータスコード …〕 | Off}

(a) 内容

バックエンドサーバからのステータスコードが 400 番台または 500 番台の場合，レスポンスヘッダとレスポンスボディをオーバーライドします。その結果，リバースプロキシはバックエンドサーバからのレスポンスではなく，自身が生成したレスポンスをクライアントに返します。

On を指定したあとにステータスコードを指定すると，オーバーライドの対象のステータスコードを限定できます。

On：バックエンドサーバからのステータスコードが 400 番台または 500 番台の場合，レスポンスヘッダとレスポンスボディをオーバーライドします。

ステータスコードを指定しない場合，400 番台および 500 番台のステータスコードすべてがオーバーライドの対象となります。

ステータスコードを指定する場合，指定したステータスコードだけオーバーライドの対象となります。また，ステータスコードを指定したディレクティブを複数指定した場合には，指定されたすべてのステータスコードがオーバーライドの対象となります。

Off：レスポンスヘッダとレスポンスボディをオーバーライドしません。

(b) 注意事項

- リバースプロキシを使用するためには mod_proxy モジュールおよび mod_proxy_http モジュールの組み込みが必要です。リバースプロキシの詳細は，「4.7 リバースプロキシの設定」を参照してください。
- HTTP/2 通信ではレスポンスヘッダとレスポンスボディをオーバーライドしません。

- ステータスコードには，ErrorDocument に指定できるステータスコードだけを指定してください。それ以外のステータスコードを指定した場合，オーバーライドする対象のステータスコードがないため，内部エラーが発生したことを示すメッセージを返すことがあります。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(d) 指定例

(例 1)

```
ProxyErrorOverride On
```

バックエンドサーバからのステータスコードが 400 番台または 500 番台の場合，リバースプロキシが生成したレスポンスをクライアントに返します。

(例 2)

```
ProxyErrorOverride On 403 405 500 501 502 503 504
```

バックエンドサーバからのステータスコードが 403，405，500，501，502，503，504 の場合，リバースプロキシが生成したレスポンスをクライアントに返します。

(17) ProxyPass パス名 {URL | !} 〔キー=値 〔キー=値 …〕〕

(a) 内容

リバースプロキシを使用する場合，Web ブラウザからのリクエストとそれを転送するアドレスを指定します。

パス名：Web ブラウザからリバースプロキシへのリクエストをスラッシュ（/）から始まる URL で指定します。

URL：転送先となるバックエンドサーバの URL を"http，ws://IP アドレスまたはホスト名〔:ポート番号〕/"を含む形で指定します。

URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。

パス名および URL には，?以降（問い合わせ文字列），*および正規表現は使用できません。

http://から始まる URL を指定した場合は，バックエンドサーバと HTTP 通信します。

h2c://から始まる URL を指定した場合は，バックエンドサーバと平文の HTTP/2 プロトコルで通信します。

ws://から始まるURLを指定した場合は，クライアントとバックエンドサーバ間の通信をWebSocket通信にアップグレードします。ただし，クライアントおよびバックエンドサーバがWebSocketプロトコルに対応している必要があります。

!：指定したパス名に一致したリクエストを，リバースプロキシが扱うリクエストの対象から除外する場合に指定します。

キー：指定できるキーを次に示します。

キー	値	内容
timeout	1－65535（単位：秒）	次のバックエンドサーバとの送受信時の待ち時間を指定します。 • バックエンドサーバへのリクエスト送信中にデータを送信できなくなった場合の待ち時間 • バックエンドサーバへのリクエスト送信後からレスポンス受信までの待ち時間 • バックエンドサーバからのレスポンス受信中にデータを受信しなくなった場合の待ち時間 キーが省略された場合のデフォルト値は，Timeoutディレクティブ指定値です。
connectiontimeout	1－65535（単位：秒）	バックエンドサーバとの接続時の待ち時間を指定します。 キーが省略された場合のデフォルト値は，timeoutキー値です。

URLにws://から始まるURLを指定した場合でもtimeoutキーを指定できますが，WebSocket通信へのアップグレード後ではバックエンドサーバとの送受信時の待ち時間としては機能しません。connectiontimeoutキーが省略された場合のデフォルト値としてだけ機能します。

次のディレクティブ指定値と重複するパス名は指定できません。

- Aliasの URL
- AliasMatchの正規表現
- Redirectの旧パス
- RedirectMatchの正規表現
- ScriptAliasの URL
- ScriptAliasMatchの正規表現
- リダイレクタ定義ファイルのJkMountの URLパターン

(b) 注意事項

リバースプロキシを使用するためには，次のモジュールの組み込みが必要です。

- mod_proxyモジュール

  UNIX版

  ```
  LoadModule proxy_module libexec/mod_proxy.so
  ```

Windows 版

LoadModule proxy_module modules/mod_proxy.so

さらに，転送先となるバックエンドサーバの URL の指定によって，それぞれ次のモジュールの組み込みが必要です。

URL が http://で始まる指定の場合

- mod_proxy_http モジュール

  UNIX 版

  LoadModule proxy_http_module libexec/mod_proxy_http.so

  Windows 版

  LoadModule proxy_http_module modules/mod_proxy_http.so

URL が h2c://で始まる指定の場合

- mod_proxy_http2 モジュール

  UNIX 版

  LoadModule proxy_http2_module libexec/mod_proxy_http2.so

  Windows 版

  LoadModule proxy_http2_module modules/mod_proxy_http2.so

URL が ws://で始まる指定の場合

- mod_proxy_wstunnel モジュール

  UNIX 版

  LoadModule proxy_wstunnel_module libexec/mod_proxy_wstunnel.so

  Windows 版

  LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so

!を指定する場合は，URL を指定する ProxyPass ディレクティブより先に指定してください。パス名および URL には，?以降（問い合わせ文字列）を指定できません。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

```
ProxyPass /abc/def/ !
ProxyPass /abc/ http://backend.example.com/
```

/abc/def/から始まるリクエストをリバースプロキシのリクエストとして扱いません。

(18) ProxyPassReverse パス名 URL

(a) 内容

リバースプロキシを使用する場合，バックエンドサーバからのリダイレクトレスポンスの Location ヘッダで示す URL を変更します。Web ブラウザからのリダイレクトによるリクエストをリバースプロキシを通すリクエストにするために Location ヘッダをこのディレクティブの指定値に変更します。

パス名：リダイレクトのリクエスト先であるリバースプロキシのパス名を，/（スラッシュ）から始まる URL で指定します。

URL：変更対象となる Location ヘッダ中のバックエンドサーバの URL を"http://ホスト名〔:ポート番号〕"を含む形で指定します。

URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。IPv6 アドレスにはさまざまな表記方法がありますので，指定値に注意してください。IPv6 アドレスの表記が指定値と一致しない場合，ディレクティブが有効になりません。IPv6 アドレスを指定する場合は，バックエンドサーバからの応答の Location ヘッダ値に含まれる IPv6 アドレスの表記を確認してください。パス名および URL には，?以降（問い合わせ文字列），*および正規表現は使用できません。

(b) 注意事項

リバースプロキシを使用するためには mod_proxy モジュールおよび mod_proxy_http モジュールの組み込みが必要です。Location ヘッダの変更で使用されるホスト名やポート番号は，UseCanonicalName ディレクティブ指定値によって決定します。パス名および URL には，?以降（問い合わせ文字列），*および正規表現は使用できません。リバースプロキシの詳細は，「4.7 リバースプロキシの設定」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(19) ProxyPreserveHost {On | Off}

(a) 内容

リバースプロキシを使用する場合，クライアントから受信した Host ヘッダの値をそのままバックエンドサーバに転送するかどうかを指定します。

On：クライアントから受信した Host ヘッダの値をそのままバックエンドサーバに転送します。

Off：クライアントから受信した Host ヘッダの値を ProxyPass ディレクティブの指定値に従って変更して，バックエンドサーバに転送します。

(b)　注意事項

リバースプロキシを使用するためには mod_proxy モジュールおよび mod_proxy_http モジュールの組み込みが必要です。リバースプロキシの詳細は，「4.7　リバースプロキシの設定」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(d)　指定例

```
ProxyPreserveHost On
```

クライアントから受信した Host ヘッダの値をそのままバックエンドサーバに転送します。

(20)　ProxyVia　{on | off | full | block}

(a)　内容

このディレクティブはプロキシで Via ヘッダの使用を制御する場合に指定します。

on：Via ヘッダに自ホストの情報を追加します。すでにある情報は変更しません。

off：Via ヘッダに自ホストの情報を追加しません。すでにある情報は変更しません。

full：コメントとして自ホストのバージョンを付けた情報を Via ヘッダに追加します。すでにある情報は変更しません。

block：Via ヘッダに自ホストの情報を追加しません。リクエスト中の Via ヘッダは削除します。

(b)　注意事項

リバースプロキシを使用するためには mod_proxy モジュールおよび mod_proxy_http モジュールの組み込みが必要です。リバースプロキシの詳細は，「4.7　リバースプロキシの設定」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>

(21)　QOSCookieDomain　ドメイン名

(a)　内容

流量制限機能に使用するクッキーが有効とされるドメインを指定します。この値は，HWS 作成モードで使用され，ユーザ作成モードでは使用されません。複数のホストを設定している場合，このディレクティブを設定することでドメイン部分の共通するホスト間でクッキーを使用できるようになります。ドメイン名には，少なくとも"."が二つ含まれていなければなりません。

なお，IPv6 アドレスに対応したドメイン名も指定できます。

(例)

a.example.com と b.example.com の二つのホストを設定している場合，このディレクティブで .example.com と指定すると，二つのホストのどちらにアクセスしても優先度処理が行われます。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(22) QOSCookieExpires 値

～((0－86400))《300》（単位：秒）

(a) 内容

流量制限機能に使用するクッキーの有効時間を秒単位で指定します。この値は，HWS 作成モードで使用され，ユーザ作成モードでは使用されません。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(23) QOSCookieName クッキー名 〔{hws | user}〕

～《HWSCHK》

(a) 内容

流量制限機能に使用するクッキー名を指定します。クッキー名にセミコロン，コンマ，空白文字は使用できません。ホスト間および URL 間でそれぞれ異なるクッキーを利用したセション管理を行う場合は，別のクッキー名を指定する必要があります。

hws：HTTP Server が作成するクッキーを用いて，セション管理を実施します。これを HWS 作成モードと呼びます。

user：HTTP Server 以外の外部モジュールなどで作成されたクッキーを用いて，セション管理を実施します。これをユーザ作成モードと呼びます。

(b) 注意事項

- 流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。
- QOSCookieName ディレクティブを特定のブロックに指定した場合，上位に指定されている QOSCookieName ディレクティブは継承しません。

  (例)

  ```
  QOSCookieName Cookie1 hws
  <Location /loc1>
      QOSCookieName Cookie2 user
  </Location>
  ```

  この場合，"/loc1"から始まるリクエストでは，クッキー名 Cookie2 の指定が有効になります。"/loc1"以外から始まるリクエストでは，クッキー名 Cookie1 の指定が有効になります。
- QOSCookieName ディレクティブを複数指定する場合は，クッキー名を重複させないでください。重複している場合は，起動エラーになります。

  (例)

  ```
  QOSCookieName Cookie1 hws
  QOSCookieName Cookie1 user
  ```

  この場合，クッキー名が重複しているため起動エラーになります。
- HWS 作成モードの QOSCookieName ディレクティブを複数指定した場合は，後ろに指定した方が有効になります。

  (例)

  ```
  QOSCookieName Cookie1 hws
  QOSCookieName Cookie2 hws
  ```

  この場合，クッキー名 Cookie1 の指定は無効になり，Cookie2 の指定が有効になります。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(24) QOSCookieSecure {on | off}

(a) 内容

クライアントに対し，SSL によるアクセス時だけにクッキーを送信させるよう設定します。この値は，HWS 作成モードで使用され，ユーザ作成モードでは使用されません。クッキーの確認は SSL の暗号処理の終了後であることに注意してください。

on：SSL によるアクセス時だけ，クライアントにクッキーを送信させるよう設定します。

off：SSL 以外によるアクセス時にも，クライアントにクッキーを送信させるよう設定します。

(例)

SSL が有効であるホストと無効であるホストを設定している場合，このディレクティブを設定すると，SSL が有効なホストへのアクセスだけクッキーが送信されます。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(25) QOSCookieServers 値

UNIX 版の場合

～((0－MaxClients ディレクティブ指定値))《10》

Windows 版の場合

～((0－ThreadsPerChild ディレクティブ指定値))《10》

(a) 内容

リクエスト待ち状態のサーバプロセス数が減少した場合に，クッキーを送信してきたリクエストだけを処理するときの，サーバプロセス数を指定します。

Windows 版の場合は，サーバスレッド数を指定します。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(26) QOSRedirect 旧パス 新パス

(a) 内容

流量制限機能によって処理が拒否された場合に，クライアントからのリクエストを指定されたパスにリダイレクトさせるときに指定します。新パスには，"プロトコル名://ホスト名［:ポート番号］"を含む URL のパスを指定します。また，新パスに指定する URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。

旧パスでリクエストを受けた場合，ステータスコード 302 と Location ヘッダに新パスを設定したレスポンスを返します。レスポンスをカスタマイズすることはできません。

旧パス，新パスの指定については，Redirect ディレクティブを参照してください。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(27) QOSRejectionServers 値

UNIX 版の場合

～((0－MaxClients ディレクティブ指定値))《1》

Windows 版の場合

～((0－ThreadsPerChild ディレクティブ指定値))《1》

(a) 内容

リクエスト待ち状態のサーバプロセス数が減少し，受信したすべてのリクエストを拒否するようになるときの，サーバプロセス数を指定します。

Windows 版の場合は，サーバスレッド数を指定します。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(28) QOSResponse {file［MIME タイプ］ファイル名｜message テキスト}

