SSL 終了の設定

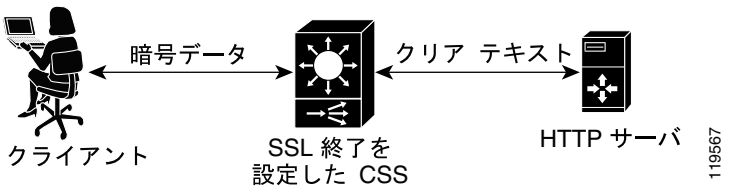
この章では、CSS を SSL 終了用の仮想 SSL サーバとして設定するための手順を説明します。この章の主な内容は次のとおりです。

SSL 終了の概要

CSS で SSL 終了が実行されるのは、プロキシ サーバとして機能している SSL モジュールがクライアントからの SSL 接続を終了して、サーバへの TCP 接続を確立するときです。SSL モジュールは、SSL 接続を終了するときにデータを復号化し、それをロード バランシング判定用のクリア テキストとして CSS に送信します。CSS で受信されたデータは、クリア テキストとして HTTP サーバに送信されるか SSL モジュールに返され、そこで暗号化されて設定済みのバックエンド SSL サーバに送信されます。

図 4-1 に、SSL サーバとして機能する SSL モジュールのある CSS とクライアント間の SSL 接続を示します。

図 4-1　　SSL 終了



SSL モジュール、クライアント、およびサーバ間の SSL 情報の流れは、SSL プロキシ リストによって決定されます。SSL プロキシ リストは、インデックス エントリで識別される 1 つ以上の仮想 SSL サーバの定義で構成されています。CSS の SSL モジュールは、仮想 SSL サーバを使用して、クライアントとサーバ間の SSL 通信を適切に処理し、終端させます。1 つの SSL プロキシ リストには、最大 256 の仮想 SSL サーバを定義できます。

SSL プロキシ リストを作成して仮想 SSL サーバを定義したら、リストをアクティブにする必要があります。続いて、そのプロキシ リストをサービスに追加し、SSL 設定データの SSL モジュールへの転送を開始します。サービスをアクティブ化すると、CSS はデータをモジュールに転送します。次に各 SSL サービスを SSL コンテンツ ルールに追加します。

SSL プロキシ リストの作成

SSL プロキシ リストは、同じ SSL サービスに関連付けられている、いくつかの仮想 SSL サーバのグループです。SSL プロキシ リストの作成には、**ssl-proxy-list** コマンドを使用します。

ssl-proxy-list 設定モードには、ACL モード、ブート モード、グループ モード、rmon モード、および所有者設定モードを除く、ほとんどの設定モードからアクセスできます。また、このコマンドを ssl-proxy-list 設定モードから使用して、別の SSL プロキシ リストにアクセスすることもできます。SSL プロキシ リスト名は、1 ～ 31 文字のテキスト文字列を引用符で囲まずに入力します。

たとえば、SSL プロキシ リスト ssl_list1 を作成するには、次のコマンドを入力します。

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

SSL プロキシ リストを作成すると、CLI が ssl-proxy-list 設定モードに入ります。

```
(config-ssl-proxy-list[ssl_list1])#
```

既存の SSL プロキシ リストを削除するには、次のコマンドを入力します。

```
(config)# no ssl-proxy-list ssl_list1
Delete ssl-list <ssl_list1>, [y/n]: y
```

（注） SSL サービスが使用中の場合、サービスに含まれるアクティブな SSL プロキシ リストは削除できません。この場合は、最初に SSL サービスを一時停止してから、特定の SSL プロキシ リストを削除する必要があります。

SSL プロキシ リストへの説明の追加

SSL プロキシ リストの説明を指定するには、**description** コマンドを使用します。説明は、スペースを含む 64 文字以内のテキスト文字列を引用符で囲んで入力します。

たとえば、*ssl_list1* SSL プロキシ リストに説明を追加するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# description "This is the SSL list
for www.brandnewproducts.com"
```

特定の SSL プロキシ リストの説明を削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no description
```

SSL プロキシ リストでの仮想 SSL サーバの設定

ここでは、SSL プロキシ リストで 1 つ以上の仮想 SSL サーバを作成する方法について説明します。**ssl-server** コマンドを使用して SSL プロキシ リスト内にインデックス エントリを作成し、このエントリにこの仮想 SSL サーバに関連する個々の SSL パラメータを設定します。CSS の SSL モジュールは、仮想 SSL サーバを使用して、クライアントとサーバ間の SSL 通信を適切に処理し、終端させます。SSL プロキシ リスト パラメータを設定する前に、SSL サーバのインデックス番号を定義する必要があります。1 つの SSL プロキシ リストには、最大で 256 の仮想 SSL サーバを定義できます。

たとえば、Brand New Products という e コマース ベンダーが、CSS による SSL 終了の導入を検討しているとします。この場合、同社 Web サイト（https://www.brandnewproducts.com）宛ての全トラフィックが、実際には CSS 内の SSL モジュールに搬入されるように設定します。そのためには、SSL プロキシ リスト内で仮想 SSL サーバの VIP アドレスを指定し、このリストをコンテンツ ルールと同じ VIP アドレスにリンクする必要があります。この VIP アドレスを使用するには、次の各 SSL 設定パラメータが必要です。

- コンテンツ ルールに対応する仮想 TCP ポート番号の識別情報
- 識別用の既存の RSA または DSA 証明書
- 暗号化および署名を実行するための適切な SSL キー ペア（RSA キー ペアを使用している場合）
- CSS SSL セキュリティに Diffie-Hellman キー交換アルゴリズムが必要な場合は Diffie-Hellman パラメータ
- 暗号スイートの割り当て

（注） アクティブな SSL プロキシ リストには、変更を加えることはできません。変更を加える前に SSL プロキシ リストの使用を一時停止し、変更が終了したらリストを再度アクティブ化します。CCS は、そのプロキシ リストを使用して SSL サービスへ追加情報または変更内容を送信します。詳細については、「SSL プロキシ リストの変更」を参照してください。

次の項で、SSL プロキシ リストでの仮想 SSL サーバの作成方法について説明します。

- SSL サーバ インデックスの作成
- 仮想 IP アドレスの指定
- 仮想ポートの指定
- サーバ認証用の証明書、キー、および暗号スイートの割り当て
- クライアント認証の設定
- HTTP ヘッダー挿入の設定
- SSL または TLS バージョンの指定
- TCP FIN メッセージ単独でのクライアント接続の終了
- セキュア URL リライトの指定
- SSL セッション キャッシュ タイムアウトの指定
- SSL セッションのハンドシェイク再ネゴシエーションの指定
- SSL キュー データの遅延時間の設定
- SSL TCP クライアント側の接続タイムアウト値の指定
- SSL TCP サーバ側接続タイムアウト値の指定
- SSL TCP 接続における確認応答遅延の変更
- SSL TCP 接続の Nagle アルゴリズムの指定
- SSL TCP 接続の TCP バッファリングの指定

SSL プロキシ リストの設定情報を表示する方法については、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。

SSL サーバ インデックスの作成

SSL プロキシ リスト パラメータを設定する前に、仮想 SSL サーバを作成する必要があります。SSL プロキシ リストに SSL 固有のパラメータを指定するには、**ssl-server** *number* コマンドを使用します。このコマンドを実行すると、仮想 SSL サーバに関連付けられた個々の SSL パラメータ（VIP アドレス、証明書名、キーペアなど）の設定に使用する番号（インデックス エントリ）が SSL プロキシ リスト内に作成されます。1 ～ 256 の整数を入力します。

たとえば、仮想 SSL サーバ 20 を指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20
```

仮想 SSL サーバを SSL プロキシ リストから削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20
```

仮想 IP アドレスの指定

この VIP アドレスは SSL モジュールによって、受け入れるべきトラフィックを識別する手段として使用されます。VIP アドレスは、コンテンツ ルールに設定されている VIP アドレスと同じものを指定してください。仮想 IP（VIP）アドレスを指定するには、**ssl-server** *number* **vip address** *ip_or_host* コマンドを使用します。SSL コンテンツ ルールに一致する仮想 SSL サーバの VIP アドレスを入力します。「SSL 終了のコンテンツ ルールの設定」を参照してください。

有効な VIP アドレスを、ドット付き 10 進表記（たとえば、192.168.11.1）またはニーモニック ホスト名（たとえば、myhost.mydomain.com）で入力します。

（注） VIP アドレスをニーモニック ホスト名で入力した場合、CSS はドメイン ネーム サービス（DNS）を使用し、myhost.mydomain.com などのホスト名を、192.168.11.1 などの IP アドレスに変換します。ホスト名が解決できない場合、VIP アドレス設定は受け入れられず、ホストの解決に失敗したことを示すエラー メッセージが表示されます。DNS の設定の詳細については、『*Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*』を参照してください。

仮想 SSL サーバの VIP ポートを設定しないまま SSL プロキシ リストのアクティブ化を試みる（「SSL TCP 接続の Nagle アルゴリズムの指定」参照）と、エラー メッセージがログに出力され、SSL プロキシ リストはアクティブ化されません。設定された SSL サービスでコンテンツ ルールをアクティブ化すると、コンテンツ ルールで設定された各 VIP アドレスが、追加された各サービスの SSL プロキ

シ リストで設定された 1 つ以上の VIP アドレスに一致するかが確認されます。一致しない場合、エラー メッセージがログに記録され、コンテンツ ルールはアクティブ化されません。

たとえば、コンテンツ ルールで設定した VIP アドレスに一致する、仮想 SSL サーバの VIP アドレスを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 vip address
192.168.3.6
```

仮想 SSL サーバから VIP アドレスを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 vip address
```

仮想ポートの指定

SSL モジュールは仮想ポートを使用して、受け入れるトラフィックを特定します。仮想 SSL サーバの仮想 TCP ポート番号を指定するには、**ssl-server** *number* **port** *number* コマンドを使用します。SSL コンテンツ ルールに一致する TCP ポート番号（SSL コンテンツ ルールで使用される TCP ポート番号）を入力します。

ポート番号は 1 ～ 65535 の範囲内で指定します。デフォルトのポート番号は 443 です。コンテンツ ルールに設定したポートと一致するポート番号を指定してください（「SSL 終了のコンテンツ ルールの設定」参照）。

仮想 SSL サーバの仮想ポートを設定しないまま SSL プロキシ リストのアクティブ化を試みる（「SSL TCP 接続の Nagle アルゴリズムの指定」参照）と、エラー メッセージがログに出力され、SSL プロキシ リストはアクティブ化されません。設定された SSL サービスでコンテンツ ルールをアクティブ化すると、コンテンツ ルールで設定された各仮想ポートが、追加された各サービスの SSL プロキシ リストで設定された 1 つ以上のポートに一致するかどうかを CSS は確認します。一致しない場合、エラー メッセージがログに記録され、コンテンツ ルールはアクティブ化されません。

たとえば、仮想ポートを 444 に設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 port 444
```

仮想ポートをデフォルトの 443 にリセットするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 port
```

サーバ認証用の証明書、キー、および暗号スイートの割り当て

CSS は、サーバ認証の目的で各クライアントに送信するサーバ証明書をサポートします。証明書を仮想 SSL サーバと関連付けるには、CSS にインポートしたか、CSS で生成した証明書およびキー（第 3 章「SSL 証明書とキーの設定」参照）を割り当てる必要があります。また、証明書およびキーに対応する暗号スイートも割り当てる必要があります。

ここでは、サーバ認証のための各種設定について説明します。

- RSA 証明書名の指定
- RSA キー ペア名の指定
- DSA 証明書名の指定
- DSA キー ペア名の指定
- Diffie-Hellman パラメータ ファイル名の指定
- 暗号スイートの指定

RSA 証明書名の指定

認証およびパケット暗号化の公開 / 秘密キー ペアの交換で使用される RSA 証明書アソシエーションの名前を指定するには、**ssl-server** *number* **rsacert** *name* コマンドを使用します。既存の RSA 証明書アソシエーションのリストを表示するには、**ssl-server** *number* **rsacert ?** コマンドを使用します。

指定した RSA 証明書は CSS にロード済みで、関連付けが済んでいる必要があります（第 3 章「SSL 証明書とキーの設定」参照）。適切な RSA 証明書アソシエーションが存在しない場合、SSL プロキシ リストをアクティブ化しようとするとエラー メッセージがログに記録され、リストはアクティブ化されません。

