CHAPTER 15

スタンドアロン Content Engine の拡張透過キャッシング機能の設定

この章では、ACNS 5.4.x ソフトウェアおよびそれ以降のリリースを実行しているスタンドアロン Content Engine に拡張透過キャッシング機能を設定する方法について説明します。この章の内容は、次のとおりです。

拡張透過キャッシング機能の概要

透過ネットワーク キャッシングの基本原則の 1 つは、Content Engine がエンド ユーザに対して常に透過的でなければならないということです。また、透過キャッシング ソリューションが原因でネットワークに何らかの障害が発生したり、二次的に障害を引き起こしたりしないようにする必要があります。

ACNS ソフトウェアでは、クライアント ブラウザが動作しない場合や、Web サーバが HTTP に準拠していない場合でも、WCCP 対応ルータと各種の先進技術を使用して Content Engine の透過性を確保します。

表 15-1 は、拡張透過キャッシング機能の一覧です。この一連の機能により、シスコ キャッシング ソリューションの配置時に（キャッシュ エンジンとしてのスタンドアロン Content Engine の配置も含む）、予想しない問題が発生するのを防ぐことができます。

表 15-1　スタンドアロン Content Engine のための拡張透過キャッシング機能

技術サービス	利点
Bypass	
認証トラフィックのバイパス	選択中のクライアント / サーバのペア用のバイパス リストを Content Engine に生成させることで、キャッシュの透過性を維持し、サービスの中断を防ぎます。詳細は、「スタンドアロン Content Engine での認証トラフィック バイパスの設定」（p.15-4）を参照してください。
スタティック バイパス	指定のソースからのトラフィックを許可し、Content Engine をバイパスします。詳細は、「スタンドアロン Content Engine でのスタティック バイパスの設定」（p.15-7）を参照してください。
過負荷バイパス	トラフィック負荷が Content Engine の処理能力を超えたときに、Content Engine がボトルネックになるのを防ぎます。詳細は、「スタンドアロン Content Engine での過負荷バイパスの設定」（p.15-8）を参照してください。
WCCP フロー保護	Content Engine をクラスタに追加、またはクラスタから除去したのが原因で WCCP クラスタ負荷分散が変わったときに、既存のフローが途切れるのを防止します。詳細は、「WCCP フロー保護の設定」（p.15-10）を参照してください。
WCCP スロー スタート	新しい Content Engine を重い負荷のクラスタに追加したときに、クラスタが不安定になるのを防ぎます。詳細は、「WCCP スロー スタートの設定」（p.15-11）を参照してください。
WCCP IP スプーフィング	オリジン Web サーバとの接続時にクライアントの IP アドレスを使用します。詳細は、「WCCP IP スプーフィングの設定」（p.15-12）を参照してください。

ACNS ソフトウェアにはビルトイン バイパス機能も組み込まれています。この機能はユーザ側で設定することはできません。ビルトイン バイパス機能は、WCCP 起動前に開いていた接続と、接続終了後のクライアント パケット再伝送に作用します。Content Engine は、このビルトイン バイパス機能を通して、これらのトラフィックを自動的にルータに送り返します。

この章の各項では、スタンドアロン Content Engine で拡張透過キャッシング機能を設定する方法について説明します。

- スタンドアロン Content Engine でのバイパスの設定（p.15-3）
- WCCP フロー保護の設定（p.15-10）
- WCCP スロー スタートの設定（p.15-11）
- WCCP IP スプーフィングの設定（p.15-12）

スタンドアロン Content Engine でのバイパスの設定

バイパスとは、オリジン サーバからの各種エラー応答（認証の失敗を含む）を処理するために Content Engine 側で使用する手法のことです。Content Engine は、オリジン サーバからエラー応答を受信すると、そのサーバ用のエントリをバイパス リストに追加します。その後、バイパスしたサーバ上のコンテンツに対する要求を受信すると、Content Engine はバイパス ゲートウェイにパケットをリダイレクトします。バイパス ゲートウェイが未設定の場合は、リダイレクト側のレイヤ 4 スイッチにパケットが送り返されます。

バイパス機能は、WCCP Version 2 ルータのほか、Cisco Content Switching Module や Content Services Switch（CSS）スイッチなどのレイヤ 4 スイッチで使用できます。

ここでは、次のタイプのバイパスをサポートするようにスタンドアロン Content Engine を設定する方法について説明します。

- スタンドアロン Content Engine での認証トラフィック バイパスの設定（p.15-4）
- スタンドアロン Content Engine でのスタティック バイパスの設定（p.15-7）
- スタンドアロン Content Engine での過負荷バイパスの設定（p.15-8）

（注）　バイパス機能は、WCCP Version 2 がローカル ネットワーク内で有効になっている場合にのみ使用できます。Content Engine は、WCCP でリダイレクトされたトラフィックのみをバイパスでき、プロキシスタイルの要求をバイパスしません。

バイパス リスト内のエントリ数を含むバイパス サマリーを表示するには、**show bypass summary** EXEC コマンドを入力します。

```
ContentEngine# show bypass summary
Total number of requests bypassed = 0
        Requests bypassed due to system overload                  = 0
        Requests bypassed due to authentication issues            = 0
        Requests bypassed due to facilitate error transparency    = 0
        Requests bypassed due to static configuration             = 0
Total number of entries in the bypass list = 1
        Number of Authentication bypass entries = 0
        Number of Error bypass entries          = 0
        Number of Static Configuration entries  = 1
L2 Bypass:
        Number of L2 bypassed packets = 0
ContentEngine#
```