(a) 内容

流量制限機能によって処理が拒否された場合に，503 ステータスコードとともに返送するコンテンツを指定します。コンテンツはサーバプロセス内にキャッシュされるため，変更する場合にはサーバの再起動が必要です。

file：指定したファイルを，指定した MIME タイプで返送します。MIME タイプを省略したときは"text/html"が設定されます。また，ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

message：指定したテキストを返送します。テキストは先頭に"を記述して文字列を指定します。MIME タイプには"text/html"が設定されます。

(b) 注意事項

流量制限機能を使用するためには mod_hws_qos モジュールの組み込みが必要です。流量制限機能については，「4.9 流量制限機能」を参照してください。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Location>

(d) 指定例

```
QOSResponse file ”text/html; charset=ISO-8859-1” htdocs/busy.html
QOSResponse message ”Server busy.
```

(29) ReadmeName ファイル名

(a) 内容

ディレクトリインデクス表示時の Readme として付けるコメントを記述したファイルのファイル名（パス情報なし）を指定します。HTML またはプレーンテキストで記述できます。ただし，AddType ディレクティブまたは TypesConfig ディレクティブで指定したファイルで，MIME タイプが正しく定義されている必要があります。プレーンテキストでコメントを作成した場合，ディレクトリインデクス表示の HTML には<PRE>タグが追加されます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

Indexes レベル

(d) 指定例

```
ReadmeName README.html
```

ディレクトリ下の README.html ファイルの内容を表示します。

(30) Redirect 〔{permanent | temp | seeother | gone | ステータスコード}〕 旧パス 新パス

(a) 内容

旧パスに対するクライアントからのリクエストを，新パスに再リクエスト（リダイレクト）する場合に指定します。

旧パスには，スラッシュから始まるリクエスト URL のパスを指定します。ただし，旧パスには，?以降（問い合わせ文字列）を指定できません。

次のディレクティブ指定値と重複する旧パスは指定できません。

- Alias の URL
- AliasMatch の正規表現
- ProxyPass のパス名
- RedirectMatch の正規表現
- ScriptAlias の URL
- ScriptAliasMatch の正規表現
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
Redirect temp /aaa/bbb/ http://aaa.example.com/
ProxyPass /aaa/ http://aaa.example.com/
```

新パスには，"プロトコル名://ホスト名〔:ポート番号〕"を含む URL のパスを指定します。また，新パスに指定する URL には，IPv6 アドレスまたは IPv6 アドレスに対応したホスト名も指定できます。

旧パスでリクエストを受けた場合，指定したステータスコードと Location ヘッダに新パスを設定した応答を返します。通常，300 番台のステータスコードを受けた Web ブラウザは，自動的に Location ヘッダに指定されたアドレスに対してリダイレクトします。

Redirect ディレクティブでは，特定のファイルへのリクエストを特定のファイルへリダイレクトするか，特定のディレクトリ下の，任意のパスへのリクエストを特定のディレクトリ下の，同名パスへリダイレクトする指定ができます。特定のディレクトリ下の，任意のパスへのリクエストを，特定のファイルへリダイレクトしたい場合は RedirectMatch ディレクティブを使用してください。

permanent：ステータスコード 301 Moved Permanently を応答します。

temp：ステータスコード 302 Found を応答します。

seeother：ステータスコード 303 See Other を応答します。

gone：ステータスコード 410 Gone を応答します。新パスは指定できません。

ステータスコード：指定したステータスコードを応答します。指定できる値は，「付録 A　ステータスコード」を参照してください。ただし，300 番台以外を指定する場合，新パスは指定できません。

(b)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c)　上書き許可

FileInfo レベル

(d)　指定例

```
Redirect temp /index.html http://ホスト名:ポート番号/default.html
```

/index.html に対するリクエストを，ステータスコード 302 で"http://ホスト名:ポート番号/default.html"にリダイレクトします。

(31)　RedirectMatch 〔{permanent | temp | seeother | gone | ステータスコード}〕 正規表現　新パス

(a)　内容

正規表現で記述した条件を満たすパスに対するクライアントからのリクエストを，新パスに再リクエスト（リダイレクト）する場合に指定します。

正規表現には，スラッシュから始まるリクエスト URL の旧パスを正規表現で指定します。ただし，旧パスには，?以降（問い合わせ文字列）を指定できません。

次のディレクティブ指定値と重複する正規表現は指定できません。

- Alias の URL
- AliasMatch の正規表現
- ProxyPass のパス名
- Redirect の旧パス
- ScriptAlias の URL
- ScriptAliasMatch の正規表現
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
RedirectMatch ^/aaa/bbb/(.*) http://aaa.example.com/$1
ProxyPass /aaa/ http://aaa.example.com/
```

新パスには，"プロトコル名://ホスト名〔:ポート番号〕/"を含むURLのパスを指定します。また，新パスに指定するURLには，IPv6アドレスまたはIPv6アドレスに対応したホスト名も指定できます。

正規表現で括弧（）を使用してグループ化している場合，そのi番目のグループの表現にマッチした文字列を，新パスで$iを使用して参照できます。iには1から9までの数字を指定します。正規表現で記述した条件を満たすパスへのリクエストを受信した場合に，指定したステータスコードと，新パスを設定したLocationヘッダを応答します。通常，300番台のステータスコードを受けたWebブラウザは，自動的にLocationヘッダに指定されたアドレスに対して再リクエスト（リダイレクト）します。

各ステータスコードの指定については，Redirectディレクティブを参照してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfoレベル

(d) 指定例

(例1)

```
RedirectMatch ^/other/ http://www.example.com/
```

/other/で始まるすべてのリクエストを，ステータスコード302で"http://www.example.com/"にリダイレクトします。

(例2)

```
RedirectMatch permanent ^/old/(.*) http://www.example.com/new/$1
```

"/old/ファイル名"に対するリクエストを，ステータスコード301で"http://www.example.com/new/ファイル名"にリダイレクトします。

(32) RequestHeader {set | append | add} ヘッダ ヘッダ値 | unset ヘッダ} 〔env=〔!〕環境変数〕

(a) 内容

クライアントから受信したヘッダ値をカスタマイズする場合に指定します。

set：ヘッダを設定します。ヘッダがある場合は，指定したヘッダ値に書き換えます。

append：存在するヘッダにヘッダ値を追加します。存在するヘッダ値との間は，コンマで区切られます。ヘッダがない場合は，ヘッダを設定します。

add：ヘッダがあっても，別の行にヘッダを設定します。同じヘッダを複数行設定する場合に使用します。

unset：指定したヘッダがある場合，そのヘッダをすべて削除します。

env=環境変数：指定した環境変数が設定されている場合に，RequestHeader ディレクティブで指定した内容を実行します。

env=!環境変数：指定した環境変数が設定されていない場合に，RequestHeader ディレクティブで指定した内容を実行します。

ヘッダ値に空白がある場合は，"（引用符）で囲む必要があります。ヘッダ値は文字だけから成る文字列，フォーマット指示子を含む文字列または両方から成る文字列を指定できます。フォーマット指示子を次に示します。

フォーマット指示子	意味
%t	リクエストを受け取った時刻を，1970 年 1 月 1 日 0 時 0 分 0 秒(GMT：Greenwich Mean Time)から経過した時間で表示する。単位はマイクロ秒。先頭には"t="が付けられる。
%D	リクエスト処理に掛かった時間を表示する。単位はマイクロ秒。先頭には"D="が付けられる。
%{env_name}e	環境変数 env_name の値。

(b)　注意事項

ヘッダカスタマイズ機能を使用するためには mod_headers モジュールの組み込みが必要です。ヘッダカスタマイズ機能については，「4.10　ヘッダカスタマイズ機能」を参照してください。

(c)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d)　上書き許可

FileInfo レベル

(e)　指定例

```
RequestHeader set Host www.example.com
```

(33)　RequestReadTimeout　タイプ=時間　〔タイプ=時間〕

～《header=20 body=20》

(a) 内容

リクエスト受信開始からリクエストヘッダ受信完了まで，およびリクエストボディの受信開始から完了までのタイムアウト時間を秒単位で指定します。タイムアウト時間には，0 から 2147483647 までの値を指定します。データの送信が遅いリクエストに対して，サーバのリソースを長時間専有されることを避ける場合に有効です。なお，指定のないタイプについてはタイムアウトに 20 秒を設定して，タイムアウト時間に 0 を指定したタイプについてはタイムアウトを設定しません。

リクエストヘッダ受信時にタイムアウトを検知した場合，ステータスコード 408 を返します。

リクエストボディ受信時にタイムアウトを検知した場合，ステータスコード 400 を返します。

タイプには次に示す header と body が指定できます。

header：リクエスト受信開始からリクエストヘッダ受信完了までの経過時間を監視する場合に指定します。

body：リクエストボディの受信開始から受信完了までの経過時間を監視する場合に指定します。

(b) 注意事項

このディレクティブを使用するためには，mod_reqtimeout モジュールの組み込みが必要です。

UNIX 版の場合

LoadModule reqtimeout_module libexec/mod_reqtimeout.so

Windows 版の場合

LoadModule reqtimeout_module modules/mod_reqtimeout.so

また，モジュールトレースの出力の抑止を同時に設定することを推奨します。

HWSSuppressModuleTrace mod_reqtimeout.c

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

(例 1)

```
RequestReadTimeout header=10 body=30
```

リクエスト受信開始からリクエストヘッダ受信完了までのタイムアウト時間を 10 秒，リクエストボディの受信開始から完了までのタイムアウト時間を 30 秒に設定します。

(例 2)

```
RequestReadTimeout body=0
```

リクエスト受信開始からリクエストヘッダ受信完了までのタイムアウト時間を 20 秒に設定します。リクエストボディの受信開始から完了までのタイムアウトは設定しません。

(34) Require {user ユーザ名 〔ユーザ名 …〕 | group グループ名 〔グループ名 …〕 | valid-user | file-owner | file-group}

(a) 内容

AuthName ディレクティブ，AuthType ディレクティブ，AuthUserFile ディレクティブ（または AuthGroupFile ディレクティブ）と一緒に指定し，アクセス制限を定義します。

user：AuthUserFile ディレクティブで指定したパスワードファイルに登録されているユーザのうち，ユーザ名で指定したユーザだけアクセスできます。

group：AuthGroupFile ディレクティブで指定したグループファイルに登録されているグループ名で指定したグループに属するユーザだけがアクセスできます。

valid-user：AuthUserFile ディレクティブで指定したパスワードファイルに登録されているすべてのユーザがアクセスできます。

file-owner：AuthUserFile ディレクティブで指定したパスワードファイルに登録されているユーザのうち，アクセス対象ファイルのシステムの所有ユーザと一致しているユーザだけがアクセスできます（Windows 版では指定できません）。

file-group：AuthGroupFile ディレクティブで指定したグループファイルに登録されているグループ名で指定したグループに属するユーザのうち，グループ名がアクセス対象ファイルのシステムの所有グループに一致しているユーザだけがアクセスできます（Windows 版では指定できません）。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

6.2.7　S で始まるディレクティブ

(1)　Satisfy　{any | all}

(a)　内容

コンテンツへのアクセスが，ユーザ認証（AuthUserFile，Require ディレクティブなどを指定）とホスト名または IP アドレス（Allow from，Deny from ディレクティブなどを指定）の両方によって制限されている場合にその関係を設定します。

any：そのどちらかの条件を満たしていれば，コンテンツへのアクセスを許可します。

all：そのどちらの条件も満たさなければ，コンテンツへのアクセスを禁止します。

(b)　記述できる場所

<Directory>，.htaccess

(2)　Script　メソッド　CGI スクリプト名

(a)　内容

指定されたメソッドによるリクエストがあった場合に CGI スクリプト名で示すスクリプトを実行します。

指定できるメソッド：GET，POST，PUT，DELETE

メソッドは大文字，小文字を区別します。

ただし，GET メソッドの場合，スクリプトは問い合わせ引数があるときだけ（例えば，/foo.html?bar）呼ばれます。

(b)　記述できる場所

httpsd.conf，<VirtualHost>，<Directory>

(c)　指定例

```
Script POST /cgi-bin/search
```

(3)　ScriptAlias　URL　ディレクトリ名

(a)　内容

Web ブラウザから URL で指定された CGI プログラム実行のリクエストに対して，実行する CGI プログラムのあるディレクトリ名を指定します。

次のディレクティブ指定値と重複する URL は指定できません。

- Alias の URL
- AliasMatch の正規表現
- ProxyPass のパス名
- Redirect の旧パス
- RedirectMatch の正規表現
- ScriptAliasMatch の正規表現
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
ScriptAlias /aaa/bbb/ C:/alias/
ProxyPass /aaa/ http://aaa.example.com/
```

ディレクトリ名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
ScriptAlias /cgi-bin/ "<Application Serverのインストールディレクトリ>/httpsd/cgi-bin/"
```

(4) ScriptAliasMatch 正規表現 新パス

(a) 内容

Web ブラウザから指定された CGI プログラム実行要求の URL が正規表現で記述した条件を満たす場合，指定した新パスの CGI プログラムを実行します。正規表現で括弧（）を使用してグループ化している場合，その i 番目のグループの表現にマッチした文字列を，新パスで $i を使用して参照できます。i には 1 から 9 までの数字を指定します。

新パスは，絶対パスで指定してください。また，新パスの文字として，'$'または'&'を含める場合は，その文字の前に'¥'を付加してください。なお，$i を指定する際には，'$'の前に'¥'を付加する必要はありません。

次のディレクティブ指定値と重複する正規表現は指定できません。

- Alias の URL
- AliasMatch の正規表現
- ProxyPass のパス名
- Redirect の旧パス
- RedirectMatch の正規表現

- ScriptAlias の URL
- リダイレクタ定義ファイルの JkMount の URL パターン

例えば，次のような指定はできません。

```
ScriptAliasMatch ^/aaa/bbb/(.*) C:/alias/$1
ProxyPass /aaa/ http://aaa.example.com/
```