たとえば、以前に定義した、*rsacert* という名前の RSA 証明書アソシエーションを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1
```

特定の仮想 SSL サーバから RSA 証明書アソシエーションを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsacert
```

RSA キー ペア名の指定

RSA キー ペア アソシエーションの名前を指定するには、**ssl-server** *number* **rsakey** *name* コマンドを使用します。RSA キー ペアは、他のデバイス（クライアントまたはサーバ）が CSS と SSL 証明書を交換するために必要です。既存の RSA キー ペア アソシエーションのリストを表示するには、**ssl-server** *number* **rsakey ?** コマンドを使用します。

RSA キー ペアは CSS にロード済みで、関連付けが済んでいる必要があります（第 3 章「SSL 証明書とキーの設定」参照）。適切な RSA キー ペア アソシエーションが存在しない場合、SSL プロキシ リストをアクティブ化しようとするとエラー メッセージがログに記録され、リストはアクティブ化されません。

たとえば、*rsakey* という名前の定義済みの RSA キー ペア アソシエーションを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1
```

特定の仮想 SSL サーバから RSA キー ペア アソシエーションを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsakey
```

DSA 証明書名の指定

デジタル シグニチャの交換に使用する DSA 証明書アソシエーションの名前を指定するには、**ssl-server** *number* **dsacert** *name* コマンドを使用します。既存の DSA 証明書アソシエーションのリストを表示するには、**ssl-server** *number* **dsacert ?** コマンドを使用します。

指定した DSA 証明書は CSS にロード済みで、関連付けが済んでいる必要があります（第 3 章「SSL 証明書とキーの設定」参照）。適切な RSA 証明書アソシエーションが存在しない場合、SSL プロキシ リストをアクティブ化しようとするとエラー メッセージがログに記録され、リストはアクティブ化されません。

たとえば、*dsacert* という名前の定義済みの DSA 証明書アソシエーションを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsacert mydsacert1
```

特定の仮想 SSL サーバから DSA 証明書アソシエーションを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsacert
```

DSA キー ペア名の指定

DSA キー ペアはパケット データの署名に使用されます。このキー ペアは、他のデバイス（クライアントまたはサーバ）と CSS 間で SSL 証明書を交換する前に用意する必要があります。DSA キー ペア アソシエーション名を指定するには、**ssl-server** *number* **dsakey** *name* コマンドを使用します。既存の DSA キー ペア アソシエーションのリストを表示するには、**ssl-server** *number* **dsakey ?** コマンドを使用します。

DSA キー ペアは CSS にロード済みで、関連付けが済んでいる必要があります（第 3 章「SSL 証明書とキーの設定」参照）。適切な DSA キー ペア アソシエーションが存在しない場合、SSL プロキシ リストをアクティブ化しようとするとエラー メッセージがログに記録され、リストはアクティブ化されません。

たとえば、*dsakey* という名前の設定済みの DSA キー アソシエーションを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsakey mydsakey1
```

特定の仮想 SSL サーバから DSA キー ペア アソシエーションを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsakey
```

Diffie-Hellman パラメータ ファイル名の指定

Diffie-Hellman キー交換パラメータ ファイルを使用すると、データ交換を行う 2 つのデバイスはパケット暗号化と認証に関する共有キーを共同で作成します。Diffie-Hellman キー交換パラメータ ファイル アソシエーションの名前を指定するには、**ssl-server** *number* **dhparam** *name* コマンドを使用します。既存の Diffie-Hellman キー交換パラメータ ファイルのリストを表示するには、**ssl-server** *number* **dhparam ?** コマンドを使用します。

Diffie-Hellman パラメータ ファイルは CSS にロード済みで、関連付けが済んでいる必要があります（第 3 章「SSL 証明書とキーの設定」参照）。適切な Diffie-Hellman パラメータ ファイル アソシエーションが存在しない場合、SSL プロキシ リストをアクティブ化しようとするとエラー メッセージがログに記録され、リストはアクティブ化されません。

定義済みの Diffie-Hellman パラメータ ファイル アソシエーションを指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dhparam mydhparams1
```

Diffie-Hellman パラメータ ファイル アソシエーションを特定の仮想 SSL サーバから削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dhparam
```

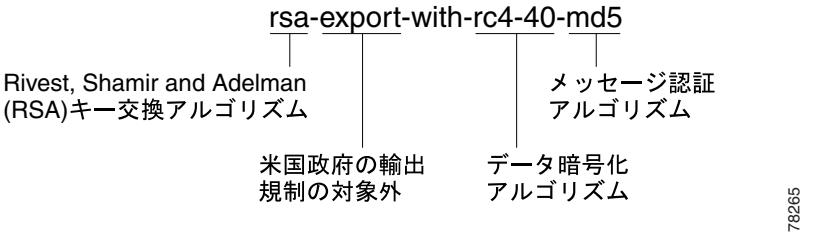
暗号スイートの指定

SSL プロトコルは、サーバやクライアントの相互認証、証明書の送信、セッション キーの設定などの操作に使用できる様々な暗号アルゴリズム（暗号化）をサポートしています。クライアントとサーバがサポートする暗号スイート（暗号のセット）は、サポートする SSL のバージョン、受け入れ可能な暗号の強度に関する企業ポリシー、SSL 対応ソフトウェアの輸出に関する政府の規制といった様々な要因によって、それぞれ異なる可能性があります。サーバとクライアントは、相互認証と証明書の送信、およびセッション キーの設定に使用する暗号スイートを、ネゴシエーションを通じて取り決めます。このネゴシエーションの方法を決定するのは、SSL ハンドシェイク プロトコルの機能の 1 つです。

（注）　輸出可能な暗号スイートは、米国のソフトウェア製品に関する輸出規制に定義されているように、他の一部の暗号スイートに比べ強度が弱い暗号スイートです（たとえば、128 ビットの暗号化を行う 3DES または RC4）。輸出可能な暗号スイートは、米国からほとんどの国に輸出でき、輸出可能な製品に利用できる最も強力な暗号化を提供します。

各暗号スイートで、キー交換アルゴリズムのセットが定義されています。図 4-2 は、rsa-export-with-rc4-40-md5 暗号スイートに関連付けられているアルゴリズムをまとめたものです。

図 4-2　暗号スイート アルゴリズム



SSL プロキシ リストに暗号スイートを割り当てるには、**ssl-server** *number* **cipher** コマンドを使用します。選択する暗号スイートは、CSS でインポートまたは生成した証明書やキーと相互に関連している必要があります。たとえば、**all-cipher-suites** を選択する場合は、SSL プロキシ リストをアクティブ化する前に、RSA 証明書とキー、DSA 証明書とキー、および Diffie-Hellman パラメータ ファイルを用意する必要があります。

利用可能な SSL バージョンごとに、サポートされる暗号スイートのリストが個別に用意され、暗号化アルゴリズムおよびパラメータの選択が示されています。選択する暗号スイートは、環境、使用中の証明書とキー、およびセキュリティ要件によって決まります。デフォルトでは、サポートされる暗号スイートは有効になっていません。

このコマンドのシンタックスは次のとおりです。

ssl-server *number* **cipher** *name ip_address_or_hostname port* {**weight** *number*}

このコマンドのオプションと変数は次のとおりです。

- **ssl-server** *number* : SSL プロキシ リスト内の仮想 SSL サーバを識別するために使用する番号
- **cipher** *name*：特定の暗号スイートの名前（表 4-1 参照）
- *ip_address_or_hostname* : 暗号スイートで使用されるバックエンドのコンテンツ ルールに割り当てる IP アドレス。IP アドレスはドット付き 10 進表記（たとえば、192.168.11.1）またはニーモニック ホスト名（たとえば、myhost.mydomain.com）で指定します。
- *port*：バックエンド HTTP 接続の送信トラフィックが使用するバックエンドのコンテンツ ルールの TCP ポート
- **weight** *number* ：オプションのパラメータ。暗号スイートに優先度を割り当てます。10 が最大の優先度です。デフォルトでは、設定された暗号スイートの優先度は 1 です。SSL モジュールは、使用する暗号スイートのネゴシエーション時に、最も優先度の高い暗号スイートに基づいてクライアントリストから選択します。優先度が高ければ、指定した暗号スイートが有利になります。暗号スイートに優先度を設定するには、1 ～ 10 の数値を入力します。デフォルトは 1 です。

たとえば、優先度 5 が割り当てられた *dhe-rsa-with-3des-ede-cbc-sha* 暗号スイートを選択するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha 192.168.11.1 80 weight 5
```

特定の仮想 SSL サーバから特定の暗号スイートを削除するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha
```

表 4-1 に、特定の SSL サーバ（および対応する SSL プロキシ リスト）でサポートされるすべての暗号スイートと値をリストします。表 4-1 には、暗号スイートの CSS からのエクスポートの可否と、それぞれの暗号スイートに必要な認証証明書と暗号キーも示しています。

デフォルトの設定を使用するか **all-cipher-suite** オプションを選択した場合、暗号スイートは表 4-1 と同じ順序で、rsa-with-rc4-128-md5 から順次送信されます。

（注） **all-cipher-suites** 設定は、特定の暗号スイートが設定されていない場合だけに使用されます。**all-cipher-suites** 設定に戻すには、明示的に設定したすべての暗号スイートを削除する必要があります。

注意 dh-anon シリーズの暗号スイートは、どのパーティも認証されない完全に匿名の Diffie-Hellman 通信用です。この暗号スイートは攻撃を受けやすい点に注意してください。

タイトルに「export」が含まれる暗号スイートは、米国政府による輸出規制の対象外で、キー サイズが制限された暗号化アルゴリズムを使用しています。

表 4-1　CSS でサポートされる SSL 暗号スイート

暗号スイート	エクスポートの可否	使用する認証証明書	使用されるキー交換アルゴリズム
all-cipher-suites	不可	RSA 証明書、DSA 証明書	RSA キー交換、Diffie-Hellman
rsa-with-rc4-128-md5	不可	RSA 証明書	RSA キー交換
rsa-with-rc4-128-sha	不可	RSA 証明書	RSA キー交換
rsa-with-des-cbc-sha	不可	RSA 証明書	RSA キー交換
rsa-with-3des-ede-cbc-sha	不可	RSA 証明書	RSA キー交換
dhe-dss-with-des-cbc-sha	不可	DSA （DSS） 証明書	Ephemeral Diffie-Hellman
dhe-dss-with-3des-ede-cbc-sha	不可	DSA （DSS） 証明書	Ephemeral Diffie-Hellman
dhe-rsa-with-des-cbc-sha	不可	RSA 証明書	Ephemeral Diffie-Hellman キー交換

表 4-1　CSS でサポートされる SSL 暗号スイート（続き）

暗号スイート	エクスポートの可否	使用する認証証明書	使用されるキー交換アルゴリズム
dhe-rsa-with-3des-ede-cbc-sha	不可	RSA 証明書	Ephemeral Diffie-Hellman キー交換
dh-anon-with-rc4-128-md5	不可	どちらも使用しない	Diffie-Hellman
dh-anon-with-des-cbc-sha	不可	どちらも使用しない	Diffie-Hellman
dh-anon-with-3des-ede-cbc-sha	不可	どちらも使用しない	Diffie-Hellman
dhe-dss-with-rc4-128-sha	不可	DSA （DSS）証明書	Ephemeral Diffie-Hellman
rsa-export-with-rc4-40-md5	可	RSA 証明書	RSA キー交換
rsa-export-with-des40-cbc-sha	可	RSA 証明書	RSA キー交換
dhe-dss-export-with-des40-cbc-sha	可	DSA （DSS）証明書	Ephemeral Diffie-Hellman キー交換
dhe-rsa-export-with-des40-cbc-sha	可	RSA 証明書	Ephemeral Diffie-Hellman
dh-anon-export-with-rc4-40-md5	可	どちらも使用しない	Diffie-Hellman
dh-anon-export-with-des40-cbc-sha	可	どちらも使用しない	Diffie-Hellman
rsa-export1024-with-des-cbc-sha	可	RSA 証明書	RSA キー交換
dhe-dss-export1024-with-des-cbc-sha	可	DSA （DSS）証明書	Ephemeral Diffie-Hellman
rsa-export1024-with-rc4-56-sha	可	RSA 証明書	RSA キー交換
dhe-dss-export1024-with-rc4-56-sha	可	DSA （DSS）証明書	Ephemeral Diffie-Hellman