バイパス リスト内のエントリの一覧（下記の出力例を参照）を表示するには、**show bypass list** EXEC コマンドを入力します。

```
ContentEngine# show bypass list

         Client                Server         Entry type
         ------                ------         ----------
        1.1.1.1:0             5.5.5.5:0     static-config
```

スタンドアロン Content Engine での認証トラフィック バイパスの設定

Web サイトでは、クライアントの IP アドレスを使用してクライアントを認証する場合があります。このクライアント認証方式は通常、旧型の Web サーバでしか使用されません。ただし、このような状況では、オリジン Web サーバ側でクライアントを認証できるように、クライアントと オリジン Web サーバ間に Content Engine が「その場所から離れる」ための脇道が必要になります。ダイレクトなプロセス ルーティングを使用した場合、この方法は使えません。発信プロキシ サーバとして Content Engine にダイレクトにアクセスするようにクライアントが設定されているためです（Content Engine は、クライアント ブラウザやメディア プレーヤーから要求を直接受け取ります）。ただし、WCCP で代行受信される要求の場合は、Content Engine で認証トラフィック バイパス機能を有効にすると、要求が Content Engine をバイパスできるため、オリジン Web サーバ側でクライアントを認証できます。

このバイパス機能の別の使用例として、IP 認証が原因でクライアントに代わってダイレクトに接続するのを Content Engine に許可しない Web サイトが挙げられます。キャッシュの透過性を保ち、サービスの中断を防ぐために、Content Engine 側では、このバイパス機能を使用して、選択されたクライアント / サーバのペアのためのダイナミック アクセス リストを自動生成できます。階層キャッシングのケースでは、認証トラフィック バイパス トリガーがアップストリームとダウンストリームにも伝搬します。

デフォルトでは、Content Engine の認証トラフィック バイパス機能は無効になります。スタンドアロン Content Engine で認証トラフィック バイパス機能を有効にするには、**bypass auth-traffic** グローバル コンフィギュレーション コマンドを使用します。クライアント / サーバのペアが認証トラフィック バイパスに入ると、**bypass timer** グローバル コンフィギュレーション コマンドで設定した時間（デフォルトは 20 分）内は、このペアがバイパスされます。たとえば、クライアント / サーバのペアが認証バイパスを実行すると、設定された時間、バイパスされます。この時間は、**bypass timer** グローバル コンフィギュレーション コマンドで設定します。

Content Engine で認証トラフィック バイパス機能を有効にすると、次のような状況が発生します。

1. クライアント（ブラウザを使用してコンテンツを要求しているエンド ユーザ）は、コンテンツ要求を Web サーバ（オリジン Web サーバ）に送信します。

   （注）　WCCP で代行受信される要求のみをバイパスできます。プロキシ スタイルの要求の場合は、プロキシ サーバとして Content Engine にダイレクトに接続するようにクライアント ブラウザが明示的に設定されるため、プロキシ スタイルの要求に対してはバイパスを設定できません。

2. WCCP ルータは、コンテンツ要求を透過的に代行受信し、その要求を Content Engine（透過プロキシ サーバ）に転送します。
3. Content Engine は、オリジン Web サーバを装って、クライアントに応答します。同時に Content Engine が自身の IP アドレスを使用して、オリジン Web サーバに要求を送信します。
4. オリジン Web サーバは、IP アドレスに基づく要求認証を実行している場合、この要求を拒否します。
5. Content Engine は、オリジン Web サーバから 401、403、501、503、502、503、504、505 のいずれかの応答を受信すると、次のような認証トラフィック バイパス アクションを実行します。
   a. 完全に同じ URL を持つクライアントにリダイレクトを送信します。
   b. バイパス エントリ（クライアント / サーバ ペアのエントリ）をバイパス リストに追加します。

6. クライアントから再試行された要求は、WCCP Version 2 ルータによって再び代行受信され、Content Engine に転送されます。このとき、Content Engine は要求を処理せず、オリジン Web サーバに要求を転送します。これは、直前にクライアント / サーバ バイパス エントリがバイパス リストに作成されているためです。
7. オリジン Web サーバがクライアントに応答し、応答はクライアントにダイレクトに送信されます。

Content Engine GUI か CLI のいずれかを使用して、スタンドアロン Content Engine に認証トラフィック バイパスを設定できます。

- Content Engine GUI で **Caching > Bypass** の順に選択します。Bypass ウィンドウが表示されます。**Authentication Bypass On** オプション ボタンをクリックして、認証トラフィック バイパスを有効にします。Bypass Entry Expiration Time フィールドに、バイパス アクセス リスト上のクライアント / サーバ ペアのアイドル状態を維持する時間（分）を指定します。デフォルト値は 20 分です。**Update** をクリックして設定値を保存します。
- Content Engine CLI から **bypass auth-traffic** と **bypass gateway** のグローバル コンフィギュレーション コマンドを使用します。これらのパラメータについては、表 15-2 を参照してください。

  bypass {**auth-traffic enable** | **gateway** *ipaddress* | **timer** *minutes*}

表 15-2　認証トラフィック バイパス コマンドのパラメータ

パラメータ	説明
auth-traffic	認証トラフィック バイパス機能を設定します。
enable	認証トラフィック バイパスを有効にします。
gateway	レイヤ 4 スイッチによってリダイレクトされた要求を Content Engine が受信したときに、バイパスするパケットのリダイレクト先ルータに設定します。
ipaddress	バイパス ゲートウェイとして動作するルータの IP アドレス
timer	認証バイパス タイマー（分）を設定します。タイマーが切れると、ダイナミック リストからバイパス エントリが削除されます。
minutes	分単位の時間（1 ～ 1440）