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
ScriptAliasMatch ^/cgi-bin/(.*) "<Application Serverのインストールディレクトリ>/httpsd/cgi-bin/$1"
```

(5) ScriptInterpreterSource { registry | script } W

(a) 内容

CGI スクリプトの実行に使用されるインタプリタを定義します。

registry：レジストリが検索され，拡張子に関連づけられているプログラムがインタプリタとして使用されます。

script：スクリプト内の#!行で指定されたインタプリタが使用されます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(6) ScriptLog ファイル名

(a) 内容

CGI スクリプトのエラーログ出力先のファイルを指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

UNIX の場合，指定するファイルは，User ディレクティブで指定したユーザの権限で書き込みができるようになっている必要があります。

(b) 記述できる場所

httpsd.conf

(7) ScriptLogBuffer バッファ数

～((0－2147483647))《1024》（単位：バイト）

(a) 内容

PUT，POST メソッドによるリクエストのボディ部のログを採取する場合の最大値をバイト単位で指定します。ScriptLog ディレクティブでエラーログ出力先のファイルを指定した場合だけ，この指定は有効になります。

このディレクティブでの指定値分の領域が，リクエスト処理中に確保されます。そのため，大きい値を指定すると，メモリ確保失敗となって，Web サーバが終了する場合があります。デフォルト値または必要最小限の値を指定することを推奨します。

(b) 記述できる場所

httpsd.conf

(8) ScriptLogLength ファイルサイズ

～((0－2147483647))《10385760》（単位：バイト）

(a) 内容

CGI スクリプトのエラーログファイルの最大サイズをバイト単位で指定します。ScriptLog ディレクティブでエラーログ出力先のファイルを指定した場合だけ指定が有効になります。

(b) 記述できる場所

httpsd.conf

(9) SendBufferSize 送信バッファサイズ

～((512－2147483647))《0》（単位：バイト）

(a) 内容

Web サーバの TCP 送信バッファサイズをバイト数で指定します。0 を指定した場合，OS のデフォルト値が使用されます。

高速のネットワーク環境では，OS のデフォルト値よりも大きな値を設定することで，レスポンス送信の性能が向上する場合があります。

(b)　記述できる場所

httpsd.conf

(c)　指定例

```
SendBufferSize 131072
```

(10)　ServerAdmin　E-Mail アドレス

(a)　内容

サーバ管理者の E-Mail アドレスを指定します。ServerSignature ディレクティブで E-Mail を指定する場合は，必ず指定してください。

(b)　記述できる場所

httpsd.conf，<VirtualHost>

(c)　指定例

```
ServerAdmin www-admin@server.example.com
```

(11)　ServerAlias　ホスト名　〔ホスト名　…〕

(a)　内容

サーバ名に基づくバーチャルホストで使用するホスト名（ServerName）の別名を指定します。IPv6 アドレスに対応したホスト名も指定できます。

(b)　記述できる場所

<VirtualHost>

(12)　ServerLimit　プロセス数　U

～((1-1000)))《16》

(a)　内容

MaxClients ディレクティブと ThreadsPerChild ディレクティブによって生成可能なサーバプロセス数を求めますが，そのサーバプロセス数を制限するために上限値を指定します。このディレクティブの指定値に従い，サーバ稼働情報を保持する共有メモリ領域を確保します。そのため，実際に動作するサーバプロセス以上の値を指定した場合，使用しない余分な共有メモリ領域を確保することになります。また，Web サーバが起動できないことや不安定になることがあるため，このディレクティブには MaxClients ディレクティブの値を ThreadsPerChild ディレクティブの値で割った値と同じ値を指定することを推奨します。

このディレクティブは再起動で指定値を変更できません。指定値を変更する場合は，Web サーバを停止させたあとに起動させてください。

このディレクティブは，worker MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
ServerLimit 64
```

(13) ServerName サーバ名〔:ポート番号〕

(a) 内容

HTTP Server のサーバ名およびポート番号を指定します。ポート番号を省略した場合は，Port ディレクティブ指定値が設定されます。

サーバ名は，FQDN（完全修飾ドメイン名）または IP アドレスで指定します。また，サーバ名には，IPv6 アドレスまたは IPv6 アドレスに対応した FQDN も指定できます。IPv6 アドレスを指定し，かつポート番号を指定する場合は，IPv6 アドレスを[ ]で囲んでください。

UseCanonicalName ディレクティブ指定値に従い，イメージマップの利用または末尾を/（スラッシュ）で閉じないディレクトリ指定のリクエストなど，Web サーバでリダイレクトが指示された場合のリダイレクト先として Location ヘッダに設定されクライアントに返信されるため，クライアントからアクセスできるサーバ名を指定しなければなりません。このディレクティブの指定は必須です。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
ServerName www.example.com
ServerName 2001::123:4567:89ab:cdef
ServerName [2001::123:4567:89ab:cdef]
ServerName [2001::123:4567:89ab:cdef]:8080
```

(14) ServerPath パス名

(a) 内容

サーバ名に基づくバーチャルホストで，Host ヘッダの代わりにパス名を利用して各ホストに接続する場合に指定します。

(b) 記述できる場所

<VirtualHost>

(15) ServerRoot ディレクトリ名

～《/opt/hitachi/httpsd》 (UNIX 版)

～《<Application Server のインストールディレクトリ>¥httpsd》 (Windows 版)

(a) 内容

HTTP Server のルートディレクトリを絶対パスで指定します。ここに指定した絶対パスはほかのディレクティブから，相対パス指定時の基点になります。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
ServerRoot "C:/Program Files/Hitachi/Cosminexus/httpsd"
```

(16) ServerSignature {On | Off | Email}

(a) 内容

Web サーバが作成するエラーメッセージなどのコンテンツのフッタに署名するかどうかを指定します。

On：ServerTokens ディレクティブに従った文字列（「Cosminexus HTTP Server」やバージョン番号など）および UseCanonicalName ディレクティブ指定値に従ったサーバ名とポート番号を表示します。

```
Cosminexus HTTP Server 09-00 at www.example.com Port 80
```

Off：コンテンツのフッタに署名を表示しません。

Email：On を指定した場合の表示に加え ServerAdmin ディレクティブの指定値を mailto タグで追加します。

なお，On を指定した場合，ServerName ディレクティブに指定した IPv6 アドレスまたは IPv6 アドレスに対応したホスト名を表示できます。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 指定例

```
ServerSignature On
```

(17) ServerTokens {Minimal | OS | Full | ProductOnly}

(a) 内容

HTTP レスポンスヘッダの Server ヘッダのフォーマットを設定します。それぞれの設定による Server ヘッダの値を次に示します。OS 種別には，Unix，Win32 または Win64 が設定されます。Server ヘッダの値がどのように利用されるかはクライアントの仕様によります。

Minimal：Cosminexus HTTP Server バージョン番号

OS：Cosminexus HTTP Server バージョン番号(OS 種別)

Full：Cosminexus HTTP Server バージョン番号(OS 種別) 付加 PP で設定された情報

ProductOnly：Cosminexus HTTP Server

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
ServerTokens Full
```

(18) SetEnv 環境変数 値

(a) 内容

CGI スクリプトに任意の環境変数を渡す場合に設定する環境変数の値を指定します。このディレクティブを複数指定する場合，同じ環境変数に異なる値は指定できません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
SetEnv MY_ENV myenv
```

(19) SetEnvIf リクエスト値 正規表現 環境変数〔=値〕〔環境変数〔=値〕 …〕

(a) 内容

クライアントからのリクエストを基に環境変数を定義します。クライアントからのリクエスト値が正規表現で表した条件を満たす場合，指定した環境変数を設定します。設定する値のデフォルト値は 1 です。環境変数の前に!が付いたときは，その環境変数の設定を解除します。正規表現で括弧（）を使用してグループ化している場合，そのi番目のグループの表現にマッチした文字列を，値で$i を使用して参照できます。また，$0 を使用することで正規表現にマッチした文字列全体を指定することが可能です。i には 0 から 9 までの数字を指定します。

リクエスト値としては，HTTP リクエストヘッダか次の表に示す値を指定できます。先に指定された環境変数をリクエスト値として指定することで環境変数の検査ができます。ただし，この場合の環境変数は，HTTP リクエストヘッダにも次の表に示す指定値にも一致していない必要があります。

リクエスト値	意味
Remote_Addr	クライアントの IP アドレス
Remote_Host	クライアントのホスト名（リクエストに設定されている場合だけ）
Request_Protocol	リクエストのプロトコル（HTTP/1.1　など）
Request_Method	リクエストのメソッド名（GET，POST，HEAD など）
Request_URI	リクエストの URI
Server_Addr	リクエストを受信したサーバの IP アドレス

なお，リクエスト値に Remote_Host を指定した場合，正規表現には IPv6 アドレスに対応したホスト名も指定できます。また，IPv6 を使用した接続に対しては，Remote_Addr と Server_Addr のリクエスト値は使用できません。Remote_Addr と Server_Addr を使用したい場合は，HWSSetEnvIfIPv6 ディレクティブで設定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

(例 1)

```
SetEnvIf User-Agent "Mozilla.*" SETENVIF_USER_AGENT=Mozilla
```

(例 2)

```
SetEnvIf Request_URI "¥.(gif|jpg)$" request_is_image
```

(例 3)

IPv4 を使用した接続のうち，特定のクライアントに対して環境変数を設定する場合は，次のように指定します。

```
Listen 123.123.123.123:80
Listen [2001::123:4567:89ab:cdef]:80
<VirtualHost 123.123.123.123:80>
        SetEnvIf Remote_Addr ^234¥.234¥.234¥.234$ IPV4_CLIENT
</VirtualHost>
```

(例 4)

Origin ヘッダの値が http://または https://から始まり，test1.com または test2.com を含む場合，環境変数 ORIGIN に正規表現にマッチした文字列全体を設定します。Origin ヘッダの値が http://xxx.xxx.test1.com の場合，環境変数 ORIGIN に http://xxx.xxx.test1.com が設定されます。

```
SetEnvIf Origin "^https?://(.*.test1.com|.*.test2.com)" ORIGIN=$0
```

(20) SetEnvIfNoCase リクエスト値 正規表現 環境変数〔=値〕〔環境変数〔=値〕 …〕

(a) 内容

クライアントからのリクエストを基に環境変数を定義します。クライアントからのリクエスト値が正規表現で表した条件を満たす場合，指定した環境変数を設定します。設定する値のデフォルト値は 1 です。環境変数の前に!が付いたときは，その環境変数の設定を解除します。

リクエスト値に指定できる値については，SetEnvIf ディレクティブを参照してください。

ただし，このディレクティブでは，正規表現の大文字，小文字の区別をしません。

なお，リクエスト値に Remote_Host を指定した場合，正規表現には IPv6 アドレスに対応したホスト名も指定できます。また，IPv6 を使用した接続に対しては，Remote_Addr と Server_Addr のリクエスト値は使用できません。Remote_Addr と Server_Addr を使用したい場合は，HWSSetEnvIfIPv6 ディレクティブで設定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(21) SetHandler ハンドラ名

(a) 内容

指定した<Directory>またはアクセスコントロールファイルの範囲すべてのリクエストをハンドラ名で指定したハンドラに関連づける場合，指定します。ハンドラ名として none を指定すると，それまでの SetHandler ディレクティブの設定が無効になります。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(22) SSLBanCipher 暗号種別 〔暗号種別 …〕

(a) 内容

指定した暗号種別でのアクセスを拒否し，クライアントにステータスコード 403 Forbidden を応答します。

指定できる暗号種別は，SSLCipherSuite ディレクティブを参照してください。

TLSv1.2 以前と TLSv1.3 プロトコルの暗号種別を同時に指定できます。

このディレクティブを指定しない場合は，暗号種別でのアクセス拒否をしません。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(23) SSLCACertificateFile ファイル名

(a) 内容

SSL 機能を使用しクライアント認証する場合，CA（認証局）の証明書（PEM 形式）のファイル名を指定します。複数の証明書ファイルを連結させて，一つのファイルに複数の証明書が混在できます。

ファイル名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
SSLCACertificateFile ”<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/cacert/anycert.pem”
```

(24) SSLCACertificatePath ディレクトリ U

(a) 内容

SSL 機能を使用しクライアント認証する場合，CA の証明書（PEM 形式）へのハッシュリンクを格納したディレクトリを指定します。ハッシュリンクの作成および運用方法については「5.2.8 ハッシュリンクの作成（UNIX 版）（openssl.sh x509 コマンド）」を参照してください。

ディレクトリ名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
SSLCACertificatePath /opt/hitachi/httpsd/conf/ssl/cacerts
```

(25) SSLCARevocationCheck {none | leaf}

(a) 内容

証明書失効リスト（CRL）チェックをするかどうかを指定します。CRL のチェックをする場合は，SSLCARevocationFile ディレクティブを指定する必要があります。

none：CRL のチェックをしません。

leaf：クライアント証明書について，CRL のチェックをします。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(26) SSLCARevocationFile ファイル名

(a) 内容

PEM 形式の CRL を優先順に連結したファイルを絶対パスで指定します。これはクライアント認証時に CRL を適用したい場合に指定します。ただし，SSLCARevocationCheck に none を指定した場合は適用されません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
SSLCARevocationFile ”<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/crl/crl.pem”
```

PEM 形式の CRL ファイルを指定します。

(27) SSLCertificateFile ファイル名

(a) 内容

Web サーバの証明書を格納したファイル名を指定します。サーバ認証するために Web サーバの証明書を先頭として，中間 CA（認証局），ルート CA の順に CA の証明書を連結することができます。証明書は PEM 形式です。

RSA 暗号の証明書と楕円曲線暗号の証明書を指定する場合，別のファイルに分けて一つずつ指定できます。

ファイル名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
# RSA
SSLCertificateFile ”<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server/httpsd-rsa.pem”
# ECC
SSLCertificateFile ”<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server/httpsd-ecc.pem”
```