クライアント認証の設定

SSL プロキシ サーバを設定して、クライアントに証明書の送信を要求すれば、セキュリティをいっそう強化できます。デフォルトでは、クライアント証明書認証は無効です。クライアント認証を有効にすると、CSS は SSL ハンドシェイク中に、クライアントに証明書の交換を要求します。CSS は次の点を確認します。

- 証明書を送信するクライアントが対応するキーを持っているか
- 証明書が失効していないか
- シグニチャが有効であるか
- 発行元 CA がその証明書を取り消していないか

期限切れ、無効、または取り消されている証明書を CSS で処理する方法を設定できます。

ここでは、クライアント認証のための各種設定について説明します。

- クライアント認証の有効化
- クライアント証明書を検証するための CA 証明書の指定
- CRL レコードの設定
- CRL レコードの仮想 SSL サーバへの割り当て
- CSS による CRL レコードの強制ダウンロード
- クライアント認証失敗時の処理

クライアント認証の設定情報を表示するには、**show ssl-proxy-list ssl-server** コマンドを使用します。クライアント認証関連の活動の SSL カウンタを表示するには、**show ssl statistics** コマンドを使用します。詳細については、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。

クライアント認証の有効化

デフォルトでは、CSS のクライアント認証は無効に設定されています。クライアント認証の有効、無効は、**ssl-server** コマンドの **authentication** オプションによって指定できます。たとえば、クライアント認証を有効にするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication
enable
```

クライアント認証をデフォルトの無効状態に戻すには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 authentication
```

disable オプションを使用して、クライアント認証を無効に戻すこともできます。次に例を示します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication
disable
```

CSS のクライアント認証を有効にした後、CSS がクライアント証明書の検証に使用する CA 証明書を指定する必要があります。

クライアント証明書を検証するための CA 証明書の指定

CA 証明書には、CA の公開キーが含まれます。サーバに CA 公開キーがある場合、サーバはクライアント証明書がその CA により署名されていることを確認できます。CA 証明書を仮想 SSL サーバに割り当てると、CSS はこの CA 証明書に含まれるキーを使用して、クライアント証明書のデジタル署名を検証するようになります。

（注） SSL プロキシ リストをアクティブ化する前に、CA 証明書を設定する必要があります。

仮想 SSL サーバに証明書を設定するには、その前に CSS に CA 証明書をインポートして、その証明書をファイル名に関連付ける必要があります。CA 証明書のインポートについては、第 3 章「SSL 証明書とキーの設定」の「証明書と秘密キーのインポートまたはエクスポート」を参照してください。証明書とファイル名の関連付けについては、第 3 章「SSL 証明書とキーの設定」の「証明書とファイルとの関連付け」を参照してください。

少なくとも 1 つの証明書を設定する必要があり、最大 4 つの証明書を設定できます。5 つ以上の証明書を設定しようとすると、CSS でエラー メッセージが出力されます。

クライアント認証を有効にした後、**ssl-server** *number* **cacert** コマンドにより CA 証明書を仮想 SSL サーバに割り当てることができます。たとえば、仮想 SSL サーバに mycert1 CA 証明書アソシエーションを指定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cacert mycert1
```

仮想 SSL サーバから証明書アソシエーションを削除するには、**no** form of the **ssl-server** *number* **cacert** コマンドを使用します。たとえば、mycert1 CA 証明書アソシエーションを削除するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cacert mycert1
```

CRL レコードの設定

CA は証明書を取り消すと、その証明書を certificate revocation list（CRL）に追加し、CRL を公開します。CSS で CRL を使用するには、CRL を現在の場所から取得して、HTTP 経由でダウンロードする必要があります。この処理を実行するには、CSS に CRL レコードが必要です。CRL レコードには、CSS が CRL を現在の場所から定期的に取得してインポートするために必要な URL 情報が含まれます。CRL レコードを設定すると、そのレコードを仮想 SSL サーバに割り当てることができます。

（注）　CRL を取得するための HTTP 要求には、ソース IP アドレスとして仮想 SSL サーバの VIP アドレスを記述します。

CSS は最大 10 の CRL レコードを保存するように設定できますが、1 つの仮想 SSL サーバに割り当てることができる CRL レコードは 1 つだけです。CRL レコードを設定するには、**ssl crl-record** コマンドをグローバル設定モードで使用します。このコマンドのシンタックスは次のとおりです。

ssl crl-record *crl_name url sign_cert hours*

変数の内容は次のとおりです。

- *crl_name* : CRL レコード名。スペースを含まない 31 文字以内の文字列を入力します。
- *url* : CRL が置かれている URL。スペースを含まない 168 文字以内の文字列を入力します（http://www.example.com/crl/clientcert.crl など）。
- *sign_cert* : CRL に署名した CA 証明書名。CA 証明書によって CRL の正当性が証明されます。CRL を設定する前に、この証明書を CSS にインポートする必要があります。CA 証明書のインポートについては、第 3 章「SSL 証明書とキーの設定」の「証明書と秘密キーのインポートまたはエクスポート」を参照してください。証明書とファイル名の関連付けについては、第 3 章「SSL 証明書とキーの設定」の「証明書とファイルとの関連付け」を参照してください。
- *hours* : 更新された CRL を取得するまでの待ち時間。0 ～ 2000 の値を入力します。0 を指定すると CRL の取得が無効になり、CRL は更新されません。

CSS の SSL モジュールに、設定されたすべての CRL のリストが保存されます。このモジュールが CRL の取得を試みるのは、次の場合だけです。

- CRL レコードを含む SSL プロキシ リストがアクティブ化されている。
- その SSL プロキシ リストに関連付けられたサービスまたはコンテンツ ルールがアクティブ化されている。
- 前回の CRL の取得後、CRL レコード内で設定されている時間が経過した。

次の例は、mycrl という名前の CRL レコードを設定する方法を示しています。CRL の URL は crl.verisign.com です。CRL を認証する CSS 上の CA 証明書名は、verisign_cacert です。CSS は CRL を 24 時間ごとに更新します。次のように入力します。

```
(config)# ssl crl-record mycrl http://crl.verisign.com/class1.crl
verisign_cacert 24
```

CRL レコードを削除するには、次のように入力します。

```
(config)# no ssl crl-record mycrl
```

CRL の設定情報を表示するには、**show ssl crl-record** コマンドを使用します。このコマンドの詳細については、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。

CRL レコードの仮想 SSL サーバへの割り当て

CRL レコードを設定したら、そのレコードを仮想 SSL サーバに割り当てることができます。CRL レコードを仮想 SSL サーバに割り当てるには、**ssl-server** *number* **crl** コマンドを使用します。仮想 SSL サーバに割り当てられるのは、1 つの CRL レコードだけです。たとえば、mycrl CRL レコードを割り当てるには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 crl mycrl
```

mycrl CRL レコードを仮想 SSL サーバから削除するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 crl mycrl
```

CSS による CRL レコードの強制ダウンロード

すべての Certificate Revocation Lists （CRL; 証明書失効リスト）または ssl-accel サービス タイプで設定されたアクティブ サービスのアクティブな SSL プロキシ リストの特定の CRL を、CSS で強制的にダウンロードさせることができます。ダウンロードを強制的に実行するには、スーパーユーザ モードで **ssl force-crl-reload** コマンドを使用します。

特定の CRL をダウンロードするには、コマンド入力時にその名前を指定します。たとえば、次のように入力します。

```
# ssl force-crl-reload mycrl
```

コマンドで特定の CRL 名を指定しなかった場合は、CSS により、アクティブ サービスのアクティブな SSL プロキシ リストに関連付けられた設定済みの CRL がダウンロードされます。たとえば、次のように入力します。

```
# ssl force-crl-reload
```

バックアップ状態にある CSS の CRL はダウンロードできません。

クライアント認証失敗時の処理

クライアント証明書が無効、期限切れ、または CA により取り消されている場合、このクライアント証明書の認証に失敗します。デフォルトでは、CSS はクライアント証明書の認証に失敗すると、そのクライアントの接続を拒否します。

（注） CSS が CRL をダウンロードできない場合、クライアントの接続は失敗し、Revoked SSL アラートが使用されます。CRL が正常にロードされたことを確認するには、**show ssl statistics ssl** コマンドを使用します。

ssl-server *number* **failure** コマンドと次のオプションにより、CSS が認証に失敗したクライアント証明書をどのように処理するかを設定できます。

- **ignore**：CSS はクライアント認証の失敗を無視し、証明書が無効でも接続を許可します。たとえば、CSS がクライアント認証の失敗を無視するよう設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure ignore
```

（注）　**ignore** オプションは、セキュリティ リスクにつながるおそれがあります。

- **reject**：クライアント認証に失敗した場合の処理を、CSS のデフォルトの動作であるクライアント接続拒否に戻します。次に例を示します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure reject
```

- **redirect**：認証に失敗した接続を、設定した URL に送信します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure redirect
```

CSS がクライアント接続をリダイレクトする URL を設定するには、**ssl-server** *number* **failure-url** コマンドを使用します。スペースを含まない 168 文字以内の URL を指定します。たとえば、クライアント認証に失敗したときにクライアント接続を URL www.service_css.com にリダイレクトするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure-url
http://www.service_css.com
```

既存のリダイレクト URL を変更する場合は、**no ssl-server** *number* **failure-url** コマンドを使用して URL を削除し、**ssl-server** *number* **failure-url** コマンドを再発行して新しい URL を設定する必要があります。この変更を行う前に、アクティブな仮想 SSL サーバを一時停止してください。

たとえば、URL を削除するには次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 failure-url
```

（注）　失敗時の処理の設定に関係なく、クライアント認証の失敗は、常にエラー メッセージとして syslog に記録されます。

HTTP ヘッダー挿入の設定

SSL 接続では、クライアントに関する特定の情報を、バックエンド サーバに渡す必要が生じる場合もあります。HTTP ヘッダー挿入により、クライアントの接続中に情報を HTTP ヘッダーに埋め込むことができます。たとえば、クライアントが仮想 SSL サーバに接続しており、CSS でデータを復号化する場合、CSS は HTTP 要求のヘッダーにこの SSL セッションに関する情報と、クライアントおよびサーバ証明書を挿入してから、そのヘッダーをバックエンド サーバに渡すことができます。

（注）　HTTP 1.1 の固定接続の場合、HTTP ヘッダーの挿入は最初の HTTP 要求でだけ実行されます。同じ TCP 接続内のその後の要求は、変更されずに送信されます。HTTP 1.0 では固定接続が実装されていないため、すべての HTTP 要求にヘッダーが挿入されます。

CSS はクライアント データを復号化した後、次の情報のうち 1 つ以上を HTTP ヘッダーに挿入できます。

- クライアント証明書フィールドおよび関連情報
- サーバ証明書フィールドおよび関連情報
- SSL セッション フィールドおよび関連情報
- スタティックなテキスト文字列

次の操作も行うことができます。

- HTTP ヘッダーに挿入するクライアント証明書、サーバ証明書、または SSL セッション フィールドにプレフィックスを付ける。プレフィックスの設定は、スタティックなテキスト文字列の挿入には影響しません。
- クライアント証明書、サーバ証明書、または SSL セッションの HTTP ヘッダー フィールドを変更する。

ここでは、HTTP ヘッダーへの各種情報の挿入について説明します。

- クライアント証明書情報の挿入
- サーバ証明書情報の挿入
- セッション情報の挿入

- HTTP ヘッダーに挿入されたフィールドへのプレフィックスの追加
- スタティック テキスト文字列の挿入
- HTTP ヘッダー挿入フィールドの変更
- すべての HTTP 要求への HTTP ヘッダーの挿入

HTTP ヘッダー挿入の設定を表示するには、**show ssl-proxy-list** コマンドを使用します。このコマンドの詳細については、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。

クライアント証明書情報の挿入

クライアント証明書情報をバックエンド サーバに送信する必要がある場合は、クライアント証明書フィールドおよび関連情報を HTTP ヘッダーに挿入するように CSS を設定できます。フィールドにプレフィックスを付ける方法については、「HTTP ヘッダーに挿入されたフィールドへのプレフィックスの追加」を参照してください。

（注） SSL プロキシ リストがアクティブな場合は、HTTP ヘッダー挿入を設定または無効化する前にプロキシ リストを非アクティブにします。HTTP ヘッダー挿入の設定後、SSL プロキシ リストを再度アクティブ化します。

クライアント証明書フィールドを HTTP ヘッダーに挿入するよう CSS を設定するには、**ssl-server** *number* **http-header client-cert** コマンドを使用します。たとえば、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header
client-cert
```

クライアント証明書フィールドおよび情報の HTTP ヘッダーへの挿入を無効にするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
client-cert
```