次の例では、すべての認証済み HTTP トラフィックに 24 時間、強制的に Content Engine をバイパスさせます。

```
ContentEngine(config)# bypass auth-traffic enable
ContentEngine(config)# bypass timer 1440
```

オリジン サーバからエラーを受信したときに Content Engine が応答を送信する WCCP Version 2 ルータを識別するには、**bypass gateway** *ipaddress* グローバル コンフィギュレーション コマンドを使用します。*ipaddress* には、Content Engine のレイヤ 2 近隣ルータの IP アドレスを指定します。

レイヤ 4 スイッチでのバイパスを有効にするには、**http l4 switch enable** グローバル コンフィギュレーション コマンドを使用します。

スタンドアロン Content Engine で認証トラフィック バイパス機能を無効にするには、**bypass auth-traffic** グローバル コンフィギュレーション コマンドの **no** 形式を使用します。

（注）　**bypass auth-traffic** グローバル コンフィギュレーション コマンドを使用して、スタンドアロン Content Engine での透過エラー処理を有効にすることもできます。

例 1：Web サーバ エラー受信時のダイナミック バイパス

次の例およびその次の例では、WCCP リターン パス機能が導入されます。この機能により、Content Engine が WCCP 対応ルータまたはスイッチにトラフィックを返し、Content Engine が存在しないかのようにパケットを転送するよう、ルータまたはスイッチに指示することができます。

HTTP トラフィック フロー全体の約 3 % が失敗します。これらの失敗したフローは、認証バイパスまたはダイナミック クライアント バイパスを使用して自動的に再試行されます。これにより、障害の原因が以前から存在しており、透過キャッシングの配置が原因ではないことが証明されます。

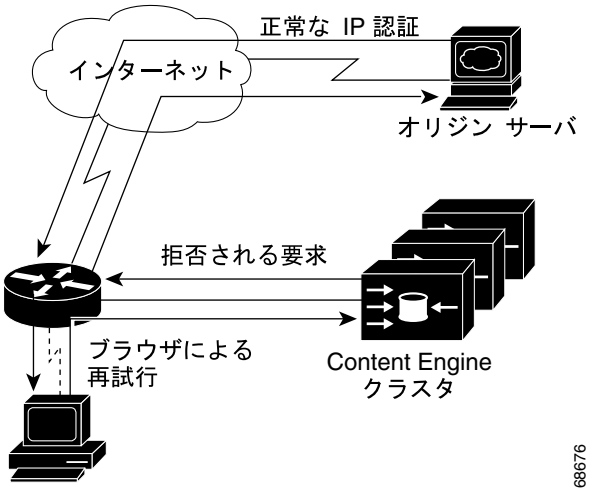
ユーザは、Web ブラウザから HTTP 要求を発行します。この要求は透過的に代行受信され、Content Engine にリダイレクトされます。Content Engine は、Web ブラウザから着信 TCP 接続を受け入れ、この要求がストレージ内に存在しないオブジェクトに対する要求である（キャッシュ ミスである）ことを判別し、オブジェクト要求をオリジン Web サーバに送信します。ただし、Web サーバから何らかのエラー メッセージ（たとえば、プロトコル エラーや認証エラー）を受け取ります。

Content Engine は、Web ブラウザからの TCP 接続をすでに受け入れているので、3 方向の TCP ハンドシェイクが行われています。Content Engine は、Web サーバとのトランザクションが失敗したことは認知できますが、その原因を特定することはできません（たとえば、オリジン Web サーバがユーザの送信元 IP アドレスに基づく認証を実行している場合や、TCP スタック間で互換性がない場合）。

このケースでのダイナミック クライアント バイパスは、Content Engine が HTTP 応答コードをブラウザに返すことを意味します。返される応答は、リフレッシュを要求するメタ タグ付きの HTML ページです。Content Engine は、「Connection: close」HTTP 応答ヘッダーを Web ブラウザに出して、Web ブラウザと Content Engine 間の TCP 接続を解除します。ここでブラウザは、自動的に接続を再試行します。

接続の再試行時には、Content Engine は接続を受け入れません。Content Engine は、要求を代行受信しないまま WCCP 対応ルータまたはスイッチに返します。次にルータは、Web ブラウザから直接、オリジン Web サーバにフローを送信し、Content Engine をバイパスします（図 15-1 を参照）。

図 15-1　ダイナミック トラフィック バイパス



例 2：サポートされないプロトコル受信時のダイナミック バイパス

Content Engine が TCP ポート 80 を介して非 HTTP 要求を受信したとき、Content Engine は、retry 応答を出し、接続を切断し、以後の接続は、例 1 と同じように受け入れません。retry 応答は、リフレッシュまたは再試行を要求していることを伝える通常の HTTP 応答です。