(28) SSLCertificateKeyFile ファイル名

(a) 内容

Web サーバの秘密鍵のファイル名を指定します。秘密鍵のファイルは PEM 形式です。RSA 暗号の秘密鍵と楕円曲線暗号の秘密鍵を指定する場合，別のファイルに分けて一つずつ指定できます。

ファイル名は，絶対パスで指定してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
# RSA
SSLCertificateKeyFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server
/httpsdkey-rsa.pem"
# ECC
SSLCertificateKeyFile "<Application Serverのインストールディレクトリ>/httpsd/conf/ssl/server
/httpsdkey-ecc.pem"
```

(29) SSLCipherSuite 〔TLSv1.3〕 暗号種別〔:暗号種別…〕

(a) 内容

SSL 機能で使用可能とする暗号種別を指定します。ディレクティブに指定した暗号種別とクライアントが使用できる暗号種別との間で一致するものがあれば，SSL 通信が確立され HTTP リクエストを受信します。一致するものがない場合は，SSL 通信は確立されないで HTTP リクエストを受信しません。

TLSv1.3 プロトコルの暗号種別を指定する場合，暗号種別の前に TLSv1.3 を指定してください。TLSv1.3 を指定しない場合は，TLSv1.2 以前のプロトコルについての暗号種別の指定となります。このディレクティブを指定しない場合は，指定可能なすべての暗号種別が使用可能となります。

TLSv1.2 以前のプロトコルで指定できる暗号種別を次に示します。

表 6-7 TLSv1.2 以前のプロトコルで指定できる暗号種別

暗号種別	鍵交換方式	認証方式	対称鍵暗号方式	暗号鍵サイズ（bit）	メッセージ認証アルゴリズム
AES128-SHA	RSA	RSA	AES	128	SHA
AES128-SHA256	RSA	RSA	AES	128	SHA256
AES256-SHA	RSA	RSA	AES	256	SHA
AES256-SHA256	RSA	RSA	AES	256	SHA256
AES128-GCM-SHA256	RSA	RSA	AES	128	AEAD※
AES256-GCM-SHA384	RSA	RSA	AES	256	AEAD※
ECDHE-RSA-AES128-SHA	ECDH	RSA	AES	128	SHA
ECDHE-RSA-AES256-SHA	ECDH	RSA	AES	256	SHA
ECDHE-RSA-AES128-SHA256	ECDH	RSA	AES	128	SHA256
ECDHE-RSA-AES256-SHA384	ECDH	RSA	AES	256	SHA384
ECDHE-RSA-AES128-GCM-SHA256	ECDH	RSA	AES	128	AEAD※
ECDHE-RSA-AES256-GCM-SHA384	ECDH	RSA	AES	256	AEAD※
ECDHE-ECDSA-AES128-SHA	ECDH	ECDSA	AES	128	SHA

暗号種別	鍵交換方式	認証方式	対称鍵暗号方式	暗号鍵サイズ（bit）	メッセージ認証アルゴリズム
ECDHE-ECDSA-AES256-SHA	ECDH	ECDSA	AES	256	SHA
ECDHE-ECDSA-AES128-SHA256	ECDH	ECDSA	AES	128	SHA256
ECDHE-ECDSA-AES256-SHA384	ECDH	ECDSA	AES	256	SHA384
ECDHE-ECDSA-AES128-GCM-SHA256	ECDH	ECDSA	AES	128	AEAD※
ECDHE-ECDSA-AES256-GCM-SHA384	ECDH	ECDSA	AES	256	AEAD※

注※

AEAD: Authenticated Encryption with Associated Data(認証付き暗号)

TLSv1.3 プロトコルで指定できる暗号種別を次に示します。

表 6-8　TLSv1.3 プロトコルで指定できる暗号種別

暗号種別	鍵交換方式	認証方式	対称鍵暗号方式	暗号鍵サイズ(bit)	メッセージ認証アルゴリズム
TLS_AES_128_GCM_SHA256	any	any	AES	128	AEAD※
TLS_AES_256_GCM_SHA384	any	any	AES	256	AEAD※
TLS_CHACHA20_POLY1305_SHA256	any	any	CHACHA20/POLY1305	256	AEAD※
TLS_AES_128_CCM_SHA256	any	any	AES	128	AEAD※
TLS_AES_128_CCM_8_SHA256	any	any	AES	128	AEAD※

注※

AEAD: Authenticated Encryption with Associated Data(認証付き暗号)

(b)　注意事項

SSLCipherSuite ディレクティブに指定できる暗号種別以外を含んでいる場合，TLS プロトコルのバージョンおよび HTTP Server のバージョンによって起動時の動作が異なります。

- TLSv1.3 プロトコル（HTTP Server 11-20-09 以前）の場合は起動エラーになります。
- TLSv1.3 プロトコル（HTTP Server 11-20-10 以降）の場合は起動エラーになりません。
- TLSv1.2 以前のプロトコルの場合は起動エラーになりません。

HTTP/2 プロトコル機能で TLSv1.2 プロトコルを使用する場合は，次の暗号種別だけが使用できます。

- ECDHE-RSA-AES128-GCM-SHA256

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES256-GCM-SHA384

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

（例 1）

```
SSLCipherSuite AES128-SHA256:AES256-SHA256
```

TLSv1.2 以前のプロトコルで使用する暗号種別を指定します。

（例 2）

```
SSLCipherSuite TLSv1.3 TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY130
5_SHA256
```

TLSv1.3 プロトコルで使用する暗号種別を指定します。

(30) SSLEngine {On | Off}

(a) 内容

SSL を有効にするかどうかを指定します。デフォルトは SSL 無効です。

<VirtualHost>ブロック内で SSL を有効にしたい場合，有効にしたい<VirtualHost>ブロック内で SSLEngine On を指定してください。

On：SSL を有効にします。

Off：SSL を無効にします。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(31) SSLOptions 〔+ | -〕オプション 〔〔+ | -〕オプション …〕

(a) 内容

SSL 機能を使用する場合のオプションを指定します。

+：オプションで指定した機能を有効にします。

-：オプションで指定した機能を無効にします。

オプション	機能
StdEnvVars	CGI 環境変数に SSL 関連の標準の環境変数を設定します。環境変数の設定は性能に影響を与えるため，CGI で SSL 関連の環境変数が必要な場合のみ，このオプションを有効にしてください。
ExportCertData	CGI 環境変数に SSL_SERVER_CERT，SSL_CLIENT_CERT および SSL_CLIENT_CERT_CHAIN_n（n = 0，1，2，...）を設定します。CGI でサーバおよびクライアント証明書の情報が必要な場合に指定します。
FakeBasicAuth	SSL クライアント認証の機能と併せて，Web ブラウザでユーザ ID とパスワードを入力することなく，クライアント証明書の提示だけで Basic 認証する場合に指定します。AuthUserFile ディレクティブで指定するファイルには X509 クライアント証明書の Subject とパスワードを記述します。パスワードは，常に"{SHA}W6ph5Mm5Pz8GgiULbPgzG37mj9g="固定とします（"password"を暗号化したもの）。 openssl.sh x509 または openssl.bat x509 コマンドで表示する証明書の Subject フィールドの値 Subject: C = JP, ST = Kanagawa, L = Yokohama-shi, O = HITACHI, OU = Software, CN = username, emailAddress = username@userhost この場合 AuthUserFile ディレクティブで指定するファイルは次のように指定します。 /C=JP/ST=Kanagawa/L=Yokohama-shi/O=HITACHI/OU=Software/CN=username/emailAddress=username@userhost:{SHA}W6ph5Mm5Pz8GgiULbPgzG37mj9g=

(b) 注意事項

+-を使用しないでオプションを指定すると，最後に指定したオプションだけが有効になります。

(c) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(d) 上書き許可

Options レベル

(e) 指定例

(例 1)

```
SSLOptions StdEnvVars
SSLOptions FakeBasicAuth
```

このようにオプションに+-を指定しないディレクティブを 2 行指定した場合，あとに指定した FakeBasicAuth だけが有効となります。

(例 2)

```
SSLOptions StdEnvVars
SSLOptions +FakeBasicAuth
```

このように+を指定したオプションを後に指定した場合，StdEnvVars に加えて FakeBasicAuth も有効となります。

（例 3）

```
SSLOptions +StdEnvVars FakeBasicAuth
```

+-のないオプションを後に指定した場合，前に指定したオプションは無効になり，あとに指定した FakeBasicAuth だけが有効となります。

(32) SSLProtocol 〔+｜-〕プロトコル名 〔〔+｜-〕プロトコル名 …〕

～《All》

(a) 内容

使用する SSL プロトコルのバージョンを指定します。

+： プロトコル名で指定したプロトコルを有効にします。

-： プロトコル名で指定したプロトコルを無効にします。

プロトコル名として設定できる値は次のとおりです。

TLSv1: TLS プロトコルバージョン 1.0 を使用する。

TLSv1.1: TLS プロトコルバージョン 1.1 を使用する。

TLSv1.2: TLS プロトコルバージョン 1.2 を使用する。

TLSv1.3: TLS プロトコルバージョン 1.3 を使用する。

All: 上記すべてのプロトコルを使用する。

(b) 注意事項

+-を使用しないでプロトコル名を指定すると，最後に指定したプロトコルだけが有効になります。連続したプロトコルバージョンを指定していない場合，古いプロトコルバージョンは無効になります。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

（例 1）

```
SSLProtocol +TLSv1.2 +TLSv1.3
```

TLSv1.2 と TLSv1.3 が有効となります。

（例 2）

```
SSLProtocol +TLSv1 +TLSv1.2
```

連続したプロトコルを指定していない場合，後継のプロトコル TLSv1.2 だけが有効となります。

(例 3)

```
SSLProtocol All -TLSv1
```

TLSv1.1，TLSv1.2，TLSv1.3 が有効となります。

(33) SSLRequireSSL

(a) 内容

SSL 以外によるアクセスを禁止する場合に指定します。このディレクティブが指定されている場合，http によるアクセスがステータスコード 403 Forbidden で拒否されます。異なるディレクティブの記述場所で，不用意に SSL を無効にしコンテンツを公開してしまうことを防止します。

(b) 記述できる場所

<Directory>，.htaccess

(c) 上書き許可

AuthConfig レベル

(d) 指定例

```
<VirtualHost 172.17.40.10:443>
 SSLEngine Off
 ...
  <Directory /secure/dir>
    SSLRequireSSL
    ...
  </Directory>
</VirtualHost>
```

172.17.40.10 ホストの 443 ポートに対する http アクセスは，/secure/dir ディレクトリへのアクセスを除いてできます。/secure/dir ディレクトリへの http アクセスは，ステータスコード 403 Forbidden を応答します。

(34) SSLVerifyClient {none | optional | require}

(a) 内容

クライアント認証時の証明書に関する設定を指定します。

none：証明書の要求をしません。

optional：クライアントは証明書を提示できます。運用テスト用。

require：クライアントは証明書を提示しなければなりません。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
SSLVerifyClient require
```

(35) SSLVerifyDepth 段階数

～((0－10))《1》

(a) 内容

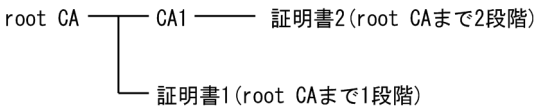
証明書のチェーンを何段階までたどるかを指定します。

クライアント認証に使用する CA 証明書のチェーンについて，認証チェックをする段階数を指定します。チェーンされた CA をどこまで信用するかを制限するために使用します。クライアント証明書が自己署名の証明書の場合に 0 であるため，通常は段階数に 1 以上を指定します。例を次に示します。

(例)

条件

- CA1 は，root CA に署名されている。
- 証明書 1 は，root CA に署名されている。
- 証明書 2 は，CA1 に署名されている。



SSLVerifyDepth の指定

この場合，証明書 1，証明書 2 とも認証チェックをするためには，SSLVerifyDepth ディレクティブに 2 以上を指定します。また，証明書 1 は認証チェックし，証明書 2 は認証チェックをしないようにするには SSLVerifyDepth ディレクティブに 1 を指定します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
SSLVerifyDepth 10
```

(36) StartServers プロセス数 U

worker MPM

～((1－(MaxClients/ThreadsPerChild))) 《3》

prefork MPM

～((1－1024))《5》

(a) 内容

Web サーバ起動時のサーバプロセス数を指定します。

worker MPM

StartServers ディレクティブの指定値は，1 から，MaxClients ディレクティブの値を ThreadsPerChild の値で割った値までの範囲です。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
StartServers 5
```

6.2.8 T，U で始まるディレクティブ

(1) ThreadLimit スレッド数 W

～((1－15000))《1024》

(a) 内容

サーバプロセスに生成するサーバスレッド数の上限値を指定します。このディレクティブの指定値に従い，サーバ稼働情報を保持する共有メモリ領域を確保します。そのため，ThreadsPerChild ディレクティブで指定した実際に動作するサーバスレッド数以上の値を指定した場合，使用しない余分な共有メモリ領域を確保します。このディレクティブには ThreadsPerChild ディレクティブと同じ値を指定することを推奨します。

また，このディレクティブおよび ThreadsPerChild ディレクティブにシステムが扱える以上の値を指定した場合，Web サーバが起動できないことや不安定になることがあるため，必要以上に大きな値を指定しないでください。

このディレクティブは再起動で指定値を変更できません。指定値を変更する場合は，Web サーバを停止させたあとに起動させてください。

(b) 記述できる場所

httpsd.conf

(2) ThreadLimit スレッド数 U

～((1－1000))《64》

(a) 内容

サーバプロセスに生成するサーバスレッド数の上限値を指定します。このディレクティブの指定値に従い，サーバ稼働情報を保持する共有メモリ領域を確保します。そのため，ThreadsPerChild ディレクティブで指定した実際に動作するサーバスレッド数以上の値を指定した場合，使用しない余分な共有メモリ領域を確保することになります。また，Web サーバが起動できないことや不安定になることがあるため，このディレクティブには ThreadsPerChild ディレクティブと同じ値を指定することを推奨します。