表 4-2 に、挿入できるクライアント証明書フィールド、説明、形式、および例を示します。証明書の生成方法と、使用されたキー アルゴリズムによっては、これらのフィールドの一部が証明書に含まれない場合があります。

表 4-2　HTTP ヘッダーに挿入されるクライアント証明書フィールド

Client Certificate フィールド	説明、形式、および例
ClientCert-Fingerprint	説明：ハッシュ出力 形式：コロンで区切られた 16 進バイトの ASCII 文字列 例：ClientCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA
ClientCert-Subject-CN	説明：X.509 サブジェクトの共通名 形式：証明書を発行されるサブジェクトの共通名を表す文字列 例：ClientCert-Subject-CN: www.cisco.com
ClientCert-Issuer-CN	説明：X.509 証明書発行元の共通名 形式：証明書発行元の共通名を表す文字列 例：ClientCert-Issuer-CN: www.exampleca.com
ClientCert-Certificate-Version	説明：X.509 証明書のバージョン 形式：X.509 のバージョンの数値（3、2、または 1）に、X.509 バージョンを ASN.1 で表した値（2、1、または 0）をカッコで囲んで付加 例：ClientCert-Certificate-Version: 3 (0x2)
ClientCert-Serial-Number	説明：証明書のシリアル番号 形式：認証局で割り当てられた番号の任意の整数値 例：ClientCert-Serial-Number: 2
ClientCert-Data-Signature-Algorithm	説明：X.509 のハッシュおよび暗号化方式 形式：証明書の署名に使用されたアルゴリズム（md5WithRSAEncryption、sha1WithRSAEncryption、または dsaWithSHA1）、およびアルゴリズムのパラメータ 例：ClientCert-Signature-Algorithm: md5WithRSAEncryption

表 4-2　HTTP ヘッダーに挿入されるクライアント証明書フィールド（続き）

Client Certificate フィールド	説明、形式、および例
ClientCert-DSA-Public-Key-Size	説明：証明書に DSA キーが含まれる場合の DSA 公開キーのサイズ（ビット数） 形式：ビット数を示す整数値に、文字列 **bit** を付加 例：ClientCert-DSA-Public-Key-Size: 1024 bit
ClientCert-DSA-Public-Key	説明：DSA アルゴリズムの公開キー （証明書に DSA キーが含まれ場合だけ使用される） 形式：キーを表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列 例：ClientCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77
ClientCert-DSA-Private-Key-Size	説明：証明書に DSA キーが含まれる場合の DSA 秘密キーのサイズ（ビット数） 形式：ビット数を示す整数値に、文字列 **bit** を付加 例：ClientCert-DSA-Private-Key-Size: 1024 bit
ClientCert-Subject	説明：X.509 サブジェクトの識別名 形式：認証対象の秘密キーの所有者であるサブジェクトを表す文字列 例：ClientCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
ClientCert-Issuer	説明：X.509 証明書発行元の識別名 形式：この証明書を発行した認証局を表す文字列 例：ClientCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, 0=Root
ClientCert-Not-After	説明：証明書の失効日 形式：Validity フィールドの Not After にある協定世界時（UTC）または Generalized Time 形式の文字列 例：ClientCert-Not-After: 2003-1-27 23:59.59 UTC

表 4-2　HTTP ヘッダーに挿入されるクライアント証明書フィールド（続き）

Client Certificate フィールド	説明、形式、および例
ClientCert-Not-Before	説明：証明書の発効日 形式：Validity フィールドの Not Before にある協定世界時（UTC）または Generalized Time 形式の文字列 例：ClientCert-Not-Before: 2002-1-27 00:00:00.00 UTC
ClientCert-Public-Key-Algorithm	説明：公開キーに使用されるアルゴリズム 形式：証明書に含まれる公開キーの生成に使用されたアルゴリズム（rsaEncryption、rsa、または dsaEncryption） 例：ClientCert-Public-Key-Algorithm: rsaEncryption
ClientCert-RSA-Modulus-Size	説明：RSA 公開キーのサイズ 形式：RSA モジュラスの整数値のビット数（通常は 512、1024、または 2048）に、文字列 **bit** を付加 例：ClientCert-RSA-Modulus-Size: 1024 bit
ClientCert-RSA-Modulus	説明：RSA モジュラス 形式：RSA アルゴリズム モジュラス（n）を表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列。指数（e）とともに用いて、RSA 証明書の公開キー部分を表します。 例：ClientCert-RSA-Modulus: + 00:d8:1b:94:de:52:a1:20:51:b1:77
ClientCert-RSA-Exponent	説明：公開 RSA 指数 形式：RSA アルゴリズム指数（e）を表す整数値 例：ClientCert-RSA-Exponent: 65537
ClientCert-Subject-Key-Identifier	説明：X.509 サブジェクトのキー識別子 形式：X.509 バージョン 3 のサブジェクトのキー識別子を表す、コロンで区切られた 16 進バイトの ASCII 文字列 例：ClientCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99

表 4-2　HTTP ヘッダーに挿入されるクライアント証明書フィールド（続き）

Client Certificate フィールド	説明、形式、および例
ClientCert-Authority-Key-Identifier	説明：X.509 認証局キー識別子 形式：X.509 バージョン 3 の認証局キー識別子を表す、コロンで区切られた 16 進バイトの ASCII 文字列 例：ClientCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ClientCert-Basic-Constraints	説明：X.509 の基本的な制約 形式：証明書のサブジェクトが認証局としても機能できるかどうかを示す文字列。値は CA=TRUE または CA=FALSE 例：ClientCert-Basic-Constraints: CA=TRUE
ClientCert-Signature-Algorithm	説明：証明書署名アルゴリズム 形式：md5WithRSAEncryption、sha1WithRSAEncryption、または dsaWithSHA1（Secure Hash Algorithm の場合） 例：ClientCert-Signature-Algorithm: md5WithRSAEncryption
ClientCert-Signature	説明：証明書署名 形式：証明書内の他のフィールドのセキュア ハッシュ、およびこのハッシュのデジタル署名を表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列 例：ClientCert-Signature: 33:75:8e:a4:05:92:65

サーバ証明書情報の挿入

サーバ証明書情報をバックエンド サーバに送信する必要がある場合は、サーバ証明書フィールドおよび関連情報を挿入するように CSS を設定できます。サーバ証明書は CSS 上に存在し、**ssl-server** *number* **rsacert** または **ssl-server** *number* **dsacert** コマンドにより設定されます。フィールドにプレフィックスを付加する方法については、「HTTP ヘッダーに挿入されたフィールドへのプレフィックスの追加」を参照してください。

（注）　SSL プロキシ リストがアクティブな場合は、HTTP ヘッダー挿入を設定または無効化する前にプロキシ リストを非アクティブにします。HTTP ヘッダー挿入の設定後、SSL プロキシ リストを再度アクティブ化します。

サーバ証明書情報の挿入を設定するには、**ssl-server** *number* **http-header server-cert** コマンドを使用します。たとえば、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header
server-cert
```

サーバ証明書フィールドおよび情報の HTTP ヘッダーへの挿入を無効にするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
server-cert
```

表 4-3 に、挿入されるサーバ証明書の各フィールドと、それらについての説明を示します。証明書の生成方法と、使用されたキー アルゴリズムによっては、これらのフィールドの一部が証明書に含まれない場合があります。

表 4-3　HTTP ヘッダーに挿入されるサーバ証明書フィールド

フィールド	内容
ServerCert-Fingerprint	説明：ハッシュ出力 形式：コロンで区切られた 16 進バイトの ASCII 文字列 例：ServerCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA
ServerCert-Subject-CN	説明：X.509 サブジェクトの共通名 形式：証明書を発行されるサブジェクトの共通名を表す文字列 例：ServerCert-Subject-CN: www.cisco.com

表 4-3　HTTP ヘッダーに挿入されるサーバ証明書フィールド（続き）

フィールド	内容
ServerCert-Issuer-CN	説明：X.509 証明書発行元の共通名 形式：証明書発行元の共通名を表す文字列 例：ServerCert-Issuer-CN: www.exampleca.com
ServerCert-Certificate-Version	説明：X.509 証明書バージョン 形式：X.509 のバージョンの数値（3、2、または 1）に、X.509 バージョンを ASN.1 で表した値（2、1、または 0）をカッコで囲んで付加 例：ServerCert-Certificate-Version: 3 (0x2)
ServerCert-Serial-Number	説明：説明書シリアル番号 形式：認証局で任意に割り当てられた整数値 例：ServerCert-Serial-Number: 2
ServerCert-Data-Signature-Algorithm	説明：X.509 のハッシュおよび暗号化方式 形式：証明書に署名するのに使用する md5WithRSAEncryption、sha1WithRSAEncryption、または dsaWithSHA1 アルゴリズム、およびアルゴリズム パラメータ 例：ServerCert-Signature-Algorithm: md5WithRSAEncryption
ServerCert-DSA-Public-Key-Size	説明：証明書に DSA キーが含まれる場合の DSA 公開キーのサイズ（ビット数） 形式：ビット数を示す整数値に、文字列 **bit** を付加 例：ServerCert-DSA-Public-Key-Size: 1024 bit
ServerCert-DSA-Public-Key	説明：DSA アルゴリズムの公開キー（証明書に DSA キーが含まれる場合だけ使用される） 形式：キーを表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列 例：ServerCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77

表 4-3　HTTP ヘッダーに挿入されるサーバ証明書フィールド（続き）

フィールド	内容
ServerCert-DSA-Private-Key-Size	説明 : 証明書に DSA キーが含まれる場合の DSA 秘密キーのサイズ（ビット数） 形式：ビット数を示す整数値に、文字列 **bit** を付加 例：ServerCert-DSA-Private-Key-Size: 1024 bit
ServerCert-Subject	説明：X.509 サブジェクトの識別名 形式 : 認証対象の秘密キーの所有者であるサブジェクトを表す文字列 例：ServerCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
ServerCert-Issuer	説明：X.509 証明書発行元の共通名 形式：この証明書を発行した認証局を表す文字列 例：ServerCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, 0=Root
ServerCert-Not-After	説明：証明書の失効日 形式：Validity フィールドの Not After にある協定世界時（UTC）または Generalized Time の文字列 例：ServerCert-Not-After: 2003-1-27 23:59.59 UTC
ServerCert-Not-Before	説明：証明書の発効日 形式 : Validity フィールドの Not Before にある協定世界時（UTC）または Generalized Time 形式の文字列 例：ServerCert-Not-Before: 2002-1-27 00:00:00.00 UTC
ServerCert-Public-Key-Algorithm	説明：公開キーに使用されるアルゴリズム 形式 : 証明書に含まれる公開キーの生成に使用されるアルゴリズム（rsaEncryption、rsa、または dsaEncryption） 例：ServerCert-Public-Key-Algorithm: rsaEncryption

表 4-3　HTTP ヘッダーに挿入されるサーバ証明書フィールド（続き）

フィールド	内容
ServerCert-RSA-Modulus-Size	説明：RSA 公開キーのサイズ 形式：RSA モジュラスの整数値のビット数（通常は 512、1024、または 2048）に、文字列 **bit** を付加 例：ServerCert-RSA-Modulus-Size: 1024 bit
ServerCert-RSA-Modulus	説明：RSA モジュラス 形式：RSA アルゴリズム モジュラス（n）を表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列。指数（e）とともに用いて、RSA 証明書の公開キー部分を表します。 例：ServerCert-RSA-Modulus: + 00:d8:1b:94:de:52:a1:20:51:b1:77
ServerCert-RSA-Exponent	説明：RSA 公開指数 形式：RSA アルゴリズム指数（e）を表す整数値 例：ServerCert-RSA-Exponent: 65537
ServerCert-Subject-Key-Identifier	説明：X.509 サブジェクトのキー識別子 形式：X.509 バージョン 3 のサブジェクトのキー識別子を表す、コロンで区切られた 16 進バイトの ASCII 文字列 例：ServerCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ServerCert-Authority-Key-Identifier	説明：X.509 認証局キー識別子 形式：X.509 バージョン 3 の認証局キー識別子を表す、コロンで区切られた 16 進の ASCII 文字列 例：ServerCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99