（注）　非 HTTP には、非準拠 HTTP や、Secure Shell（SSH ; セキュア シェル）、Simple Mail Transfer Protocol（SMTP; 簡易メール転送プロトコル）、または Network News Transport Protocol（NNTP）などのさまざまなプロトコルがあります。非準拠 HTTP とは、たとえば Web サーバが、HTTP ヘッダー セクションの末尾でキャリッジ リターンとライン フィードを 2 回実行できない場合などです。

すべての HTTP クライアントが HTML をサポートしているとは限りません。クライアントまたはサーバが HTTP/HTML をサポートしていない場合には、クライアント側で問題が発生する可能性があり、Content Engine では、クライアントに要求されたコンテンツを処理できません。

スタンドアロン Content Engine でのスタティック バイパスの設定

bypass static 機能を使用して、特定の送信元からのトラフィックが Content Engine をバイパスできるようにします。トラフィック送信元のタイプは、次のとおりです。

- 特定の Web クライアントから特定の Web サーバまで
- 特定の Web クライアントから任意の Web サーバまで
- 任意の Web クライアントから特定の Web サーバまで

スタンドアロン Content Engine でスタティック バイパス機能をイネーブルにし、設定するには、**bypass static** グローバル コンフィギュレーション コマンドを使用します。

bypass static {*clientip* | **any-client**} {*serverip* | **any-server**}

表 15-3 では、コマンド パラメータを説明します。

表 15-3　bypass static コマンドのパラメータ

パラメータ	説明
static	バイパス リストにスタティック エントリを追加します。
clientip	要求が Content Engine をバイパスするときの送信元の IP アドレス。ワイルドカードはサポートされません。
serverip	要求が Content Engine をバイパスするときの送信先の IP アドレス。ワイルドカードはサポートされません。
any-server	特定のクライアントからサーバへの要求が Content Engine をバイパスします。
any-client	特定のサーバへ送られるすべてのクライアントからの HTTP トラフィックが Content Engine をバイパスします。

（注）　すべてのスタティック設定リストをクリアするには、**bypass static** グローバル コンフィギュレーション コマンドの **no** 形式を使用します。

次の例は、**bypass static** グローバル コンフィギュレーション コマンドによるスタンドアロン Content Engine でのスタティック バイパス機能の設定方法を示しています。

この例では、特定のクライアントから特定のサーバへの HTTP トラフィックが、Content Engine をバイパスするように強制されます。

```
ContentEngine(config)# bypass static 10.1.17.1 172.16.7.52
```

次の例では、特定のサーバ宛てのすべての HTTP トラフィックが強制的に Content Engine をバイパスします。

```
ContentEngine(config)# bypass static any-client 172.16.7.52
```

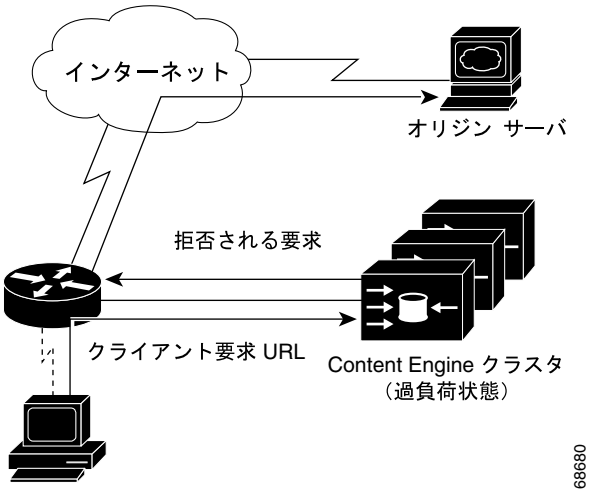
次の例では、特定のクライアントから任意の Web サーバへのすべての HTTP トラフィックが強制的に Content Engine をバイパスします。

```
ContentEngine(config)# bypass static 10.1.17.1 any-server
```

スタンドアロン Content Engine での過負荷バイパスの設定

Content Engine が過負荷になったときに、過負荷バイパス オプションがイネーブルになっていると、Content Engine がバケットをバイパスして、過負荷トラフィックを再ルーティングします。それでも負荷が大きすぎる場合は、さらにバケットがバイパスされ、Content Engine が負荷を処理できるようになるまでバイパスが続けられます（図 15-2 を参照）。

図 15-2　過負荷バイパス



（注）　バケットは、Content Engine クラスタ内の各 Content Engine に割り当てられたハッシュのサブセクションとして定義されます。この環境に Content Engine が 1 つだけ存在する場合は、256 のバケットが Content Engine に割り当てられます。

最初のバケットのバイパスが発生したあと、Content Engine がバイパスされたバケットへのサービスを再開するには、ある程度の時間が必要です。デフォルトでは、この時間間隔は 10 分に設定されます。

バイパスされたトラフィックのサービスを Content Engine が再開する際に、最初は 1 つのバイパス バケットからサービスを開始します。それ以上の負荷のサービスが可能ならば、Content Engine はバイパス バケットをもう 1 つ受け取り、以後も同様に行います。あるバケットを取り出してから、次のバケットを取り出すまでの時間間隔は、デフォルトでは 60 秒に設定されます。