このディレクティブは再起動で指定値を変更できません。指定値を変更する場合は，Web サーバを停止させたあとに起動させてください。

このディレクティブは，worker MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(3) Timeout 時間

～((0－65535))《60》（単位：秒）

(a) 内容

次の待ち時間を秒単位で指定します。0 を指定すると即時タイムアウトとなるため，推奨しません。なお，少しのデータでも送受信できた場合，待ち時間はリセットされます。

- クライアントからのリクエスト受信（コネクション確立後，HTTP プロトコルの受信）中にデータを受信しなくなった場合の待ち時間
- クライアントへのレスポンス送信中にデータを送信できなくなった場合の待ち時間
- CGI プログラムへのリクエスト送信中にデータを送信できなくなった場合の待ち時間
- CGI プログラムへのリクエスト送信後からレスポンス受信までの待ち時間
- CGI プログラムからのレスポンス受信中にデータを受信しなくなった場合の待ち時間
- CGI プログラムからのレスポンス受信後，入出力用のパイプを閉じるまでの待ち時間
- リバースプロキシを使用している場合の，バックエンドサーバへのリクエスト送信中にデータを送信できなくなった場合の待ち時間

- リバースプロキシを使用している場合の，バックエンドサーバへのリクエスト送信後からレスポンス受信までの待ち時間
- リバースプロキシを使用している場合の，バックエンドサーバからのレスポンス受信中にデータを受信しなくなった場合の待ち時間

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
Timeout 300
```

(4) ThreadsPerChild スレッド数 W

～((1－ThreadLimit ディレクティブ指定値))《64》

(a) 内容

HTTP Server として起動するスレッド数を指定します。指定したスレッド数は HTTP Server の最大同時接続数（TCP/IP のコネクション最大数）を示します。

(b) 記述できる場所

httpsd.conf

(5) ThreadsPerChild スレッド数 U

～((1－ThreadLimit ディレクティブ指定値))《25》

(a) 内容

一つのサーバプロセスに生成するサーバスレッド数を指定します。

このディレクティブは，worker MPM を使用する場合に指定できます。

(b) 記述できる場所

httpsd.conf

(6) TraceEnable {On | Off | extended}

(a) 内容

TRACE メソッドによるリクエストを拒否するかどうかを指定します。

On：TRACE メソッドによるリクエストを許可します。ただし，リクエストボディが付加されている場合は，413 Request Entity Too Large を応答します。

Off：TRACE メソッドによるリクエストを拒否します。TRACE メソッドによるリクエストの場合は，ステータスコード 403 Forbidden を応答します。

extended：TRACE メソッドによるリクエストを許可します。リクエストボディが付加されていても許可します。ただし，リバースプロキシ以外のリクエストボディサイズの上限は 64KB です。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
TraceEnable Off
```

(7) TransferLog {ファイル名 | パイプ}

(a) 内容

ログを格納するファイルまたはログを出力するプログラムを指定します。ログのフォーマットはラベル名を指定しない LogFormat ディレクティブで指定できます。

LogFormat ディレクティブでログのフォーマットを指定する場合は，IPv6 アドレスや IPv6 アドレスに対応したホスト名も出力できます。指定できるフォーマットは CustomLog ディレクティブを参照してください。

LogFormat ディレクティブでフォーマットを指定しない場合は，標準のログフォーマットで出力します。

ファイル名：ログを格納するファイル名を指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

パイプ：標準入力からログ情報を受け取るプログラムを"|プログラム名"のフォーマットで指定します。Windows 版での注意事項は，CustomLog ディレクティブを参照してください。

(b) 記述できる場所

httpsd.conf，<VirtualHost>

(c) 指定例

```
TransferLog ”|¥”¥”<Application Serverのインストールディレクトリ>/httpsd/sbin/rotatelogs.exe¥” ¥”<Application Serverのインストールディレクトリ>/httpsd/logs/access¥” 86400¥””
```

rotatelogs プログラムを使用してログを 24 時間ごとに分割して採取します。

(8) TypesConfig ファイル名

～《conf/mime.types》

(a) 内容

ファイル拡張子とコンテントタイプ（MIME タイプ）の関係を定義する設定ファイルを指定します。ファイル名には，絶対パスまたは ServerRoot ディレクティブの指定値からの相対パスが指定できます。

設定ファイル内の指定形式は，MIME タイプ ファイル拡張子 〔ファイル拡張子 …〕です。MIME タイプだけ指定している行は無視します。また，行の最初に#を付けると，コメント行になります。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
TypesConfig conf/mime.types
```

MIME タイプの設定ファイルは mime.types

(9) UnsetEnv 環境変数 〔環境変数 …〕

(a) 内容

CGI スクリプトに渡す環境変数から，SetEnv ディレクティブまたは PassEnv ディレクティブで指定した環境変数を削除する場合に指定します。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>，.htaccess

(c) 上書き許可

FileInfo レベル

(d) 指定例

```
UnsetEnv MY_ENV
```

(10) UseCanonicalName {On | Off | dns}

(a) 内容

サーバの正式な名前の生成方法を指定します。サーバの正式な名前は，自サーバを参照する URL や環境変数の SERVER_NAME と SERVER_PORT に設定されます。

On：サーバの正式な名前は，ServerName ディレクティブ指定値から作成され，自サーバを参照する URL や環境変数に設定されます。VirtualHost 指定時に IP アドレスを使用する場合は，VirtualHost ブロック内で ServerName を指定してください。ブロック内で ServerName を指定していない場合は，IP アドレスからホスト名を取得します。

Off：サーバの正式な名前は，Host ヘッダによってクライアントから与えられたホスト名称とポート番号から作成され，自サーバを参照する URL や環境変数に設定されます。ただし，Host ヘッダが与えられない場合は，ServerName ディレクティブ値と，実際のコネクションに使用されているポート番号から作成されます。

dns：Host ヘッダを持たない古いクライアントのためのオプションです。このオプション指定時には，サーバの正式な名前は，クライアントから与えられたサーバの IP アドレスから逆引きしたホスト名称および実際にコネクションに使用されているポート番号から作成され，自サーバを参照する URL や環境変数に設定されます。

なお，On，Off，dns すべての場合で，IPv6 アドレスに対応しています。

(b) 記述できる場所

httpsd.conf，<VirtualHost>，<Directory>

(11) User ユーザ名 U

～《#-1》

(a) 内容

サーバプロセスが動作するときのユーザ名を指定します。

(b) 記述できる場所

httpsd.conf

(c) 指定例

```
User nobody
```

(12) UserDir {ディレクトリ名 | disabled 〔ユーザ名 〔ユーザ名 …〕〕}

(a) 内容

Web ブラウザからの/~ユーザ名/へのリクエストに対して公開するサーバ上の場所をディレクトリ名で指定します。disabled を指定すると，Web コンテンツを公開しないユーザを指定できます。

ディレクトリ名は，相対パスまたは絶対パスで指定します。

Windows 版では，絶対パスだけ有効です。

ディレクトリ名：

- 相対パスで指定した場合

  サーバ上にユーザ ID を持つユーザが，ユーザのホームディレクトリ下の Web コンテンツを公開する場合の場所を指定します。/~ユーザ名/へのリクエストがあった場合，"ユーザのホームディレクトリ/ディレクトリ名" にアクセスします。

- 絶対パスで指定した場合

  ユーザディレクトリの場所を指定します。/~ユーザ名/へのリクエストがあった場合，"ディレクトリ名/ユーザ名" にアクセスします。

disabled：

Web ブラウザからの/~ユーザ名/へのリクエストに対して，Web コンテンツを公開しないユーザを指定します。指定されたユーザ名でのリクエストに対しては，アクセスするディレクトリ名を変換しません。ユーザ名の指定がない場合は，すべてのユーザについて disabled を指定したことになります。

(b) 注意事項

- 複数の UserDir ディレクティブでディレクトリ名を指定した場合，あとから指定したものに上書きされます。
- disabled に指定するユーザ名は，複数の UserDir ディレクティブを用いて指定できます。

(c) 記述できる場所

httpsd.conf，<VirtualHost>

(d) 指定例

(例 1)

```
UserDir public_html
```

ユーザ user1 のホームディレクトリを/home/user1 とすると，リクエスト http://ホスト名〔:ポート番号〕/~user1/index.html で，/home/user1/public_html/index.html にアクセスします。

(例 2)

```
UserDir /home
UserDir disabled user3
UserDir disabled user4 user5
```

リクエスト http://ホスト名〔:ポート番号〕/~user1/index.html で，/home/user1/index.html にアクセスします。ただし，user3 は disabled が指定されているため，http://ホスト名〔:ポート番号〕/~user3/index.html というリクエストで/home/user3/index.html にアクセスできません。user4，user5 についても user3 と同様です。

(例 3)