表 4-3　HTTP ヘッダーに挿入されるサーバ証明書フィールド（続き）

フィールド	内容
ServerCert-Basic-Constraints	説明：X.509 の基本的な制約 形式：証明書のサブジェクトが認証局としても機能できるかどうかを示す文字列。値は CA=TRUE または CA=FALSE 例：ServerCert-Basic-Constraints: CA=TRUE
ServerCert-Signature-Algorithm	説明：証明書署名アルゴリズム 形式：md5WithRSAEncryption、sha1WithRSAEncryption、または dsaWithSHA1（Secure Hash Algorithm の場合） 例：ServerCert-Signature-Algorithm: md5WithRSAEncryption
ServerCert-Signature	説明：証明書署名 形式：証明書内の他のフィールドのセキュア ハッシュ、およびこのハッシュのデジタル署名を表す、コロン（:）で区切られたビッグエンディアン形式の 16 進数（先頭に 0x を付けない）の小文字の英数字文字列 例：ServerCert-Signature: 33:75:8e:a4:05:92:65

セッション情報の挿入

SSL セッション情報をバックエンド サーバに送信する場合は、SSL セッション フィールドおよび関連情報を挿入するように CSS を設定できます。フィールドにプレフィックスを付ける方法については、「HTTP ヘッダーに挿入されたフィールドへのプレフィックスの追加」を参照してください。

（注） SSL プロキシ リストがアクティブな場合は、HTTP ヘッダー挿入を設定または無効化する前にプロキシ リストを非アクティブにします。HTTP ヘッダー挿入の設定後、SSL プロキシ リストを再度アクティブ化します。

セッション情報の挿入を設定するには、**ssl-server** *number* **http-header session** コマンドを使用します。たとえば、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header session
```

SSL セッション フィールドおよび情報の HTTP ヘッダーへの挿入を無効にするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
session
```

表 4-4 に、挿入される SSL セッションの各フィールドと、それらについての説明を示します。

表 4-4　HTTP ヘッダーに挿入される SSL セッション フィールド

フィールド	内容
Session-Cipher-Name	説明：対称暗号スイート 形式：このセッション中にネゴシエートされる暗号スイートの OpenSSL バージョン名 例：Session-Cipher-Name: EXP1024-RC4-SHA
Session-Cipher-Key-Size	説明：対称暗号キー サイズ 形式：公開キーの長さ（バイト数）を表す整数 例：Session-Cipher-Key-Size: 128
Session-Cipher-Use-Size	説明：対称暗号の使用 形式：このセッション中に対称暗号に使用されるキーの長さ（バイト数）を表す整数 例：Session-Cipher-Use-Size: 56
Session-Protocol-Version	説明：SSL または TLS のバージョン 形式:SSL または TLS プロトコルにバージョン番号を付加した文字列 例：Session-Protocol-Version: TLSv1

表 4-4　HTTP ヘッダーに挿入される SSL セッション フィールド（続き）

フィールド	内容
Session-Id	説明：SSL セッション ID 形式：このセッションでネゴシエートされた、またはネゴシエートされている 32 バイトのセッション ID を表す小文字の英数字文字列（先頭に 0x を付けず、コロン（:）で区切られたビッグエンディアン形式の 16 進数） 例：Session-Id: 75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25:75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25
Session-Verify-Result	説明：SSL セッションの検証結果 形式：SSL セッションの検証結果を示す数値 例：Session-Verify-Result: 0

HTTP ヘッダーに挿入されたフィールドへのプレフィックスの追加

HTTP ヘッダーに挿入されたクライアント証明書、サーバ証明書、またはセッション フィールドにプレフィックスを付けるには、**ssl-server** *number* **http-header prefix** コマンドを使用します。16 文字以内のテキスト文字列を引用符で囲んで入力します。

（注）　SSL プロキシ リストがアクティブな場合は、HTTP ヘッダー挿入を設定または無効化する前にプロキシ リストを非アクティブにします。HTTP ヘッダー挿入の設定後、SSL プロキシ リストを再度アクティブ化します。

たとえば、挿入されたすべてのフィールドにプレフィックス「Acme-SSL」を追加するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header prefix
"Acme-SSL"
```

これで ClientCert-Certificate-Version フィールドが Acme-SSL-ClientCert-Certificate-Version として表示されます。

プレフィックスを含まないデフォルトに戻すには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
prefix
```

（注）　**ssl-server** *number* **http-header prefix** コマンドは、HTTP ヘッダーに挿入するように設定した SSL フィールドだけに影響を与えます。このコマンドは、スタティック テキスト文字列の挿入には影響を与えません。

スタティック テキスト文字列の挿入

バックエンド サーバに送信される HTTP ヘッダーにテキスト文字列を挿入するのは、Microsoft Outlook Web Access（OWA）アプリケーションへの対応が主な理由ですが、別の理由でスタティック テキストを挿入することもできます。スタティック テキスト文字列の挿入を設定するには、**ssl-server** *number* **http-header static** コマンドを使用します。スペースを含む 199 文字以内のテキスト文字列を引用符で囲んで入力します。OWA をサポートするには、テキスト文字列「FRONT-END-HTTPS: on」を入力します。

（注）　SSL プロキシ リストがアクティブな場合は、HTTP ヘッダー挿入を設定または無効化する前にプロキシ リストを非アクティブにします。HTTP ヘッダー挿入の設定後、SSL プロキシ リストを再度アクティブ化します。

たとえば、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static
"FRONT-END-HTTPS: on"
```

\r\n 文字列を使用することによって、その前後の文字列を改行して挿入することもできます。行を終了する \r\n 文字列で、199 文字のうちの 4 文字を使用します。次の例では、3 つの文字列「FRONT-END-HTTPS: on」、「session cache: on」、および「vip address: www.acme.com」の挿入しています。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static
"FRONT-END-HTTPS: on\r\nsession cache: on\r\nvipaddress: www.acme.com"
```

HTTP ヘッダーへのスタティック文字列の挿入を無効にし、その文字列を削除するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
static
```

HTTP ヘッダー挿入フィールドの変更

挿入する HTTP ヘッダーのフィールドに標準はなく、アプリケーションやアプリケーションの実装方法によって異なります。このことから、CSS では、セッションへ挿入する HTTP ヘッダー タグの値を編集することができます。

（注） HTTP ヘッダー挿入機能は、HTTP ヘッダー フィールドが変更された場合にも対応している必要があります。CSS で HTTP ヘッダー挿入を有効にする方法については 「クライアント証明書情報の挿入」、「サーバ証明書情報の挿入」、または「セッション情報の挿入」を参照してください。

変更するには、次のようにします。

- クライアント証明書フィールドの場合、**ssl-server** *number* **http-header client-cert-field** コマンドを使用する。
- サーバ証明書フィールドの場合、**ssl-server** *number* **http-header server-cert-field** コマンドを使用する。
- セッション フィールドの場合、**ssl-server** *number* **http-header session-field** コマンドを使用する。

これらのコマンドのシンタックスは次のとおりです。

ssl-server *number* [**http-header client-cert-field**|**server-cert-field**| **session-field**] *default_field* **"***configured_field***"**

変数の内容は次のとおりです。

- *default_field* : 変更対象のクライアント証明書、サーバ証明書、またはセッション フィールドのデフォルト名。
  - デフォルトのクライアント証明書フィールドについては、表 4-2 を参照するか（「ClientCert-」プレフィックスは含みません）または次のように入力します。

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header
    client-cert-field ?
    ```

  - デフォルトのサーバ証明書フィールドについては、表 4-3 を参照するか（「ServerCert-」プレフィックスは含みません）または次のように入力します。

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header
    server-cert-field ?
    ```

  - デフォルトのセッションフィールドについては、表 4-4 を参照するか（「Session-」のプレフィックスは含みません）または次のように入力します。

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header session-field
    ?
    ```

- *configured_field* : デフォルトのフィールド名を置き換えて設定する名前。設定対象フィールドに 1 ～ 36 文字のテキスト文字列を引用符で囲んで入力します。テキスト文字列の末尾にコロン（:）を入力しないでください。コロンは CSS によって自動挿入されます。

  テキスト文字列を引用符で囲まずに入力すると、デフォルト値がフィールドに設定されます。

たとえば、**ClientCert-Fingerprint** クライアント証明書フィールドを **Client-Fingerprint** に変更するには、次のように入力します。

```
(config-ssl-proxy-list)# ssl-server 1 http-header client-cert-field
Fingerprint "Client-Fingerprint"
```

ServerCert-Fingerprint サーバ証明書フィールドを **Server-Fingerprint** に変更するには、次のように入力します。

```
(config-ssl-proxy-list)# ssl-server 1 http-header server-cert-field
Fingerprint "Server-Fingerprint"
```

Session-Cipher-Name セッションフィールド名を **CipherName** に変更するには、次のように入力します。

```
(config-ssl-proxy-list)# ssl-server 1 http-header session-field
Cipher-Name "CipherName"
```

クライアント証明書、サーバ証明書、またはセッション フィールドをデフォルト値に戻すには、次のコマンドを使用します。

no ssl-server *number* [**http-header client-cert-field**|**server-cert-field**|**session-field**] *default_field*

default_field 変数は、変更対象のクライアント証明書、サーバ証明書、またはセッション フィールドのデフォルト名です。デフォルトの HTTP ヘッダー挿入フィールドおよび関連付けられた設定値を表示するには、**show ssl-proxy-list** *list_name* コマンドを使用します。このコマンドの詳細については、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。デフォルトのフィールドは、引用符で囲まずに入力します。

（注） クライアント証明書、サーバ証明書、およびセッションフィールドにはプレフィックスを含めないでください。たとえば、**ClientCert-Fingerprint** フィールドに対しては Fingerprint と入力します。

また、デフォルトの **ClientCert-Fingerprint** クライアント証明書フィールドをリセットするには、次のように入力します。

```
[(config-ssl-proxy-list)# no ssl-server 1 http-header
client-cert-field Fingerprint
```

デフォルトの **ServerCert-Fingerprint** サーバ証明書フィールドをリセットするには、次のように入力します。

```
(config-ssl-proxy-list)# no ssl-server 1 http-header server-cert-field
Fingerprint
```

デフォルトの **Session-Cipher-Name** セッション フィールドをリセットするには、次のように入力します。

```
(config-ssl-proxy-list)# no ssl-server 1 http-header session-field
Cipher-Name
```

すべての HTTP 要求への HTTP ヘッダーの挿入

デフォルトでは HTTP 固定接続の場合、HTTP ヘッダーの挿入は最初の HTTP 要求でだけ実行されます。同じ TCP 接続内のその後の要求は、変更されずに送信されます。同じ TCP 接続内のすべての HTTP 要求に HTTP ヘッダーを挿入するには、**ssl-server** *number* **http-header insert-per-request** コマンドを使用します。

たとえば、次のように入力します。

```
(config-ssl-proxy-list)# ssl-server 1 http-header insert-per-request
```

HTTP 固定接続の最初の HTTP 要求でだけ、HTTP ヘッダー挿入のデフォルトの動作をリセットするには、次のように入力します。

```
(config-ssl-proxy-list)# no ssl-server 1 http-header
insert-per-request
```

SSL または TLS バージョンの指定

デフォルトの SSL バージョンは SSL バージョン 3、TLS バージョン 1 です。SSL モジュールは、SSL バージョン 3 のヘッダー、TLS バージョン 1 のメッセージを含む ClientHello を送信します。

ssl-server *number* **version** *protocol* コマンドを使用して、SSL または Transport Layer Security （TLS）プロトコルのバージョンを指定します。オプションは、次のとおりです。

- **ssl-tls** ：SSL プロトコル バージョン 3.0 と TLS プロトコル バージョン 1.0（デフォルト）
- **ssl** ：SSL プロトコル バージョン 3.0
- **tls**：TLS プロトコル バージョン 1.0

たとえば、SSL バージョン 3.0 を指定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 version ssl
```

SSL バージョンをデフォルトの SSL バージョン 3.0 と TLS バージョン 1.0 にリセットするには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 version
```

TCP FIN メッセージ単独でのクライアント接続の終了

クライアント接続は通常、SSL の Close-Notify アラートによってエラーなく終了します。ただし、MSIE ブラウザのいくつかのバージョンでは、Close-Notify アラートを受信しても接続を終了しません。CSS 側で接続が終了したものと認識しても、ブラウザがその接続を再利用しようとする場合があります。この接続では CSS が新しい要求に応答しないため、ブラウザはエラーを表示します。

ssl-server コマンドの **unclean-shutdown** オプションを使用すれば、CSS は TCP FIN メッセージだけを送信してクライアント接続を終了できます。この場合、CSS は Close-Notify アラートを送信しません。

TCP FIN メッセージだけを送信してクライアント接続を終了させるように CSS を設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 unclean-shutdown
```