Content Engine GUI か CLI のいずれかを使用して、スタンドアロン Content Engine に過負荷バイパスを設定できます。

- Content Engine GUI で **Caching > Bypass** の順に選択して、表示される Bypass ウィンドウを使用します。Bypass ウィンドウによる負荷バイパスの設定方法については、Bypass ウィンドウの **HELP** ボタンをクリックしてください。
- Content Engine CLI で **bypass load** グローバル コンフィギュレーション コマンドを使用します。

  bypass load {**enable** | **in-interval** *seconds* | **out-interval** *seconds* | **time-interval** *minutes*}

表 15-4 では、コマンド パラメータを説明します。

表 15-4　bypass load コマンドのパラメータ

パラメータ	説明
load	バイパス リストにスタティック エントリを追加します。
enable	Content Engine の過負荷バイパスを有効にします。
in-interval	バケットが戻るまでの時間間隔を設定します。
seconds	秒単位の時間（2 ～ 600）。デフォルトは 60 秒です。
out-interval	バケットのバイパス間の時間間隔を設定します。
seconds	秒単位の時間（4 ～ 600）。デフォルトは 4 秒です。
time-interval	あるバケットをバイパスし、次のバケットをバイパスするまでの時間間隔を設定します。
minutes	分単位の時間（1 ～ 1440）。デフォルトは 10 分です。

WCCP フロー保護の設定

WCCP フロー保護機能は、新しい Content Engine がオンラインになるときに、既存のフローが途切れることがないようにします。透過トラフィックの代行受信またはリダイレクションが始まる際に、WCCP フロー保護は、既存の接続済みの HTTP フローを継続させて、これらの HTTP フローがとぎれないようにします。また、WCCP フロー保護は、新しい Content Engine が既存の Content Engine クラスタに加わる際、クラスタ内の既存の Content Engine により提供されているフローが確実に継続するようにします。

WCCP フロー保護で使用するメカニズムは、フローごとのステータス情報を中央で保持することにより、多くの利点をもたらします。オーバーヘッド、スケーリングの問題、スイッチング レイヤにおいて、フローごとにステータス情報を保持することによる冗長性や復元力などの問題（非対称トラフィック フローなど）もありません。

WCCP フロー保護を実装するには、**wccp flow-redirect** グローバル コンフィギュレーション コマンドを使用します。このコマンドは、WCCP Version 2 でのみ動作します。フロー保護は、TCP フローに影響を与えず、Content Engine の導入時や新規トラフィックの再割り当て時に、Content Engine が過負荷にならないように設計されています。また、この機能は、スロー スタート メカニズムも備えています。このメカニズムにより、Content Engine は自身の容量に適した負荷を引き受けようとします。

次の例は、スタンドアロン Content Engine で WCCP フロー保護を有効にする方法を示しています。

```
ContentEngine(config)# wccp flow-redirect enable
```

（注）　バイパスが有効になっている場合は、クライアント自身がオリジン Web サーバへのアクセスを試みます。すべてのバイパス オプションを無効にして、ネットワークに不要な負荷をかけないようにしてください。

ACNS 5.3.1 ソフトウェアおよびそれ以降のリリースでは、ネイティブ FTP キャッシング サービス パケットのフローに関する要約情報を表示するには、**show wccp flows ftp-native** EXEC コマンドを入力します。

WCCP スロー スタートの設定

Content Engine のクラスタ内ユニットを追加または除去すると、TCP 接続が他の Content Engine にリダイレクトされます。新しいトラフィックの再割り当てが早すぎる場合や、太いパイプに突然つながった場合は、Content Engine が過負荷になる可能性があります。

WCCP スロー スタートは、次のタスクを実行して、Content Engine がオンラインになったときや、新しいトラフィックが割り当てられたときに、Content Engine が過負荷になるのを防ぎます。

- WCCP Version 2 が有効で、Content Engine がクラスタに参加しているときの TCP フロー保護
- WCCP Version 2 が無効で、Content Engine がクラスタから離れているときの TCP フロー保護
- ブート時のフル負荷ではなく、徐々に負荷がかかるときの Content Engine への負荷割り当て

スロー スタートは、次のケースでのみ適用可能です。

- サーバ ファームに Content Engine がまだ存在していないときの初期ブート時
- フル負荷を処理しないクラスタに新しい Content Engine を追加したとき、たとえば、クラスタによって除去されているバケットがいくつかあるとき

他のケースではスロー スタートは不要です。すべての Content Engine にトラフィック シェアをすぐに割り当てることができます。

スタンドアロン Content Engine でキャッシング サービスのスロー スタート機能を有効にするには、**wccp slow-start enable** グローバル コンフィギュレーション コマンドを入力します。スロー スタート機能を無効にするには、このコマンドの **no** 形式を入力します。

ACNS 5.3.1 ソフトウェアおよびそれ以降のリリースでは、スタンドアロン Content Engine の WCCP スロー スタート情報を表示するには、**show wccp slowstart ftp-native** EXEC コマンドを入力します。

WCCP IP スプーフィングの設定

一般的な透過キャッシングでは、エンド ユーザは Web ブラウザに HTTP 要求を発行します。この要求は透過的に代行受信され、WCCP ルータによって Content Engine（透過プロキシ サーバとして機能する）にリダイレクトされます。Content Engine は、WCCP ルータからの着信 TCP 接続を受け入れ、要求がストレージに存在しないオブジェクトを求める要求（キャッシュ ミス）かどうかを調べ、要求されたオブジェクトに対する要求をオリジン Web サーバに送ります。Content Engine は、オリジン サーバにアクセスするときには、要求発行の対象となるクライアントの IP アドレスではなく、自身の IP アドレスを使用します。