```
UserDir disabled
```

すべてのユーザに対して disabled を指定します。

付録

付録 A　ステータスコード

HTTP Server が Web ブラウザに返送するステータスコードを次に示します。ステータスコードを Web ブラウザに返送する際には，ステータスコードに応じて自動生成するエラーメッセージを charset=ISO-8859-1 の HTML として同時に返送します。

表 A-1　ステータスコード一覧

ステータスコード	内容
100 Continue	クライアントは，リクエストを継続可能です。
200 OK	正常に終了しました。
204 No Content	リクエストは正常に終了しましたが，返すリソースはありません。 ImapDefault nocontent ディレクティブの指定によって，発生します。
206 Partial Content	部分的なリソースを返します。 クライアントの Range ヘッダを用いた Partial GET リクエストの応答として，部分的なコンテンツを返す場合に発生します。
300 Multiple Choices	複数ページの利用が可能です。
301 Moved Permanently	リソースが恒久的に移動しました。 最後をスラッシュで閉じないディレクトリに対するリクエスト http://**ホスト名**〔:**ポート番号**〕/**ディレクトリ名**や，Redirect permanent ディレクティブの指定によって，発生します。
302 Found	リソースが一時的に移動しました。 Redirect temp ディレクティブの指定によって，発生します。
303 See Other	リソースが移動しました。 Redirect seeother ディレクティブの指定によって，発生します。
304 Not Modified	リクエストしたコンテンツが変更されていません。
307 Temporary Redirect	リソースが一時的に移動しました。
308 Permanent Redirect	リソースが恒久的に移動しました。
400 Bad Request	リクエストにシンタックスエラーがあります。 次のような場合に発生します。 • ヘッダとして誤ったものを指定した場合 • HTTP/1.1 で Host ヘッダがなかった場合 • リクエストヘッダの個数が LimitRequestFields ディレクティブの値を超えた場合 • 一つのリクエストヘッダのサイズが LimitRequestFieldsize ディレクティブの値を超えた場合 • HTTP Server 上に配置された静的コンテンツファイルまたは CGI プログラムへ CONNECT メソッドを使用してリクエストした場合 • リクエストボディの受信処理が，RequestReadTimeout ディレクティブの body キーに指定した時間内に完了しなかった場合

ステータスコード	内容
401 Unauthorized	リソースにアクセスするためには，認証が必要です。 AuthName ディレクティブまたは AuthUserFile ディレクティブなどでアクセスを制御した場合に発生します。
402 Payment Required	将来使用するために予約されているステータスコードです。
403 Forbidden	リソースへのアクセスが禁じられています。 アクセス制御によって，アクセスが拒否された場合または実行権限のない CGI プログラムの実行要求をした場合などに発生します。
404 Not Found	リソースが見つかりません。 サーバ上にはないファイルをリクエストした場合などに発生します。
405 Method Not Allowed	許可されていない HTTP メソッドを使用しました。 HTTP Server 上に配置された静的コンテンツファイルでは，GET，HEAD，POST，OPTIONS，TRACE が使用できます。CGI プログラムでは，CGI プログラムでの実装に依存します。
406 Not Acceptable	クライアントが Accept ヘッダで指定したタイプに応じたレスポンスを返せません。
407 Proxy Authentication Required	最初に，プロキシで，クライアントが自身を認証する必要があります。
408 Request Timeout	リクエストがタイムアウトになりました。 次のような場合に発生します。 • リクエストの読み込み時に，Timeout ディレクティブに指定した時間だけ無通信状態が発生した場合 • リクエストヘッダの読み込みが，RequestReadTimeout ディレクティブの header キーに指定した時間内に完了しなかった場合
409 Conflict	リソースの現在の状態と競合しているため，リクエストを完了できませんでした。
410 Gone	リソースが恒久的に利用できません。 Redirect gone ディレクティブの指定によって，発生します。
411 Length Required	クライアントは Content-Length ヘッダを指定する必要があります。
412 Precondition Failed	クライアントの If-Unmodified-Since ヘッダまたは If-Matched ヘッダなどで指定した条件が一致しません。
413 Request Entity Too Large	リクエストボディサイズが大き過ぎて，サーバで処理できません。 リクエストボディの長さが， LimitRequestBody ディレクティブで指定した長さよりも長い場合に発生します。
414 Request-URI Too Long	HTTP/1.1 通信のリクエスト URI が大き過ぎて，サーバで処理できません。問い合わせ文字列などを含む URI などの長さが，LimitRequestLine ディレクティブの値を超えた場合に発生します。 HTTP/2 通信のリクエストヘッダのヘッダ値が LimitRequestLine ディレクティブの値を超えた場合に発生します。:method, :authority の疑似ヘッダフィールドの場合は，ステータスコードを返送しないで HTTP/2 通信を終了します。
415 Unsupported Media Type	リクエストされたデータのメディア形式をサーバが対応していないため，サーバはリクエストの処理を拒否しています。
416 Requested Range Not Satisfiable	Range ヘッダでの指定範囲は，該当リソースの範囲を超えています。次の条件がすべて成立する場合に出力されます。

ステータスコード	内容
	• リクエストが Range ヘッダフィールドを含む。 • フィールドの範囲指定値が，選ばれたリソースの現在の範囲に重なっていない。 • リクエストに If-Range リクエストヘッダフィールドを含んでいない。
417 Expectation Failed	Expect リクエストヘッダフィールドの拡張が受け入れられませんでした。
422 Unprocessable Entity	リクエストは適正ですが，意味が誤っているために従うことができません。
423 Locked	アクセス中のリソースはロックされています。
424 Failed Dependency	要求されたアクションが別のアクションに依存していて，そのアクションが失敗したため，リソースに対してメソッドを実行できませんでした。
431 Request Header Fields Too Large	リクエストヘッダのサイズや個数が上限を超えています。 次のような場合に発生します。 • HTTP/2 通信のリクエストヘッダの個数が LimitRequestFields ディレクティブの値を超えた場合 • HTTP/2 通信の一つのリクエストヘッダのサイズが LimitRequestFieldsize ディレクティブの値を超えた場合
500 Internal Server Error	Web サーバ上でエラーが発生しました。 CGI プログラムの問題や，アクセス制御ファイル(.htaccess)のエラーなどの場合に発生します。詳細な情報は，エラーログに出力されます。
501 Not Implemented	サポートされていない HTTP メソッドの要求です。
502 Bad Gateway	プロキシサーバが不正な要求を受け取りました。 次のような場合に発生します。 • バックエンドサーバからのレスポンス受信時にステータスラインおよびレスポンスヘッダの読み込みでタイムアウトなどのエラーになった場合 • ProxyPass ディレクティブに指定したホスト名のアドレス解決に失敗した場合
503 Service Unavailable	次のような場合に発生します。 • サーバが過負荷状態で，現在リクエスト処理できない場合 • 処理中のリクエスト数が流量制限機能などで設定した上限値に達した場合 • バックエンドサーバが未起動，または ProxyPass ディレクティブの転送先 URL に不備があり，接続に失敗した場合
504 Gateway Timeout	CGI からのレスポンス受信のタイムアウトなど，リクエストを完了させるために必要な要求がタイムアウトしました。
505 HTTP Version Not Supported	サーバは，要求メッセージで使用された HTTP プロトコルバージョンをサポートしていないか，サポートを拒否しています。
506 Variant Also Negotiates	サーバに内部配置上のエラーがあります。
507 Insufficient Storage	サーバがリクエストを正常に完了するために必要な表現を保存できないため，メソッドをリソースで実行できません。
510 Not Extended	リソースにアクセスするためのポリシーがリクエストで満たされていません。

注

表 A-1 および表 A-1 以外のステータスコードが HTTP Server と連携した CGI プログラムなどの上位プログラムから出力されることがあります。その場合は，それぞれのプログラムのマニュアルを参照してください。

リバースプロキシを使用している場合には，400 Bad Request，403 Forbidden，502 Bad Gateway は，400 Proxy Error，403 Proxy Error，502 Proxy Error となる場合もあります。

付録 B　CGI プログラムに渡す環境変数

Web サーバが CGI プログラムに渡す環境変数の一覧を表 B-1，表 B-2，表 B-5，および表 B-8 に，SSL_SERVER_要素の例，SSL_SERVER_I_要素の例を，表 B-3，表 B-4 に示します。プラットフォーム，クライアントの設定，リクエストの形，Web サーバのディレクティブの設定などによって，ここで記載されている環境変数が設定されない場合や，記載していない環境変数が設定される場合もあります。表の中のサーバ名，ドメイン名，メールアドレスなどはすべて架空の値です。

表 B-1　環境変数一覧

環境変数名	内容	例
AUTH_TYPE	ユーザ認証をする場合の認証タイプ	Basic
COMSPEC	コマンドプロンプトの実行可能ファイル	C:¥WINNT¥system32¥cmd.exe
CONTENT_LENGTH	クライアントからのリクエストが POST の場合の，データのバイト数	20
CONTENT_TYPE	クライアントからのリクエストが POST の場合のコンテンツタイプ	application/x-www-form-urlencoded
DOCUMENT_ROOT	DocumentRoot ディレクティブ指定値	<Application Server のインストールディレクトリ>/httpsd/htdocs
GATEWAY_INTERFACE	CGI バージョン	CGI/1.1
HTTP_ACCEPT	クライアントが示した Accept ヘッダの値	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_ACCEPT_CHARSET	クライアントが示した Accept-Charset ヘッダの値	Shift_JIS,*,utf-8
HTTP_ACCEPT_ENCODING	クライアントが示した Accept-Encoding ヘッダの値	gzip
HTTP_ACCEPT_LANGUAGE	クライアントが示した Accept-Language ヘッダの値	ja,fr, en,it
HTTP_CONNECTION	クライアントが示した Connection ヘッダの値	Keep-Alive
HTTP_HOST	クライアントが示した Host ヘッダの値	www.hws.hitachi.co.jp:8080
HTTP_PRAGMA	クライアントが示した Pragma ヘッダの値	no-cache
HTTP_REFERER	クライアントが示した Referer ヘッダの値	http://www.hws.hitachi.co.jp:8080/test.html
HTTP_USER_AGENT	クライアントが示した User-Agent ヘッダの値	Mozilla/4.73 [ja] (WinNT; U)
PATH	Web サーバ上の PATH 情報	C:¥WINNT¥system32;C:¥WINNT;C:¥WINNT¥System32¥Wbem

環境変数名	内容	例
PATH_INFO	URL のうち CGI スクリプトより後ろの部分	/dir1/file1
PATH_TRANSLATED	ファイルシステムに変換された PATH_INFO の値	<Application Server のインストールディレクトリ>¥httpsd¥htdocs¥dir1¥file1
QUERY_STRING	クライアントから送信された Query String（問い合わせ文字列）	query1=a&query2=b
REMOTE_ADDR	クライアントのアドレス	172.17.xx.xx
REMOTE_HOST	クライアントのホスト名（HostnameLookups が Off 以外でホスト名が解決された場合）	hostxxx
REMOTE_IDENT	クライアントの ID（IdentityCheck ディレクティブ参照）	unknown
REMOTE_PORT	クライアントのポート番号	2298
REMOTE_USER	認証されたリクエストの場合の認証ユーザ名	Userxxx
REQUEST_METHOD	クライアントから送信された HTTP メソッド	GET
REQUEST_URI	クライアントから送信されたリクエスト URI	/cgi-bin/test-cgi?query1=a&query2=b
SCRIPT_FILENAME	リクエストされた CGI スクリプトのファイル名	<Application Server のインストールディレクトリ>/httpsd/cgi-bin/test-cgi
SCRIPT_NAME	リクエストされた CGI スクリプトの URI	/cgi-bin/test-cgi
SERVER_ADDR	Web サーバの IP アドレス	172.17.xx.xx
SERVER_ADMIN	ServerAdmin ディレクティブ指定値	www-admin@server.example.com
SERVER_NAME	Web サーバのホスト名（UseCanonicalName ディレクティブ参照）	www.hws.hitachi.co.jp
SERVER_PORT	Web サーバのポート名（UseCanonicalName ディレクティブ参照）	8080
SERVER_PROTOCOL	クライアントが示した HTTP バージョン	HTTP/1.0
SERVER_SIGNATURE	Web サーバの署名（HTML タグを含む）（ServerSignature ディレクティブ参照）	<ADDRESS>Cosminexus HTTP Server 09-00 at www.example.com Port 8080</ADDRESS>
SERVER_SOFTWARE	Web サーバのプログラム名	Cosminexus HTTP Server 09-00
SYSTEMROOT	システムディレクトリ	C:¥WINNT

環境変数名	内容	例
TZ	Web サーバのタイムゾーン	JST-9
WINDIR	システムディレクトリ	C:¥WINNT

表 B-2　SSL 通信時の環境変数一覧

環境変数名	内容	例
HTTPS	セキュア通信を示します。	on
HTTPS_CIPHER_ALGKEYSIZE※	対称鍵暗号の鍵のビット数	128
HTTPS_CIPHER_USEKEYSIZE※	対称鍵暗号の鍵のビット数のうち，有効なビット数	128
SSL_CIPHER※	SSL 暗号種別（HTTPS_CIPHER と同じ）	AES128-SHA256
SSL_PROTOCOL※	SSL プロトコルバージョン	TLSv1.2
SSL_SERVER_S_DN※	SSL サーバ証明書の subject の Distinguish Name	/C=JP/ST=Kanagawa/L=Yokohama-shi/O=HITACHI/OU=WebSite/CN=www.hws.hitachi.co.jp/EMAIL=www-admin@hws.hitachi.co.jp
SSL_SERVER_S_DN_要素	SSL サーバ証明書の subject の Distinguish Name の各要素	SSL_SERVER_S_DN が上記の例の場合を表 B-3 に示します。
SSL_SERVER_I_DN※	SSL サーバ証明書の issuer の Distinguish Name	/C=JP/ST=Kanagawa/L=Yokohama-shi/O=LOCAL-CA/OU=ca1/CN=ca1.hitachi.co.jp/EMAIL=ca-admin@ca1.hitachi.co.jp
SSL_SERVER_I_要素	SSL サーバ証明書の issuer の Distinguish Name の各要素	SSL_SERVER__I_DN が上記の例の場合を表 B-4 に示します。
SSL_SESSION_ID※	SSL セション ID（16 進数）	F968F8D7075B76587F35931DC594D3E3
SSL_SSLEAY_VERSION	OpenSSL のバージョン	OpenSSL 1.0.2j 26 Sep 2016

注※

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

表 B-3　SSL_SERVER_要素の例

環境変数名	内容	例
SSL_SERVER_CERT※2	SSL サーバ証明書	"-----BEGIN CERTIFICATE-----¥nMIIDrTCCAxagAwIBAgIBAjANBgk...¥n-----END CERTIFICATE-----¥n"

環境変数名	内容	例
SSL_SERVER_S_DN_C※1	SSL サーバ証明書の subject（Web サーバ）の Country Name	JP
SSL_SERVER_S_DN_CN※1	SSL サーバ証明書の subject の Common Name	www.hws.hitachi.co.jp
SSL_SERVER_S_DN_Email※1	SSL サーバ証明書の subject の E-Mail アドレス	www-admin@hws.hitachi.co.jp
SSL_SERVER_S_DN_L※1	SSL サーバ証明書の subject の Locality Name	Yokohama-shi
SSL_SERVER_S_DN_O※1	SSL サーバ証明書の subject の Organization Name	HITACHI,Ltd.
SSL_SERVER_S_DN_OU※1	SSL サーバ証明書の subject の Organization Unit Name	WebSite
SSL_SERVER_S_DN_ST※1	SSL サーバ証明書の subject の State Name	Kanagawa

注※1

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

注※2

コンフィグファイルに SSLOptions +ExportCertData を指定した場合に環境変数が設定されます。

表 B–4　SSL_SERVER_I_要素の例

環境変数名	内容	例
SSL_SERVER_I_DN_C※	SSL サーバ証明書の issuer（発行者）の Country Name	JP
SSL_SERVER_I_DN_CN※	SSL サーバ証明書の issuer の Common Name	ca1.hitachi.co.jp
SSL_SERVER_I_DN_Email※	SSL サーバ証明書の issuer の E-Mail アドレス	ca-admin@ca1.hitachi.co.jp