このコマンドに **no** をつけると、CSS のデフォルトの動作（Close-Notify アラートと TCP FIN メッセージの両方を送信してクライアント接続を終了させる）に戻ります。次に例を示します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 unclean-shutdown
```

セキュア URL リライトの指定

クライアントの HTTPS 接続は、SSL プロキシ リストで定義した仮想 SSL サーバを介してバックエンド サーバに送信されるときに、HTTP 接続になることがあります。バックエンド サーバは、その HTTP 接続で、クライアントからのクリア テキスト データを受信します。このサーバが、別の HTTP URL に向けて HTTP 300 シリーズ リダイレクトを行う場合、クライアントが、HTTPS 要求を行っていた場合でも、HTTP 要求が行われるようになります。クライアントの接続が HTTP に変わるため、要求のデータをクリア テキスト接続を使用してサーバから入手できない場合があります。

（注） SSL プロキシ リストで 1 つ以上のバックエンド SSL サーバを定義する場合は、仮想 SSL サーバの設定パラメータとしてセキュア URL リライトを指定しないでください（「SSL TCP 接続の Nagle アルゴリズムの指定」参照）。

1 つ以上の URL リライト ルールを設定することで、バックエンド サーバからの保護されていない HTTP リダイレクトを未然に防止できます。それぞれのリライト ルールは、SSL プロキシ リストで仮想 SSL サーバに関連付けられます。URL リライト ルールでは、保護されていない HTTP URL にユーザをリダイレクトする Web サイトの問題を、そのドメインを http:// から https:// に書き換えることで解決します。URL リライトを使用することで Web サーバへのすべてのクライアント接続は SSL となり、HTTPS コンテンツの配信が安全にクライアントに戻されます。

仮想 SSL サーバに URL リライト ルールを追加して、保護されていない HTTP 300 シリーズのリダイレクトを防止するには、**ssl-server** *number* **urlrewrite** コマンドを使用します。このコマンドにより CSS では、SSL モジュールを介してサーバから受信したすべての HTTP ヘッダー フィールドが調べられ、300 シリーズのリダイレクト応答（302 Found や 304 Not Modified など）がないかどうか確認されます。300 シリーズの戻りコードが見つかると、HTTP ヘッダーの Location Response-Header フィールドに URL リライト ルールで定義されているホスト名と一致するものがないかが調べられます。見つかると、Location フィールドが書き換えられ、応答の HTTPS の場所と SSL ポートが含まれるようになります。

たとえば、SSL ポートをポート 443、クリア テキスト ポートをポート 80 とそれぞれデフォルト設定のままにして、URL リライト ルールを定義するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22
www.website.com
```

この場合、http://www.website.com/ へのすべての HTTP リダイレクトは、SSL モジュールで https://www.website.com/ として書き換えられ、クライアントに転送されます。

CSS では、URL リダイレクト ルールのドメインのホスト名の一致基準の一部としてワイルドカードの使用をサポートしています。ドメイン名にワイルドカード アスタリスク（*）を使用すると、1 つのドメイン内の複数のホストが割り出されます。ワイルドカードだけのホスト名（たとえば、*）、プリフィックス ワイルドカード（たとえば、*.mydomain.com）、またはサフィックス ワイルドカード（たとえば、www.mydomain.*）を指定できます。ワイルドカードだけのホスト名を使用すると、ドメイン名全体が * （アスタリスク）となり、サーバからこの VIP アドレスを介して到着するすべての HTTP リダイレクトが HTTPS に書き換えられます。この場合、その SSL サーバにこれ以上 URL リライト ルールは必要ありません。

（注） SSL モジュールによってすべての URL 参照が予期せずしてリライトされないようにするために、ワイルドカードを使用するときは注意が必要です。リダイレクトを確認し、指定したワイルドカード ルールに一致するすべての URL を書き換えする必要があるか確認してください。

CSS では、次の順序で URL リライト検索を実行します。

1. 完全一致
2. 最短プレフィクスを使用したポストフィックス　ワイルドカード一致（たとえば、「ssl-server 1 urlrewrite 12 cisco.*」に一致する前に「ssl-server 1 urlrewrite 7 cis*」に一致）
3. 最短一致を使用したプレフィクス ワイルドカード一致（たとえば、「ssl-server 1 urlrewrite 12 *.cisco」に一致する前に「ssl-server 1 urlrewrite 7 *.cis」に一致）

4. ワイルドカード一致（たとえば ssl-server 1 urlrewrite 7 *）

ssl-server *number* **urlrewrite** コマンドのシンタックスは次のとおりです。

ssl-server *number* **urlrewrite** *number hostname* [**sslport** *port* {**clearport** *port*}]

このコマンドのオプションと変数は次のとおりです。

- **ssl-server** *number* ：SSL プロキシ リストで仮想 SSL サーバを識別するために使用する番号
- **urlrewrite** *number* ：仮想 SSL サーバに追加する URL リライト ルールの数。URL リライト ルールの数に対応する 1 ～ 32 の値を入力します。32 までの URL リライト ルールを追加して、HTTP から HTTPS へのリダイレクトを処理できます。
- *hostname*：リダイレクトする URL のドメイン名（たとえば、www.mydomain.com）。URL リライト ホストのドメイン名に対応する 240 文字までの文字列を引用符で囲まずに入力します。ホスト名にはディレクトリ パスは含めないでください。ドメイン名にワイルドカードを指定して 1 つのドメイン内の複数のホストを割り出すには、アスタリスク（*）ワイルドカードを使用します。
- **sslport** *port*：（省略可）SSL ネットワーク トラフィックに使用するポート。SSL コンテンツ ルールに一致する TCP ポート番号（SSL コンテンツ ルールで使用される TCP ポート番号）を入力します。SSL モジュールは、URL リダイレクト ルールに一致した HTTP リダイレクトを、指定した SSL ポート（またはポート番号を指定していない場合はデフォルトのポート 443）に書き換えます。1 ～ 65535 の値を入力します。デフォルト値は 443 です。
- **clearport** *port*：（省略可）クリア テキストのネットワーク トラフィックに使用するポート。SSL モジュールは、Location Response-Header フィールドのリダイレクトを、指定したクリア テキスト ポート（またはポート番号を指定していない場合はポート 80）と照合します。1 ～ 65535 の値を入力します。デフォルト値は 80 です。

たとえば、SSL トラフィックにポート 444、クリア テキストに 81 を使用するように、*www.mydomain.com* の URL リライト ルール 22 を指定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22
www.mydomain.com sslport 444 clearport 81
```

URL リライト ルール 22 を削除するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 urlrewrite 22
```

たとえば、次のようにワイルドカード アスタリスク文字（*）を使用して、HTTP URL の www.sales.acme.com および www.services.acme.com に一致させることができます（SSL ポートはデフォルトのポート 443、クリア テキスト ポートはデフォルトのポート 80 のまま）。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1
*.acme.com
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2
*.acme.com
```

または、HTTP URL の www.acmesales.com および www.acmeservices.com のワイルドカード アスタリスク文字（*）を次のように使用することもできます。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1
www.acme*
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2
www.acme*
```

SSL URL リライトに関する統計情報を表示するには、第 7 章「SSL の設定情報および統計情報の表示」を参照してください。

SSL セッション キャッシュ タイムアウトの指定

SSL では、クライアントと CSS SSL モジュールが完全なキー交換を完了し、新しいマスター秘密キーが確立されるたびに、新しいセッション ID が作成されます。SSL セッション キャッシュ タイムアウトを指定すると、SSL モジュールは次回以降のクライアントとの接続でマスター キーを再使用できるようになり、SSL ネゴシエーション プロセスを短縮できます。タイムアウト値を指定することによって、1 つの SSL セッション ID の有効性が持続する合計時間を設定できます。この時間が経過すると、SSL モジュールは完全な SSL ハンドシェイクを要求するようになり、新しい SSL セッションが確立されます。

SSL セッション キャッシュのタイムアウト値は、レイヤ 5 コンテンツ ルールで **advanced-balance ssl** ロード バランシング方式を使用する場合に、クライアントの接続先サーバの固定に使用される SSL セッション ID を適切に調整させるための重要な要素となるので、慎重に設定する必要があります。

ssl-server *number* **session-cache** *seconds* コマンドを使用して、以前設定した秘密キーでクライアントとの接続を再開できるように SSL モジュールを設定します。SSL セッション キャッシュ タイムアウト値（秒単位）には、0 （SSL セッション ID の再使用は無効）～ 72000（20 時間）を入力します。デフォルトは 300（5 分）です。このオプションを無効にする（値を 0 に設定する）と、クライアントと SSL モジュール間で新しい接続を確立するたびに完全な SSL ハンドシェイクが行われるようになります。

（注）　**ssl-server** *number* **session-cache** *seconds* コマンドに 0 を指定することはお勧めしません。0 以外の値を入力すると、SSL セッション ID が再使用され CSS のパフォーマンスが向上します。

たとえば、クライアントとの接続で SSL セッション ID が再使用されるように、タイムアウトを 10 時間に設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 session-cache 36000
```

SSL セッション再使用のタイムアウト値をデフォルトの 300 秒にリセットするには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 session-cache
```

SSL セッションのハンドシェイク再ネゴシエーションの指定

SSL セッション ハンドシェイク コマンドを実行すると、SSL HelloRequest メッセージがクライアントへ送信され、SSL ハンドシェイク ネゴシエーションが再開されます。SSL 再ハンドシェイクは、長時間にわたって接続を維持しているときに、SSL セッションを再確立してセキュリティを保証する手段として役立ちます。

CSS とクライアント間で交換できるデータの最大量を指定するには、**ssl-server** *number* **handshake data** *kbytes* コマンドを使用します。ここで指定した量のデータが送られると、CSS は SSL ハンドシェイク メッセージを送信し、SSL セッションが再確立されます。data 値を設定すると、指定したデータ量の送信後に SSL

セッションは新しいセッション キーを再取り決めします。SSL ハンドシェイクの data 値（KB 単位）には、0 （ハンドシェイク無効）～ 512000 を指定します。デフォルトは 0 です。

たとえば、クライアントとのデータ交換が 125000 KB に達した後に、SSL プロキシ リストの SSL 再ハンドシェイク メッセージを送信するように設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake data
125000
```

再ハンドシェイク データ オプションを無効にするには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake data
```

最大タイムアウト値を指定するには、**ssl-server** *number* **handshake timeout** *seconds* コマンドを使用します。このタイムアウト値が経過すると、CSS は SSL ハンドシェイク メッセージを送信して SSL セッションを再確立します。タイムアウト値を設定すれば、指定した秒数の経過後に SSL セッションは新しいセッション キーを再取り決めするようになります。SSL 再ハンドシェイクのタイムアウト値は、レイヤ 5 コンテンツ ルールで **advanced-balance ssl** ロード バランシング方式を使用する場合に、クライアントの接続先サーバの固定に使用される SSL セッション ID を適切に調整させるための重要な要素となるので、慎重に設定する必要があります。SSL ハンドシェイクのタイムアウト値（秒単位）には、0 （ハンドシェイクは無効）～ 72000 （20 時間）を指定します。デフォルトは 0 です。

たとえば、タイムアウト値の 10 時間が経過すると SSL 再ハンドシェイク メッセージが送信されるように設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake timeout
36000
```

再ハンドシェイク タイムアウト オプションを無効にするには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake timeout
```

（注） SSL スティッキを使用して接続が固定される設定の場合、CSS がハンドシェイク ネゴシエーションを実行するたびに SSL セッション ID が既存の TCP フロー内で再生成され、SSL スティッキ性が失われるため注意が必要です。このため、CSS は新しい SSL セッション ID を認識しません。この SSL フローで次の TCP 接続を検出すると、CSS はこの接続を新しい SSL セッションと見なし、別の SSL サービスに負荷を分散します。複数のサービスと複数の SSL モジュールが存在する場合、CSS はこの接続を別の SSL モジュールに送信する可能性があります。この接続は送信先の SSL モジュールでは新しい SSL セッションになるため、2 回目の再ネゴシエーションが行われます。2 回目の再ネゴシエーション後、CSS ではその SSL セッション ID を認識し、この SSL セッションはこのもう一方の SSL モジュールに固定されます。