WCCP Version 2 対応のルータと Content Engine で IP スプーフィングを設定している場合には、Content Engine（透過プロキシ サーバとして動作する）は、自身の IP アドレスを含む要求を送信するのではなく、認証を目的としてクライアントの IP アドレスをオリジン サーバに転送します。WCCP ルータはクライアントの IP アドレス宛てのパケットをサーバからも代行受信し、それらのパケットを Content Engine にリダイレクトします。

クライアントの IP アドレスをスプーフィングすることで、以下の機能がサポートされます。

- Content Engine は、クライアントの IP（Content Engine 自身の IP アドレスと異なる）を使用してパケットを転送できます。
- Content Engine は、クライアントの IP（この場合も、Content Engine 自身の IP アドレスとは異なる）を使用してパケットを受信し、待機している該当のアプリケーションにそのパケットを送信できます。
- WCCP Version 2 対応ルータは、クライアントとサーバの両方からパケットを透過的に代行受信して、TCP 接続が切れないように、これらのリダイレクトされたパケットを同じ Content Engine に送信できます。

ACNS 5.0.7 ソフトウェア リリースより前の ACNS ソフトウェアでは、透過的に代行受信されたプロキシ スタイルの要求に対する IP アドレス スプーフィングがサポートされていませんでした。ただし、オリジナル要求でプロキシ サーバを使用してコンテンツを取得するように、Content Engine が設定されている場合、ACNS 5.0.7 ソフトウェアおよびそれ以降のリリースを使用すると、透過的に代行受信されたプロキシ スタイルの要求に対して、IP アドレス スプーフィングが実行されます。オリジナル要求からプロキシ サーバを使用できるようにスタンドアロン Content Engine を設定するには、**proxy-protocols transparent original-proxy** グローバル コンフィギュレーション コマンドを使用します。

プロキシ スタイルの要求では、HTTP 要求を特定の Content Engine にダイレクトに送信するようにクライアントが設定されている場合、クライアントがプロキシ スタイルの HTTP 要求を送信します。クライアントは、HTTP 方式のオリジン サーバ名（たとえば、GET）を含む完全な送信先 URL を使用して、プロキシ サーバの IP アドレスに要求を送信します。

サーバ スタイルの HTTP 要求の場合、クライアントは、オリジン サーバのドメイン名を含む HTTP ヘッダーや、クライアントが要求しているファイルまたはスクリプトへのパスを含む HTTP 方式を使用して、その要求を送信先サーバに直接送信します。

ドメイン myserver.com の mydirectory にある myfile.html に対するプロキシ スタイルの要求は、透過的に代行受信されるとき、次のような初期 HTTP ラインになります。

GET http://myserver.com/mydirectory/myfile.html HTTP/1.1

ドメイン myserver.com の mydirectory にある myfile.html に対するサーバ スタイルの要求は、透過的に代行受信されるとき、次のような初期 HTTP ラインになります。

GET/mydirectory/myfile.html HTTP/1.1

スタンドアロン Content Engine で IP スプーフィングを有効にするには、**wccp spoof-client-ip enable** グローバル コンフィギュレーション コマンドを入力します。IP スプーフィングを無効にするには、**no wccp spoof-client-ip enable** グローバル コンフィギュレーション コマンドを入力します。

ヒント

Content Engine は、認証トラフィック バイパスを使用して、ダイナミック アクセス リストを自動生成し、選択されたクライアント / サーバのペアに対するクライアントの IP アドレスを使用して、サーバに接続することもできます。詳細は、「スタンドアロン Content Engine での認証トラフィック バイパスの設定」（p.15-4）を参照してください。ユーザは、要求を処理している Content Engine 上で **http append x-forwarded-for-header** グローバル コンフィギュレーション コマンドを使用して、IP スプーフィングをオンにすることなく、クライアントの IP アドレスを転送することもできます。

IP スプーフィングは、次のような状況で推奨されます。

- ユーザ IP アドレスのロギング
- ユーザ IP アドレスに基づいたフィルタリング
- 一部のユーザに他のユーザよりも優れたサービスを提供するためのポリシーベースのルーティング

IP スプーフィングは微妙な問題を発生することがあるので、それを回避して ACNS ユーザを保護するために、いくつかの制限が設けられています。これらの制限により、たとえグローバルな IP スプーフィングが有効になっている場合でも、一部またはすべてのクライアント トラフィックの IP スプーフィングが阻止されます。次のような状況では、IP スプーフィングは意図的に回避されます。

- クライアント要求が透過的にリダイレクトされない場合（プロキシ スタイルの要求を意味します）
- この要求が透過的にリダイレクトされるが、要求がプロキシ スタイルであり、プロキシ プロトコルに透過な元のプロキシ設定がない場合
- HTTP 発信プロキシが設定されていない場合
- クライアント要求が次のどのルール パターンにも一致しない場合
  - rule use-proxy
  - rule use-server
  - rule rewrite

スタンドアロン Content Engine での IP スプーフィング設定の例

IP スプーフィング用にスタンドアロン Content Engine と WCCP Version 2 対応ルータを設定するには、両方のルータを IP スプーフィング用に設定する必要があります。