SSL_SERVER_I_DN_L※	SSL サーバ証明書の issuer の Locality Name	Yokohama-shi
SSL_SERVER_I_DN_O※	SSL サーバ証明書の issuer の Organization Name	LOCAL-CA
SSL_SERVER_I_DN_OU※	SSL サーバ証明書の issuer の Organization Unit Name	ca1
SSL_SERVER_I_DN_ST※	SSL サーバ証明書の issuer の State Name	Kanagawa

注※

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

表 B-5　SSL クライアント認証時の環境変数一覧

環境変数名	内容	例
SSL_CLIENT_CERT※2	SSL サーバ証明書	"-----BEGIN CERTIFICATE-----¥n MIIDrTCCAxagAwIBAgIBAjANBgk... ¥n -----END CERTIFICATE-----¥n"
SSL_CLIENT_CERT_CHAIN_n※2	SSL サーバ証明書	"-----BEGIN CERTIFICATE-----¥n MIIDrTCCAxagAwIBAgIBAjANBgk... ¥n -----END CERTIFICATE-----¥n"
SSL_CLIENT_S_DN※1	SSL クライアント証明書の subject の Distinguish Name	/C=JP/ST=Kanagawa/ L=Yokohama/O=Hitachi/OU=soft/ CN=c_name/ EMAIL=c_name@soft.hitachi.co.jp
SSL_CLIENT_**要素**	SSL クライアント証明書の subject の Distinguish Name の各要素	SSL_CLIENT_S_DN が上記の例の場合を表 B-6 に示します。
SSL_CLIENT_I_DN※1	SSL クライアント証明書の issuer の Distinguish Name	/C=JP/ST=Kanagawa/ L=Yokohama-shi/O=LOCAL-CA/ OU=ca1/CN=ca1.hitachi.co.jp/ EMAIL=ca-admin@ca1.hitachi.co.jp
SSL_CLIENT_I_**要素**	SSL クライアント証明書の issuer の Distinguish Name の各要素	SSL_CLIENT_I_DN が上記の例の場合を表 B-7 に示します。

注※1

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

注※2

コンフィグファイルに SSLOptions +ExportCertData を指定した場合に環境変数が設定されます。

表 B-6　SSL_CLIENT_要素の例

環境変数名	内容	例
SSL_CLIENT_S_DN_C※	SSL クライアント証明書の subject の Country Name	JP
SSL_CLIENT_S_DN_CN※	SSL クライアント証明書の subject の Common Name	c_name
SSL_CLIENT_S_DN_Email※	SSL クライアント証明書の subject の E-Mail アドレス	c_name@soft.hitachi.co.jp
SSL_CLIENT_S_DN_L※	SSL クライアント証明書の subject の Locality Name	Yokohama
SSL_CLIENT_S_DN_O※	SSL クライアント証明書の subject の Organization Name	Hitachi

環境変数名	内容	例
SSL_CLIENT_S_DN_OU※	SSL クライアント証明書の subject の Organization Unit Name	soft
SSL_CLIENT_S_DN_ST※	SSL クライアント証明書の subject の State Name	Kanagawa

注※

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

表 B–7　SSL_CLIENT_I_要素の例

環境変数名	内容	例
SSL_CLIENT_I_DN_C※	SSL クライアント証明書の issuer の Country Name	JP
SSL_CLIENT_I_DN_CN※	SSL クライアント証明書の issuer の Common Name	ca1.hitachi.co.jp
SSL_CLIENT_I_DN_Email※	SSL クライアント証明書の issuer の E-Mail アドレス	ca-admin@ca1.hitachi.co.jp
SSL_CLIENT_I_DN_L※	SSL クライアント証明書の issuer の Locality Name	Yokohama-shi
SSL_CLIENT_I_DN_O※	SSL クライアント証明書の issuer の Organization Name	LOCAL-CA
SSL_CLIENT_I_DN_OU※	SSL クライアント証明書の issuer の Organization Unit Name	ca1
SSL_CLIENT_I_DN_ST※	SSL クライアント証明書の issuer の State Name	Kanagawa

注※

コンフィグファイルに SSLOptions +StdEnvVars を指定した場合に環境変数が設定されます。

表 B–8　HTTP/2 通信時の環境変数一覧

環境変数名	内容	例
HTTP2	HTTP/2 通信	on
H2_STREAM_ID	HTTP/2 ストリーム番号	13
H2_STREAM_TAG	サーバスレッド固有な ID-ストリーム ID	44-13

付録 C　旧バージョンからの移行に関する注意点

HTTP Server を旧バージョンから移行する場合に注意が必要な項目と，設定変更の要否を次の表に示します。

表 C-1　旧バージョンからの移行時に注意が必要な項目

項番	項目	移行前のアプリケーションサーバのバージョン		
		V8	V9	V11
1	プログラムプロダクトの名称変更	○	－	－
2	プログラムメニューの提供終了（Windows 版だけ）	○	－	－
3	SSLv2 プロトコルの非サポート	○	－	－
4	SSLv3 プロトコルの非サポート	○	○	－
5	CA 証明書の検証の厳密化	○	○	－
6	リクエストログのログ出力フォーマットの変更	○	○	－
7	ディレクティブの変更	○	○	○
8	メッセージ ID の付加	○	○	－
9	SSL 関連コマンドの変更	○	○	－
10	サポートする暗号種別の変更	○	○	－
11	静的コンテンツキャッシュ機能の非サポート	○	○	－
12	SSL セション管理機能の非サポート	○	○	－
13	ディレクトリサービスを利用したユーザ認証とアクセス制御機能の非サポート	○	○	－
14	ステータスコードの変更	○	○	－
15	NameVirtualHost ディレクティブの非サポート	○	○	－
16	アクセスログのログ出力フォーマットの変更	○	○	－
17	WebSocket を使用する場合の設定変更	○	○	○
18	アクセスログに出力するリクエストヘッダの変更	○	○	○
19	リダイレクタ機能を使用する場合の設定変更	○	○	○
20	HTTP Server 起動コマンドでの環境変数の設定	○	○	○
21	持続型接続（KeepAlive）の有効範囲の変更	○	○	－
22	リバースプロキシ機能	○	○	－

（凡例）

- V8，V9，V11：それぞれ，アプリケーションサーバの Version 8，Version 9，Version 11（このバージョンを除く）を示します。
- 設定変更が必要な項目を次に示します。

  ○：移行時に変更が必要な項目です。

  －：移行時に変更が不要な項目です。

項目ごとの注意点を次に示します。設定変更が必要な場合は，インストールから起動までに実施してください。

1. プログラムプロダクトの名称変更

   プログラムプロダクト名を「Hitachi Web Server」から「Cosminexus HTTP Server」に変更しました。これに伴い，ログや HTTP 通信などに使用している名称が「Hitachi Web Server」から「Cosminexus HTTP Server」に変更されます。

   Windows 版では，標準で作成するサービス名も変更されます。ただし，旧バージョンをインストールした環境に「Hitachi Web Server」というサービス名が存在している場合は，上書きインストールを実施してもサービス名は変更されません。

2. プログラムメニューの提供終了（Windows 版だけ）

   スタートメニューにはこのプログラムプロダクトのメニューは作成されません。

3. SSLv2 プロトコルの非サポート

   SSLProtocol ディレクティブに SSLv2 が指定できなくなります。SSLv2 だけをサポートしているクライアントからリクエストした場合，SSL ハンドシェイクでエラーとなり，接続できません。

4. SSLv3 プロトコルの非サポート

   SSLProtocol ディレクティブに SSLv3 が指定できなくなります。SSLv3 だけをサポートしているクライアントからリクエストした場合，SSL ハンドシェイクでエラーとなり，接続できません。

5. CA 証明書の検証の厳密化

   クライアント認証時に行われる CA 証明書の検証が，規約に従って厳密化されました。CA 証明書が規約に従っていない場合，セキュリティ上問題となることがあるため，クライアント認証でエラーとなるおそれがあります。次に示す方法で確認し，規約に従った CA 証明書を使用してください。

   簡易確認方法

   1.次の証明書の内容表示コマンドを実行してください。

   openssl.sh x509 -text -in CA 証明書ファイル

   2.コマンド実行で表示されるテキストに「CA:TRUE」が含まれていない場合は，クライアント認証でエラーとなるおそれがあります。

   ・出力例（抜粋）

   X509v3 extensions:

   X509v3 Basic Constraints:

   CA:TRUE

HTTP Server 09-00-60 以降，09-65-60 以降，09-80-01 以降，09-87 以降の確認方法

1.次の証明書の検証コマンドを実行してください。

openssl.sh verify -CAfile CA 証明書ファイル クライアント証明書ファイル

オペランド

・-CAfile CA 証明書ファイル

CA 証明書ファイルを指定してください。ルート CA 証明書から中間 CA 証明書を発行し，証明書チェーンを構成している場合には，各 CA 証明書を一つのファイルにまとめて指定してください。

・クライアント証明書ファイル

CA から認証された証明書ファイルを指定してください。

2.コマンド実行で「OK」が表示されない場合は，クライアント認証でエラーとなるおそれがあります。

・出力例（[クライアント証明書] に「cert.pem」を指定した場合）

cert.pem: OK

6. リクエストログのログ出力フォーマットの変更

リクエストログに出力されるサーバプロセス ID の出力場所を時刻の後ろに変更します。詳細は「4.2.6 モジュールトレースの採取」，「4.2.7　リクエストトレースの採取」および「4.2.8　I/O フィルタトレースの採取」を参照してください。

7. ディレクティブの変更

幾つかのディレクティブが 09-80 以降で追加されています。

- HWSPrfId
- HWSWebSocketLog
- MaxSpareThreads
- MinSpareThreads
- SendBufferSize
- ServerLimit
- ThreadLimit

幾つかのディレクティブが 11-00 以降で追加されています。

- SSLCARevocationCheck
- SSLCARevocationFile
- SSLCipherSuite
- SSLEngine
- SSLOptions

幾つかのディレクティブが 11-20 以降で追加されています。

- Proxy100Continue

幾つかのディレクティブが 11-30 以降で追加されています。

- H2Direct
- H2MaxWorkerIdleSeconds
- H2MaxWorkers
- H2MinWorkers
- H2Padding
- H2SerializeHeaders
- H2StreamMaxMemSize
- H2Upgrade
- H2WindowSize
- HWSH2SendGoaway
- Protocols

幾つかのディレクティブを削除しています。次に示すディレクティブは 11-00 以降では指定できません。

- DefaultType
- HWSContentCacheSize
- HWSContentCacheMaxFileSize
- HWSStackTrace
- LDAPBaseDN
- LDAPNoEntryStatus
- LDAPRequire
- LDAPServerName
- LDAPServerPort
- LDAPSetEnv
- LDAPTimeout
- LDAPUnsetEnv
- NameVirtualHost
- SSLCacheServerPath
- SSLCacheServerPort
- SSLCacheServerRunDir
- SSLCertificateKeyPassword
- SSLCRLAuthoritative
- SSLCRLDERPath

- SSLCRLPEMPath
- SSLDenySSL
- SSLDisable
- SSLECCCertificateFile
- SSLECCCertificateKeyFile
- SSLECCCertificateKeyPassword
- SSLEnable
- SSLExportCertChainDepth
- SSLExportClientCertificates
- SSLFakeBasicAuth
- SSLRequireCipher
- SSLRequiredCiphers
- SSLSessionCacheSize
- SSLSessionCacheSizePerChild
- SSLSessionCacheTimeout

また，次に示すディレクティブはデフォルト値が変更となっています。09-80 以降でデフォルト値を変更しています。必要に応じて設定を見直してください。

- AllowOverride（デフォルト値：None）
- FileETag（デフォルト値：MTime Size）
- HWSRequestLogType（デフォルト値： module-info request proxy）
- KeepAliveTimeout（デフォルト値：5）
- Options（デフォルト値：None）
- SSLVerifyClient（デフォルト値：none）
- SSLVerifyDepth（デフォルト値：1）
- ThreadsPerChild（デフォルト値：64）
- Timeout（デフォルト値：60）
- UseCanonicalName（デフォルト値：Off）
- UserDir（デフォルト値：デフォルト値なし）
- RequestReadTimeout（デフォルト値：header=20 body=20※）

  注※　LoadModule ディレクティブで mod_reqtimeout.so を指定した場合。指定していない場合は，デフォルト値なし。

次に示すディレクティブは，仕様を変更しています。「6.　ディレクティブ」を参照し，必要に応じて設定を見直してください。

- AllowOverride
- CoreDumpDirectory
- CustomLog
- HWSGracefulStopLog
- HWSKeepStartServers
- HWSNotModifiedResponseHeaders
- HWSProxyPassReverseCookie
- HWSRequestLogType
- KeepAlive
- KeepAliveTimeout
- LimitRequestBody
- LimitRequestFields
- LimitRequestFieldsize
- LimitRequestLine
- LoadModule
- MaxClients
- MaxKeepAliveRequests
- MaxSpareServers
- MinSpareServers
- Protocols
- ProxyErrorOverride
- ProxyPass
- RequestReadTimeout
- SetEnvIf
- SetEnvIfNoCase
- SSLBanCipher
- SSLCACertificateFile
- SSLCACertificatePath
- SSLCertificateFile
- SSLCertificateKeyFile
- SSLCipherSuite
- SSLProtocol

- SSLRequireSSL
- SSLVerifyClient
- SSLVerifyDepth
- StartServers
- ThreadsPerChild

8. メッセージ ID の付加

   各メッセージにユニークな ID が付加されます。詳細については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照してください。

9. SSL 関連コマンドの変更

   sslccert コマンド，sslckey コマンド，keygen コマンド，および certutil コマンドが openssl.bat コマンドまたは openssl.sh コマンドに変更となります。sslccert コマンド，sslckey コマンド，sslpasswd コマンド，sslpasswd.bat コマンド，sslpasswd.sh コマンド，keygen コマンドおよび certutil コマンドは上書きインストール時に削除されます。openssl.bat コマンドまたは openssl.sh コマンドの詳細については「5.　SSL による認証，暗号化」を参照してください。

10. サポートする暗号種別の変更

    サポートする暗号種別を変更します。詳細は「6.2.7　S で始まるディレクティブ」の「(29) SSLCipherSuite 〔TLSv1.3〕 暗号種別〔:暗号種別…〕」を参照し，必要に応じて設定を見直してください。

11. 静的コンテンツキャッシュ機能の非サポート

    静的コンテンツキャッシュ機能が非サポートとなります。

12. SSL セション管理機能の非サポート

    Web サーバの SSL セション管理機能が非サポートとなります。

13. ディレクトリサービスを利用したユーザ認証とアクセス制御機能の非サポート

    ディレクトリサービスを利用したユーザ認証機能とアクセス制御機能が非サポートとなります。

14. ステータスコードの変更

    この製品の Web サーバがベースとしている Apache HTTP Server のバージョンを 2.4 に変更したため，ステータスコードが変更となりますので，「付録 A　ステータスコード」を参照してください。

    リバースプロキシ機能使用時にバックエンドサーバと通信できない場合に 502 エラー以外に 503 エラーが返ることがあります。

    リクエストラインが受信されずにタイムアウトになった場合にステータスコード 408 としてアクセスログに出力されます。

15. NameVirtualHost ディレクティブの非サポート

    NameVirtualHost ディレクティブの非サポートにより，複数の VirtualHost ブロックに重複した IP アドレスを指定している場合にはサーバ名に基づくバーチャルホストとして動作します。旧バージョンの IP アドレスに基づくバーチャルホストの動作にならない可能性があります。必要に応じて設定を見直してください。

16. アクセスログのログ出力フォーマットの変更

標準提供の httpsd.conf に含まれる一部のログ出力フォーマットの定義を変更しました。変更したログ出力フォーマットは，次のラベル名です。

- hws_trace
- hws_std

hws_std は標準提供の httpsd.conf でアクセスログのフォーマットとして使用されています。