この場合、SSL 再ハンドシェイクを有効にしていると、ハンドシェイクの再ネゴシエーションのために、さらにリソースが消費されます。トラフィック量の多い環境で運用している場合、この状況は SSL パフォーマンス全体に影響する可能性があります。

SSL キュー データの遅延時間の設定

CSS の SSL は、サーバからのパケット データをキューに置き、そのデータを暗号化してクライアントに送信します。SSL は、次の場合にキューのデータを空にし、暗号化してクライアントに送信します。

- キューの容量が 16,400 バイトに達した場合（SSL の最大レコード サイズ）
- サーバから TCP FIN パケットが送信された場合
- キューの容量が 16,400 バイト未満でも、CSS に設定された遅延時間が経過した場合

SSL は、効率化のために、最大サイズが 16,400 バイトの SSL レコードにデータを暗号化します。暗号化用のキューを 16,400 バイトまで満たすと、SSL はキュー データを空にして暗号化するのに時間がかかります。

キュー データを空にして暗号化するまでの CSS 仮想 SSL サーバの待機時間を設定するには、**ssl-server** *number* **ssl-queue-delay** *ms* コマンドを使用します。デフォルト値は 200 ミリ秒です。遅延時間には 0 （無効）～ 10000 の値を入力します。

遅延値を 0 に設定すると、データのキューイングが無効になります。この場合、CSS の仮想 SSL サーバは、サーバからデータが届くとすぐに暗号化して、そのデータをクライアントに送信します。

たとえば、遅延時間の値を 400 ミリ秒に設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 ssl-queue-delay 400
```

遅延時間をデフォルトの 200 ミリ秒にリセットするには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 ssl-queue-delay
```

SSL TCP クライアント側の接続タイムアウト値の指定

CSS とクライアント間の TCP 接続は、指定した時間が経過すると終了します。この TCP タイムアウト機能を使用すると、CSS SSL モジュールとクライアント間の TCP 接続を、より柔軟に管理できます。

SSL プロキシ リストで仮想 SSL サーバを、クライアントとの TCP 接続を終了するように設定する方法については、次の項を参照してください。

- TCP SYN タイムアウト値の指定（クライアント側接続）
- TCP 無活動タイムアウト値の指定（クライアント側接続）

TCP SYN タイムアウト値の指定（クライアント側接続）

CSS の SYN タイマーは、TCP 3 ウェイ ハンドシェイクを終了する手段として、CSS による SYN/ACK の送信とクライアントによる ACK の応答の間の時間差をカウントします。このタイムアウト値は、クライアントと CSS モジュール間の TCP 接続で TCP 3 ウェイ ハンドシェイクが正常に完了しなかったときに、その接続をデータ転送前に終了させるために使用し、**ssl-server** *number* **tcp virtual syn-timeout** *seconds* コマンドを使用して指定します。

TCP SYN の無活動タイムアウト値（秒単位）には、1 ～ 3600 （1 時間）を入力します。デフォルトでは 30 秒に設定されています。

（注）　接続タイマーは、新しい SSL 接続と TCP 接続のどちらについても、常に再送信終了時間より短く設定する必要があります。

たとえば、TCP SYN のタイムアウトを 30 分（1800 秒）に設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
syn-timeout 1800
```

TCP SYN タイムアウトの値をデフォルトの 30 秒に戻すには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
syn-timeout
```

TCP 無活動タイムアウト値の指定（クライアント側接続）

TCP 無活動タイムアウトのカウントは、ACK を CSS がクライアントから受信したときに開始され、TCP 3 ウェイ ハンドシェイクを終了するまで行われます。この無活動タイマーは、そのトラフィック フローの SYN タイマーが停止した時点で再開されます。**ssl-server** *number* **tcp virtual inactivity-timeout** *seconds* コマンドを使用して、クライアントとの間でほとんどあるいはまったく活動していない TCP 接続を終了するために使用するタイムアウト値を指定します。

TCP 無活動タイムアウト値（秒単位）には、0 （TCP 無活動タイムアウトは無効）～ 3600 （1 時間）の値を入力します。デフォルトは 240 秒です。

たとえば、TCP 無活動タイマーを 30 分（1800 秒）に設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
inactivity-timeout 1800
```

TCP 無活動タイマーの値をデフォルトの 240 秒に戻すには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
inactivity-timeout
```

SSL TCP サーバ側接続タイムアウト値の指定

CSS とサーバ間の TCP 接続は、指定した時間が経過すると終了します。この TCP タイムアウト機能を使用すると、CSS SSL モジュールとサーバ間の TCP 接続を、より柔軟に制御できます。

SSL プロキシ リストで仮想 SSL サーバを、サーバとの TCP 接続を終了するように設定する方法については、次の項を参照してください。

- TCP SYN タイムアウト値の指定（サーバ側接続）
- TCP 無活動タイムアウト値の指定（サーバ側接続）

TCP SYN タイムアウト値の指定（サーバ側接続）

TCP SYN タイマーは、CSS が SYN を送信してバックエンドの TCP 接続を開始したタイミングと、サーバが SYN/ACK で応答したタイミングの時間差をカウントします。サーバとの TCP 接続で TCP 3 ウェイ ハンドシェイクが正常に完了しなかったときに、その接続をデータ転送前に終了させるために使用されるこのタイムアウト値は、**ssl-server** *number* **tcp server syn-timeout** *seconds* コマンドを使用して指定します。

TCP SYN のタイムアウト値（秒単位）には、1 ～ 3600 （1 時間）を入力します。デフォルトでは 30 秒に設定されています。

（注） 接続タイマーは、新しい SSL 接続と TCP 接続のどちらについても、常に再送信終了時間より短く設定する必要があります。

たとえば、TCP SYN のタイムアウトを 30 分（1800 秒）に設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server
syn-timeout 1800
```

TCP SYN タイムアウトの値をデフォルトの 30 秒に戻すには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
syn-timeout
```

TCP 無活動タイムアウト値の指定（サーバ側接続）

TCP 無活動タイムアウトのカウントは、CSS がサーバから SYN/ACK を受信した時点から開始されます。この無活動タイマーは、そのトラフィック フローの SYN タイマーが停止した時点で再開されます。サーバとの TCP 接続がほとんど、またはまったく活動していないときに、その接続を終了させるために使用されるタイムアウト値を、**ssl-server** *number* **tcp server inactivity-timeout** *seconds* コマンドを使用して指定します。

TCP 無活動タイムアウト値（秒単位）には、0 （TCP 無活動タイムアウトは無効）～ 3600（1 時間）の値を入力します。デフォルトは 240 秒に設定されています。

たとえば、TCP 無活動タイマーを 30 分（1800 秒）に設定するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server
inactivity-timeout 1800
```

TCP 無活動タイマーの値をデフォルトの 240 秒に戻すには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
inactivity-timeout
```

SSL TCP 接続における確認応答遅延の変更

クライアントまたはサーバ接続におけるデフォルトの確認応答遅延時間は 200 ミリ秒（Ms）です。次のコマンドを実行することで、確認応答遅延の SSL TCP タイマーの長さを無効にしたり調整したりできます。

ssl-server *server-num* **tcp virtual|server ack-delay** *value*

value 変数は、確認応答遅延のタイマーの長さをミリ秒（Ms）で指定します。デフォルト値は 200 です。0 ～ 10000 の値を入力します。0 を指定すると、クライアントから SSL トラフィックを受信する際の確認応答遅延は無効になります。タイマーを無効にすると、SSL セッション キャッシュ（セッション ID の再使用）の使用によりセッションのパフォーマンスが向上します。

たとえば、クライアントと SSL モジュールとの間の TCP 接続に 400 ミリ秒の確認応答遅延を設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
ack-delay 400
```

サーバと SSL モジュールとの間の TCP 接続に 400 ミリ秒の確認応答遅延を設定するには、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server ack-delay
400
```

SSL TCP 接続の Nagle アルゴリズムの指定

TCP Nagle アルゴリズムでは、クライアントと SSL モジュール間またはサーバと SSL モジュール間の TCP 接続で転送されるサイズの小さい多数のバッファ メッセージを自動的に連結します。この処理で個々の TCP 接続で送信されるパケット数が減少し、CSS のスループットが向上します。ただし、Nagle アルゴリズムと TCP 遅延確認応答の間の相互作用によっては、TCP 接続の遅延時間が長くなることもあります。TCP 接続（クリア テキストまたは SSL）に許容範囲を超える遅延が発生するようであれば、Nagle アルゴリズムを無効にしてください。

- クライアントと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを無効にする、または再度有効にするには、**ssl-server** *number* **tcp virtual nagle** コマンドを使用します。このコマンドのシンタックスは次のとおりです。

  ssl-server *number* **tcp virtual nagle enable|disable**

  クライアントと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを無効にするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
  nagle disable
  ```

  クライアントと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを再度有効にするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
  nagle enable
  ```

- サーバと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを無効、または再度有効にするには、**ssl-server** *number* **tcp server nagle** コマンドを使用します。このコマンドのシンタックスは次のとおりです。

  ssl-server *number* **tcp server nagle enable|disable**

  サーバと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを無効にするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle
  disable
  ```

  サーバと SSL モジュールの間の TCP 接続の Nagle アルゴリズムを再度有効にするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle
  enable
  ```

SSL TCP 接続の TCP バッファリングの指定

ネットワークの速度が遅くて輻輳が発生している場合、特定の TCP 接続に対して、TCP ウィンドウがシャットダウンされて 0 に設定されるまでにバッファリングするデータ量（バッファ サイズ）を増やすことができます。バッファ サイズを増やすと、クライアントへの遅い接続の待機時間は減りますが、CSS でのバッファリング時間は増加します。速度の遅いクライアント接続が多数存在すると CSS のメモリが不足する可能性があるため、この機能を使用する場合は注意が必要です。

特定の接続でクライアントまたはサーバからの TCP バッファリングを設定するには、**ssl-server** *number* **tcp buffer-share** コマンドを使用します。このコマンドのシンタックスは次のとおりです。

ssl-server *number* **tcp buffer-share rx** *number1*|**tx** *number2*

- 特定の接続でバッファリングできるクライアント　トラフィックからのデータ量（バイト数）を設定するには、**rx** *number1* キーワードと変数を使用します。デフォルトのバッファ サイズは 32768 です。バッファ サイズには 16400 ～ 262144 の値を指定できます。たとえば、値を 65535 に設定するには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share
  rx 65536
  ```

  バッファ サイズをデフォルトの 32768 にリセットするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp
  buffer-share rx
  ```

- 特定の接続でバッファリングできるサーバからクライアントへのデータ量（バイト数）を設定するには、**tx** *number2* キーワードと変数を使用します。デフォルトのバッファ サイズは 65536 です。バッファ サイズには 16400 ～ 262144 の値を指定できます。たとえば、値を 131072 に設定するには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share
  tx 131072
  ```

  バッファ サイズをデフォルトの 65536 にリセットするには、次のように入力します。

  ```
  (config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp
  buffer-share tx
  ```

SSL プロキシ リストのアクティブ化と使用中断

SSL プロキシ リストをアクティブ化する前に、仮想 SSL サーバまたはバックエンド SSL サーバの定義を 1 つ以上、リスト内に作成してください（この章で前述した「SSL プロキシ リストでの仮想 SSL サーバの設定」または「SSL TCP 接続の Nagle アルゴリズムの指定」参照）。

CSS は SSL プロキシ リストをチェックして、必要なコンポーネントがすべて設定されているか確認します。このときに、証明書とキー ペアの相互確認も行います。確認に失敗した場合、証明書名は受け入れられず、エラー メッセージ「Certificate and key pair do not match」がログに記録され、SSL プロキシ リストはアクティブ化されません。設定されているキー ペアを削除するか、適切な証明書を設定する必要があります。

新しいまたは変更された SSL プロキシ リストをアクティブ化するには、**active** コマンドを使用します。たとえば、次のように入力します。

```
(config-ssl-proxy-list[ssl_list1])# active
```

SSL プロキシ リストは、アクティブ化するとサービスに追加できます。この章で後述する「SSL 終了のサービスの設定」を参照してください。

リスト内の仮想サーバまたはバックエンド SSL サーバを表示するには、**show ssl-proxy-list** を使用します（第7章「SSL の設定情報および統計情報の表示」参照）。

アクティブな SSL プロキシ リストの使用を一時停止するには、**suspend** コマンドを使用します。

SSL プロキシ リストを一時停止するには、次のコマンドを入力します。