（注）

Content Engine で IP スプーフィングを有効にするには、クライアント、オリジン サーバ、および Content Engine をサービスするルータ インターフェイスを事前に設定しておく必要があります。WCCP Version 2 対応ルータの 3 つの独立したインターフェイスにクライアント、Content Engine、オリジン サーバを設定する必要があります。

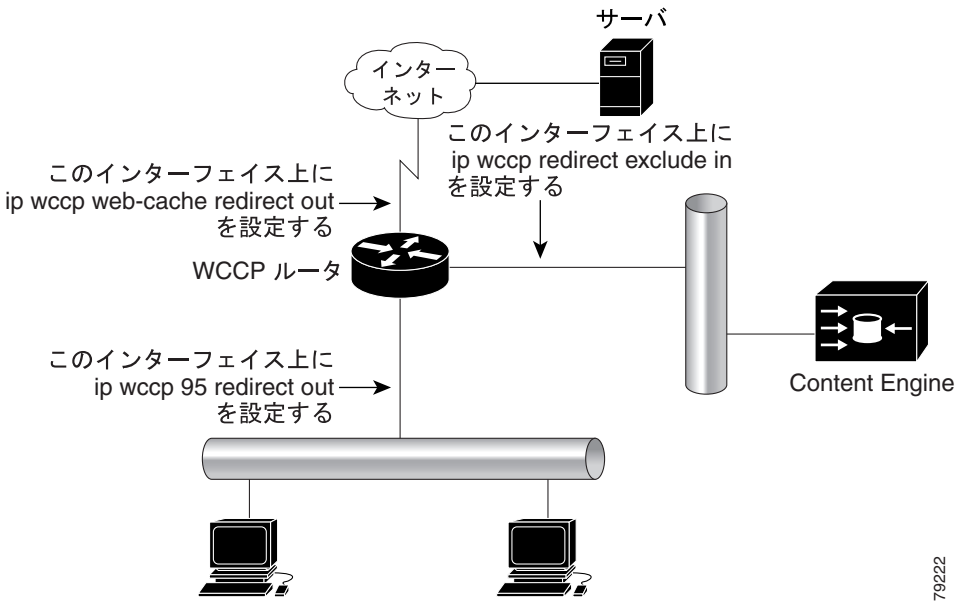
ここでは、スタンドアロン Content Engine と WCCP Version 2 対応ルータでの IP スプーフィングの設定例を示します。

- 例 1 : 異なるサブネットにある Content Engine と クライアントでの IP スプーフィング（p.15-14）
- 例 2 : リバース プロキシ サーバによる IP スプーフィング（p.15-17）

例 1：異なるサブネットにある Content Engine と クライアントでの IP スプーフィング

ここでは、IP スプーフィング用にスタンドアロン Content Engine と WCCP Version 2 ルータを設定する例を示します。この例では、Content Engine と要求側のクライアントが別々のサブネットにあり、2 つの WCCP サービス（Web キャッシュ サービスとサービス 95）が設定されます。送信先 IP アドレス上で 1 つの WCCP サービス（Web キャッシュ サービス）がハッシュされ、送信元 IP アドレス上でもう 1 つの WCCP サービス（サービス 95）がハッシュされます（図 15-3 を参照）。

図 15-3　別々のサブネットにある Content Engine とクライアントでの IP スプーフィング



（注）　送信先 IP アドレスでハッシュされる WCCP サービスに対しては、標準の Web キャッシュ サービス（サービス 0）ではなく、カスタム Web キャッシュ サービス（サービス 98）を使用することもできます。WCCP サービスのリストについては、表 B-3 を参照してください。

Content Engine と クライアントが異なるサブネットにある場合に、スタンドアロン Content Engine と WCCP Version 2 対応ルータで IP スプーフィングを設定する手順は、次のとおりです。

ステップ 1　クライアントと異なるサブネット上にある Content Engine で WCCP Version 2 を有効にします。

```
ContentEngine# configure terminal
ContentEngine(config)# wccp version 2
```

ステップ 2　Content Engine でルータ リストを設定します。このケースではルータ リスト番号 1 が作成され、このリストには、10.10.20.1 の IP アドレスを持つ 1 つの WCCP ルータしか含まれていません。

```
ContentEngine(config)# wccp router-list 1 10.10.20.1
```

ステップ 3　Content Engine 上で、ポート 80 経由で WCCP サービスに関連付けるように、ポート リスト 1 を設定します。

```
ContentEngine(config)# wccp port-list 1 80
```

ステップ 4　Content Engine 上で **wccp service-number** グローバル コンフィギュレーション コマンドを使用して、送信元 IP アドレスでハッシュされるユーザ定義の WCCP サービス（サービス 95）を設定します。

a. WCCP サービス番号を指定します。

b. このサービスを、WCCP Version 2 対応ルータのリスト（ルータ リスト番号 1）と、WCCP サービスをサポートするポート（ポート リスト番号 1）に関連付けます。

c. ハッシング パラメータを送信元 IP アドレスと送信元ポートに関連付けます。

d. 送信元 IP アドレスでハッシュする **ip match-source-port** オプションを指定します。

```
ContentEngine(config)# wccp service-number 95 router-list-num 1
port-list-num 1 application cache hash-source-ip match-source-port
```