標準提供の httpsd.conf で指定されているログ出力フォーマットを次に示します。

- Windows の場合

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥" %I %O" combinedio

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" combined

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common

  LogFormat "%{Referer}i -> %U" referer

  LogFormat "%{User-agent}i" agent

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b %{hws_thread_id}P ¥"%{hws_ap_root}n¥" %I %O %X %D ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" hws_trace

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b %T %{hws_thread_id}P ¥"%{hws_ap_root}n¥"" hws_std

- UNIX(worker MPM)の場合

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥" %I %O" combinedio

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" combined

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common

  LogFormat "%{Referer}i -> %U" referer

  LogFormat "%{User-agent}i" agent

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b %P %{tid}P ¥"%{hws_ap_root}n¥" %I %O %X %D ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" hws_trace

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b %T %P %{tid}P ¥"%{hws_ap_root}n¥"" hws_std

- UNIX(prefork MPM)の場合

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥" %I %O" combinedio

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" combined

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common

  LogFormat "%{Referer}i -> %U" referer

  LogFormat "%{User-agent}i" agent

  LogFormat "%h %l %u %t ¥"%r¥" %>s %b %P ¥"%{hws_ap_root}n¥" %I %O %X %D ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" hws_trace

LogFormat "%h %l %u %t ¥"%r¥" %>s %b %T %P ¥"%{hws_ap_root}n¥"" hws_std

17. WebSocket を使用する場合の設定変更

    11-20 以降から，WebSocket 通信をする場合に必要なモジュールが変更になりました。WebSocket 通信を使用するためには，mod_proxy モジュール，mod_proxy_http モジュール，および mod_proxy_wstunnel モジュールが必要です。「4.15　WebSocket による通信」を参照して，必要なモジュールを設定してください。

18. アクセスログに出力するリクエストヘッダの変更

    CustomLog ディレクティブで指定するログのフォーマットで%{header_name}i を指定した場合，リクエストヘッダの値をログに出力することができますが，11-20 以降はクライアントから受け付けたリクエストヘッダだけを出力します。リバースプロキシ機能が付加するリクエストヘッダは，アクセスログに出力されませんのでご注意ください。ログに出力されないリクエストヘッダの例を次に示します。

    - X-Forwarded-For
    - X-Forwarded-Host
    - X-Forwarded-Server

19. リダイレクタ機能を使用する場合の設定変更

    11-20 以降から，リダイレクタ機能を使用する場合に環境変数の設定が必要となりました。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「10.3.3　アプリケーションサーバ 11-00 または 11-10 から，11-20 までの仕様変更」の「(4)　リダイレクタ機能」を参照してください。

20. HTTP Server 起動コマンドでの環境変数の設定

    HTTP Server の起動コマンドを実行する際，事前に PRFSPOOL 環境変数の設定が必要です。詳細は「2.4.2　HTTP Server を起動，停止する（httpsdctl コマンド）」および「2.4.3　HTTP Server を起動する（httpsd コマンド）」を参照してください。

21. 持続型接続（KeepAlive）の有効範囲の変更

    持続型接続の KeepAlive ディレクティブおよび KeepAliveTimeout ディレクティブは，リバースプロキシ機能使用時での，バックエンドサーバとの接続には影響を与えません。

22. リバースプロキシ機能

    リバースプロキシ機能は，バックエンドサーバとの接続を再利用しようとするコネクションプール方式に変更しました。このことから，リバースプロキシ機能はバックエンドサーバに接続した各コネクションを解放せずに接続状態を維持します。

付録 D　旧バージョンから移行する場合の設定

SSL 機能を使用する場合，HTTP Server の旧バージョンから V11 以降に移行するに当たり，次の変更を実施します。

項番	項目	旧バージョン	V11 以降	説明
1	SSLEnable/ SSLDisable の指定がない場合のデフォルト	SSL 機能有効	SSL 機能無効	SSL 機能を有効にしたい場合は，SSLEngine On を指定してください。 ただし，<VirtualHost>ブロック内で SSL を有効にしたい場合，有効にしたい<VirtualHost>ブロック内で SSLEngine On を指定してください。
2	SSL 機能を無効にする指定	SSLDisable	SSLEngine Off	SSLEngine Off に置き換えてください。 なお，11-00 以降では SSLDisable ディレクティブの指定のままでも SSLEngine Off が指定されたものとして動作します。また，Web サーバ全体で SSL 機能を無効にする場合には，その他 SSL から始まるディレクティブの指定を削除もしくはコメント化してください。
3	SSL 機能を有効にする指定	SSLEnable	SSLEngine On	SSLEngine On に置き換えてください。 ただし，<VirtualHost>ブロック内で SSL を有効にしたい場合，有効にしたい<VirtualHost>ブロック内で SSLEngine On を指定してください。 なお，SSL 通信だけを有効にした Web サーバを Management Server を使用して起動する場合は，Web サーバの動作確認レベルを「1：プロセスの存在を確認するだけ」に変更してください。※1
4	使用する暗号種別の指定	SSLRequiredCiphers	SSLCipherSuite	SSLCipherSuite に置き換えてください。
5	使用するプロトコルの指定	SSLProtocol TLSv1 TLSv11 TLSv12	SSLProtocol +TLSv1 +TLSv1.1 +TLSv1.2	プロトコルの記述方法が異なります。使用するプロトコルを追加する場合はプロトコルの前に’+’を付加して指定します。連続したプロトコルバージョンを指定していない場合，古いプロトコルバージョンは無効になります。
6	SSLRequireSSL の指定可能場所	httpsd.conf, <VirtualHost>, <Directory>，.htaccess	<Directory>, .htaccess	httpsd.conf，<VirtualHost>で指定している場合は，<Directory>，.htaccess で指定するように変更してください。
7	SSL 機能のリクエストを禁止する設定	SSLDenySSL	なし	SSLDenySSL の指定を削除してください。 SSL 機能全体を無効にする指定などを検討してください。
8	サーバ認証で使用する CA 証明書の指定	SSLCACertificateFile	SSLCertificateFile サーバ証明書と CA 証明書を連結	SSLCACertificateFile で指定されたファイル内の証明書のうち，サーバ認証で使用する証明書を SSLCertificateFile で指定するファイルに追加してくだ

項番	項目	旧バージョン	V11 以降	説明
			したファイルを指定	さい。追加する順序は Web サーバの証明書に続き，中間 CA の証明書，ルート証明書を追加してください。 なお，旧バージョンでは，サーバ認証に使用する CA 証明書を SSLCACertificateFile ディレクティブに指定している場合は，SSLCACertificateFile ディレクティブに指定した CA 証明書をサーバ認証に使用していました。
9	暗号化秘密鍵のパスワード格納ファイルの指定	SSLCertificateKeyPassword, SSLECCCertificateKeyPassword	なし	SSLCertificateKeyPassword, SSLECCCertificateKeyPassword の指定を削除してください。※2
10	DER 形式の CRL を指定	SSLCRLDERPath で DER 形式のファイルを格納したディレクトリを指定	DER 形式は使用できない	DER 形式を PEM 形式に変換したファイルを SSLCARevocationFile で指定してください。※3 複数のファイルがある場合には連結してファイルを作成してください。
11	PEM 形式の CRL を指定	SSLCRLPEMPath で PEM 形式のファイルを格納したディレクトリを指定	SSLCARevocationFile で CRL のファイルを指定	複数の CRL ファイルがある場合には連結したファイルを SSLCARevocationFile で指定してください。
12	クライアント証明書を用いた Basic 認証の指定	SSLFakeBasicAuth	SSLOptions +FakeBasicAuth	SSLOptions ディレクティブに+FakeBasicAuth を指定してください。
13	クライアント証明書の環境変数への設定	SSLExportClientCertificates	SSLOptions +ExportCertData	SSLOptions ディレクティブに+ExportCertData を指定してください。PEM 形式の証明書を環境変数に設定します。 ただし，クライアント証明書だけでなく，サーバ証明書も環境変数に設定します。
14	楕円曲線暗号用証明書および秘密鍵指定ディレクティブ	SSLECCCertificateFile SSLECCCertificateKeyFile	SSLCertificateFile SSLCertificateKeyFile	SSLCertificateFile，SSLCertificateKeyFile で楕円曲線暗号用の証明書と秘密鍵を指定してください。 RSA 暗号と楕円曲線暗号のサーバ証明書および秘密鍵は，それぞれ一つずつだけ指定できます。
15	クライアント認証の設定をする SSLVerifyClient の指定値	0 | 1 | 2	none | optional | require	SSLVerifyClient の指定値を次のとおりに数字から文字列に変更してください。 0：none 1：optional 2：require
16	クライアント認証時の段階数 SSLVerifyDepth の指定	末端のクライアント証明書を含んだ階層数（認証局の数＋ 1）	末端のクライアント証明書を含まない階層数 （認証局だけの数）	証明書のチェインをたどる段階数を見直してください。

項番	項目	旧バージョン	V11 以降	説明
17	CGI で暗号化通信に関する環境変数を使用している場合	−	−	環境変数名が変更になっています。
18	暗号種別でのアクセス制御	SSLBanCipher SSLRequireCipher	SSLBanCipher	SSLRequireCipher で許可する暗号種別を指定していた場合，SSLBanCipher で拒否する暗号種別を指定してください。 また，httpsd.conf，<VirtualHost>で指定している場合は，<Directory>，.htaccess で指定するように変更してください。
19	ログフォーマット%{version}c で出力されるプロトコルバージョン	−	−	出力されるプロトコルバージョンの表記が変更になっています。 変更前 TLS1 TLS11 TLS12 変更後 TLSv1 TLSv1.1 TLSv1.2 TLSv1.3
20	ログフォーマット%{clientcert}c で出力されるクライアント証明書の subject の Distinguished Name	−	−	出力される情報の区切り文字が「/」から「,」に変更になっています。 変更前 /C=JP/ST=Kanagawa/L=Yokohama-shi/O=client/OU=client/CN=client 変更後 CN=client,OU=client,O=client,L=Yokohama-shi,ST=Kanagawa,C=JP

注※1

Management Server では Web サーバが正常に動作しているかどうかを，運用管理エージェントを介して，プロセスの存在を確認したり，Web サーバに HTTP でアクセスしたりすることで判断しています。

Web サーバで SSL 通信だけを有効にしている場合は，Web サーバの動作確認レベルはプロセスの存在を確認するだけに変更（adminagent.hws.watch.level=1 を設定）してください。

詳細は，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1.18　運用管理エージェントを使用するために設定する情報」を参照してください。

注※2

パスワード保護された秘密鍵からパスワード保護を無効にした秘密鍵に変換したファイルを SSLCertificateKeyFile ディレクティブに指定します。次のコマンドで秘密鍵のパスワード解除を行います。

- RSA 暗号を利用した秘密鍵の場合

  openssl.bat rsa -in パスワード保護された秘密鍵ファイル -out パスワードを解除した秘密鍵ファイル

- 楕円曲線暗号を利用した秘密鍵の場合

  openssl.bat pkcs8 -topk8 -in パスワード保護された秘密鍵ファイル -out パスワードを解除した秘密鍵ファイル -nocrypt

  UNIX の場合は openssl.bat を openssl.sh に置き換えて実行してください。

注※3

次のコマンドで DER 形式のファイルを PEM 形式に変換します。

openssl.bat crl -inform DER -in 入力ファイル -outform PEM -out 出力ファイル

UNIX の場合は openssl.bat を openssl.sh に置き換えて実行してください。

付録 E　運用管理ポータルによるリバースプロキシを使用した prefork MPM の論理 Web サーバの構築方法

UNIX 版で運用管理ポータルを使用して論理 Web サーバを構築する際，J2EE サーバの連携方法でリバースプロキシを選択した場合は worker MPM の論理 Web サーバが構築され，リダイレクタを選択した場合は prefork MPM の論理 Web サーバが構築されます。ここではリバースプロキシを使用する prefork MPM の論理 Web サーバを構築するための方法について説明します。

次の説明では，運用管理ドメイン内のホストの定義，パフォーマンストレーサの設定ができていると仮定します。

1. マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「9.10.1　Web サーバの追加」の「(2)表示手順」に従い，［Web サーバの追加］画面を表示し，「(3)操作手順」で Web サーバを追加する際に［J2EE サーバ連携］フィールドで「リダイレクタ」を選択して［追加］ボタンをクリックします。

2. マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「9.11　論理サーバの一括セットアップ」の手順に従い，1.で追加した Web サーバをセットアップします。

3. マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.9.1　Web サーバの設定」の「(2)表示手順」に従い，［Web サーバの設定］画面を表示します。［設定ファイルの参照］ボタンをクリックして［設定ファイルの内容］に表示された設定を編集します。編集方法は次のとおりです。

   - 連携する J2EE サーバへのリバースプロキシの設定を追加します。詳細は「4.7　リバースプロキシの設定」を参照してください。リバースプロキシの設定例を示します。

     （例）

     ```
     LoadModule proxy_module libexec/mod_proxy.so
     LoadModule proxy_http_module libexec/mod_proxy_http.so
     ProxyPass / http://backend.example.com:8008/
     ProxyPassReverse / http://backend.example.com:8008/
     ```

   - 「Include “< Application Server のインストールディレクトリ>/CC/web/redirector/servers/<論理 Web サーバの実サーバ名>/mod_jk.conf”」が定義されている場合はその定義を削除してください。

   - ほかに Web サーバの設定を変更する場合は［設定ファイルの内容］を適宜修正します。編集が完了したら，画面の［設定ファイルの内容を直接設定します］を選択し，［適用］ボタンをクリックします。

4. マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.9.2　リダイレクタの設定（リダイレクタを使用した場合)」の「(2)表示手順」に従い，［リダイレクタの設定］画面を表示します。［利用するパフォーマンストレーサ］で利用するパフォーマンストレーサを設定し，［適用］ボタンをクリックします。

5. マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.10　論理サーバの設定ファイルの配布」に従い，これまでに設定した情報をサーバに配布します。

これで，リバースプロキシを使用する prefork MPM の論理 Web サーバの構築は完了です。

付録 F　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引

記号

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W

あ

い