```
(config-ssl-proxy-list[ssl_list1])# suspend
```

SSL プロキシ リストの変更

SSL プロキシ リストは、リストがアクティブになっているときには変更できません。変更を加える前に SSL プロキシ リストの使用を一時停止し、変更が終了したらリストを再度アクティブ化します。

プロキシ リストを変更すると、SSL サービスを一時停止したり、そのリストを使用して再度アクティブ化したりする必要はありません。SSL モジュールにより、次のことが実行されます。

- プロキシ リストを使用して SSL サービスへ追加情報または変更内容を送信する。
- 変更または削除された SSL 関連サービスの接続を解除する。SSL モジュールがこれらの接続に関するパケットを受信すると、SSL モジュールは TCP RST を送信します。

注意　フロントエンド サーバとバックエンド サーバをフローに使用している場合は、エンドツーエンド接続を行うために両方のサーバがアクティブである必要があります。SSL プロキシ リストを変更する場合、サービスがまだアクティブなうちはリストからバックエンド サーバを削除しないでください。SSL プロキシ リストを再びアクティブ化する際、エンドツーエンド接続が失敗してしまいます。

SSL 終了のサービスの設定

SSL プロキシ リストは複数の SSL サービスに属すことができます（サービスにつき 1 つの SSL プロキシ リストを指定）。また、1 つの SSL サービスが複数のコンテンツ ルールに属すことも可能です。サービスをコンテンツ ルールに適用すると、SSL のコンテンツ要求を転送できます。

（注）　CSS は、CSS 内の SSL モジュールごとに 1 つのアクティブな SSL サービス（1 つのスロットにつき 1 つの SSL サービス）をサポートします。1 つのスロットに対して複数の SSL サービスを設定できますが、アクティブにできるのは一度に 1 つの SSL サービスだけです。

ここでは、次の内容について説明します。

- SSL サービスの作成
- SSL アクセラレーション サービス タイプの指定
- SSL 終了サービスへの SSL プロキシ リストの追加
- SSL モジュール スロットの指定
- SSL モジュールへのキープアライブ メッセージの無効化
- SSL セッション ID のキャッシュ サイズの指定
- SSL サービスのアクティブ化
- SSL サービスの一時停止

SSL サービスの作成

SSL モジュールで使用するサービスを作成する際には、そのサービスを CSS の SSL サービスとして指定し、CSS にそれを認識させる必要があります。サービスの作成についての詳細は、『*Cisco Content Services Switch Content Load-Balancing Configuration Guide*』を参照してください。

SLL サービス名は 1 ～ 31 文字で入力します。

サービス *ssl_serv1* を作成するには、次のコマンドを入力します。

```
(config)# service ssl_serv1
Create service <ssl_serv1>, [y/n]: y
```

CSS が、新しく作成されたサービスのモードに変わります。

```
(config-service[ssl_serv1])#
```

SSL アクセラレーション サービス タイプの指定

SSL サービスを作成し、CSS がサービス モードに変わったら、サービス タイプとして **ssl-accel** を指定して次の作業を行う必要があります。

- サービスを SSL アクセラレーション サービスとして設定
- SSL プロキシ リストを SSL サービスに追加

SSL アクセラレーション サービス タイプを指定するには、**type** コマンドを使用します。SSL のサービス タイプの指定については、『*Cisco Content Services Switch Content Load-Balancing Configuration Guide*』を参照してください。

SSL アクセラレーション サービス タイプを指定するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# type ssl-accel
```

SSL 終了サービスへの SSL プロキシ リストの追加

SSL モジュール用の SSL プロキシ リストで仮想 SSL サーバを定義したら、リストをアクティブ化して、SSL サービスに追加します。アクティブ リストには、CSS が特定の SSL モジュールを通じて、SSL のコンテンツ要求を処理する方法が定義されています。SSL プロキシ リストを SSL サービスに追加するには、サービス モードで **add ssl-proxy-list** コマンドを使用します。サービスに追加する作成済みの SSL プロキシ リストの名前（この章の「SSL プロキシ リストの作成」参照）を入力します。

SSL プロキシ リスト *ssl_list1* をサービス *ssl_serv1* に追加するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

SSL プロキシ リストをサービスから削除するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# remove ssl-proxy-list ssl_list1
```

SSL モジュール スロットの指定

CSS 11501 は統合型の SSL モジュールを 1 つサポートします。CSS 11503 と CSS 11506 では、複数の SSL モジュール（CSS 11503 では最大 2 つ、CSS 11506 では最大 4 つ）を使用できます。SSL サービスでは、SSL プロキシ リストと仮想 SSL サーバを特定の SSL モジュールに関連付けるために、SSL モジュールのスロット番号が必要です。SSL モジュールが装着されている CSS シャーシのスロットを指定するには、**slot** コマンドを使用します。

有効なスロット番号は次のとおりです。

- CSS 11501：2
- CSS 11503：2 と 3
- CSS 11506：2 ～ 6

スロット 1 は、SCM 用に予約されています。

（注） CSS は、CSS 内の SSL モジュールごとに 1 つのアクティブな SSL サービス（1 つのスロットにつき 1 つの SSL サービス）をサポートします。1 つのスロットに対して複数の SSL サービスを設定できますが、アクティブにできるのは一度に 1 つの SSL サービスだけです。

たとえば、CSS シャーシのスロット 3 にある SSL モジュールを指定するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# slot 3
```

SSL モジュールへのキープアライブ メッセージの無効化

SSL モジュールは、CSS シャーシに一体化して動作するデバイスであるため、SSL サービスにキープアライブ メッセージを送信する必要はありません。サービスにキープアライブ メッセージを送信しないようにするには、**keepalive type none** コマンドを使用します。キープアライブ タイプの指定の詳細については、『*Cisco Content Services Switch Content Load-Balancing Configuration Guide*』を参照してください。

SSL サービスへのキープアライブ メッセージの送信を無効にするには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# keepalive type none
```

SSL セッション ID のキャッシュ サイズの指定

このキャッシュ サイズは、SSL モジュールの専用セッション キャッシュに保存できる SSL セッション ID の最大数で表します。デフォルトでは、SSL セッション キャッシュには 10000 セッション格納できます。SSL サービスで必要な場合は、SSL セッション キャッシュのサイズを 100000 まで増やすことができます。サービスの SSL セッション ID キャッシュのサイズを再設定するには、**session-cache-size** コマンドを使用します。有効な値は 0（SSL セッション キャッシュは無効）～ 100000 セッションです。

（注） SSL セッション ID が再使用されるようにするには、**session-cache-size** コマンドに値 0 を指定することはお勧めできません。SSL セッション キャッシュとキャッシュ タイムアウトを指定すると、次回以降のそのクライアントと CSS SSL モジュール間の接続でマスター キーを再使用できるため、SSL ネゴシエーション プロセスが高速化され、CSS のパフォーマンスも向上します。

バックエンド セッション ID キャッシュに格納できるエントリ数は 4096 です。この値は変更できません。

SSL セッションのキャッシュ サイズを 0 に指定すると、その SSL サービスに関連する SSL モジュールでは、SSL セッション ID はキャッシュに一切保存されません。SSL セッション キャッシュを無効にする場合は、SSL セッション ID が使用されないようにするために、必ず次のパラメータを適切に設定します。

- SSL プロキシ リスト内で仮想 SSL サーバの **ssl-server** *number* **session-cache timeout** を 0 （無効）に設定する。
- SSL スティッキを無効にするために、コンテンツ ルールで **advanced-balance ssl** コマンドを無効にする。

たとえば、SSL セッションのキャッシュ サイズを 20000 セッションに指定するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# session-cache-size 20000
```

SSL セッションのキャッシュ サイズをデフォルトの 10000 セッションにリセットするには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# no session-cache-size
```

SSL サービスのアクティブ化

SSL プロキシ リストのサービスを設定したら、**active** コマンドを使用してそのサービスをアクティブ化します。サービスをアクティブ化すると、そのサービスは、クライアントとサーバ間で行われる SSL コンテンツ要求のロード バランシングのためのリソース プールに格納されます。

SSL サービスをアクティブ化する前に、次の作業を行います。

- 仮想 SSL サーバの場合は、サービスをアクティブ化する前に、SSL プロキシ リストを **ssl-accel** タイプのサービスに追加する必要があります。**active** コマンドを入力したときにリストが設定されていないと、次のエラー メッセージがログに記録され、サービスはアクティブ化されません。

  ```
  Must add at least one ssl-proxy-list to an ssl-accel type service
  ```

- バックエンド SSL サーバの場合は、サービスをアクティブ化する前に、SSL プロキシ リストを **ssl-accel-backend** タイプのサービスに追加する必要があります。**active** コマンドを入力したときにリストが設定されていないと、次のエラー メッセージがログに記録され、サービスはアクティブ化されません。

  ```
  Must add at least one ssl-proxy-list to an ssl-accel type service
  ```

- サービスに追加した SSL プロキシ リストを、サービスをアクティブ化する前にアクティブにします。リストが一時停止されると、次のエラー メッセージがログに記録され、そのサービスはアクティブ化されません。

  ```
  No ssl-lists on service, service not activated
  ```

サービスをアクティブ化する準備ができると、各 SSL プロキシ リストの適切な SSL 設定データが特定の SSL モジュールへ転送され、サービスがアクティブ化されます。転送時にエラーが発生すると、対応するエラーがログに記録され、サービスはアクティブ化されません。

サービス *ssl_serv1* をアクティブ化するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# active
```

SSL サービスの一時停止

SSL サービスを一時停止し、SSL コンテンツ要求のロード バランシングを行うためのリソース プールからそのサービスを削除する場合には、**suspend** コマンドを使用します。SSL サービスを一時停止しても既存のコンテンツ フローには影響ありませんが、新たに接続を確立して、そのサービスのコンテンツを得ることはできません。

サービス ssl*_serv1* を一時停止するには、次のコマンドを入力します。

```
(config-service[ssl_serv1])# suspend
```

SSL 終了のコンテンツ ルールの設定

CSS がコンテンツへの SSL 要求を転送するには、仮想サービスをコンテンツ ルールに適用します。SSL コンテンツ ルールをアクティブ化して、コンテンツが実際に存在する場所、コンテンツ要求の送信先（SSL サービス）、および使用するロード バランシング方式を定義するまで、ネットワーク トラフィックは SSL モジュールに送信されません。

仮想 SSL サーバのコンテンツ ルールでは、設定する VIP アドレスとポート番号は、SSL プロキシ リストのそのサーバのエントリの設定と一致させる必要があります。

SSL サービスで設定したコンテンツ ルールをアクティブ化すると、一致している VIP アドレスとポートがあるかどうかが検証されます。一致が見つからない場合、次のエラー メッセージがログに記録され、コンテンツ ルールはアクティブ化されません。

```
Not all content VIP:Port combinations are configured in an
ssl-proxy-list for sslAccel type of service
```

コンテンツ ルールと SSL プロキシ リストで使用されている設定済みの VIP アドレスを確認し、必要に応じて変更します。

CSS が 2 つ以上の SSL モジュールを使用している場合は、レイヤ 5 コンテンツ ルールの SSL バージョン 3 セッション ID に基づくスティッキ性を使用することをお勧めします。仮想 SSL サーバ ルールでは、次のように指定します。

- **advanced-balance ssl** コマンドを使用して、SSL に基づくコンテンツ ルールのスティッキ性を有効にする。
- **application ssl** コマンドを使用して、SSL アプリケーション タイプを指定する。

レイヤ 5 SSL スティッキ コンテンツ ルールでは、SSL セッション ID の再使用によって再ハンドシェイクが回避されるため、SSL ネゴシエーション プロセスが高速化され、全体的なパフォーマンスも向上します。

（注） 32K のスティッキ テーブルが一杯になる（サイトに 32000 人のユーザが同時にアクセスしている状態）と、テーブルの先頭に戻り、最初のユーザが「非固定」状態になります。この現象は、フロー数とスティッキ期間の長さの組み合わせが原因で発生することがあります。組み合わせによっては、スティッキ テーブル内の使用可能な領域が急激に消費されるためです。この問題は通常、複数の SSL モジュールを搭載した CSS で発生します。288M メモリ モジュールを持つ SCM では、128K のスティッキ テーブルをサポートできます。

（注） SSL スティッキを使用するレイヤ 5 コンテンツ ルールで **sticky-inact-timeout** コマンドを指定すると、スティッキ テーブルが一杯になった場合でも、SSL セッションは継続されますが、CSS は新しいセッションでスティッキ性を維持しません。