（注） WCCP サービス番号 90 ～ 97 は、表 B-3 に示すユーザ定義のサービスです。ユーザ定義のサービスは WCCP サービスの 1 つであり、このサービスによって、トラフィックを Content Engine にリダイレクトするようにポート番号を設定できます。このシナリオでは、サービス 95 を使用して、ユーザ定義の WCCP サービスを作成します。**wccp service-number** グローバル コンフィギュレーション コマンドに **application cache** オプションを指定すると、Content Engine の従来のキャッシュ プロセス（HTTP 要求用）にトラフィックがリダイレクトされますが、**application streaming** オプションを指定すると、Content Engine のメディア キャッシュ プロセス（WMT または RTSP 要求用）にトラフィックがリダイレクトされます。

ステップ 5　リダイレクトされた Web トラフィックを Content Engine が受け入れていることを WCCP ルータに知らせます。

```
ContentEngine(config)# wccp web-cache router-list-num 1
```

ステップ 6　Content Engine のクライアント IP スプーフィングを有効にします。

```
ContentEngine(config)# wccp spoof-client-ip enable
```

ステップ 7　Content Engine のグローバル コンフィギュレーション モードを終了します。

```
ContentEngine(config)# exit
```

ステップ 8　実行コンフィギュレーションを不揮発性メモリに書き込みます。

```
ContentEngine# write memory
```

これで、Content Engine に IP スプーフィングが設定されたので、残りのステップをすべて実行して、WCCP Version 2 ルータを IP スプーフィング用に設定します。

ステップ 9　ルータで WCCP Version 2 が有効になっていることを確認してください。

```
Router(config)# ip wccp version 2
```

ステップ 10　Web キャッシュ サービス（サービス 95）を実行するように WCCP ルータに指示します。

```
Router(config)# ip wccp web-cache
```

ステップ 11　WCCP ルータでサービス 95 を有効にします。

```
Router(config)# ip wccp 95
```

ステップ 12　設定するインターフェイスを指定し、インターフェイス コンフィギュレーション モードに入ります。

```
Router(config)# interface type number
```

ステップ 13　送信先 IP アドレスをハッシュするサービス（Web キャッシュ サービス）とともに、WAN インターフェイス（オリジン サーバに接続しているルータ インターフェイス）上の WCCP リダイレクションを有効にします（図 15-3 を参照）。

```
Router(config-if)# ip wccp web-cache redirect out
```

ステップ 14　送信元 IP アドレスをハッシュするサービス（サービス 95）とともに、LAN インターフェイス（クライアントに接続しているルータ インターフェイス）上の WCCP リダイレクションを有効にします（図 15-3 を参照）。

```
Router(config-if)# ip wccp 95 redirect out
```

ステップ 15　Content Engine に接続されているルータ インターフェイス上のトラフィック リダイレクションを無効にします（図 15-3 を参照）。

```
Router(config-if)# ip wccp redirect exclude in
```

Content Engine に接続されているルータ インターフェイス上のトラフィック リダイレクションを無効にして、ループバックを回避する必要があります。WCCP ルータが送信元 IP アドレスを含むパケットを Content Engine に送り返そうとするためです。

ステップ 16　グローバル コンフィギュレーション モードを終了します。

```
Router(config-if)# exit
```

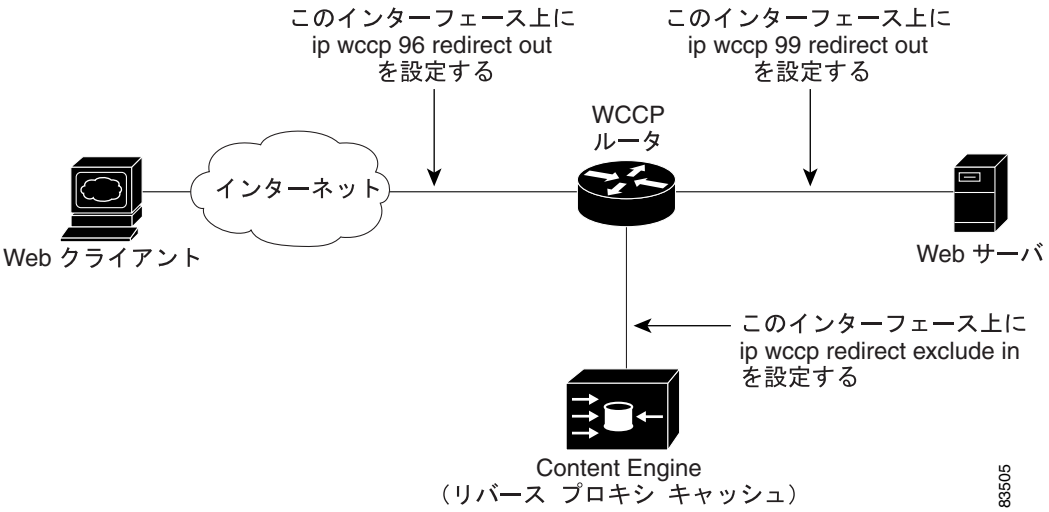
ステップ 17　ルータに設定を保存します。

```
Router# copy running-config startup-config
```

例 2：リバース プロキシ サーバによる IP スプーフィング

ここでは、IP スプーフィング用にスタンドアロン Content Engine と WCCP Version 2 ルータを設定する一例を示します。この例では、Content Engine がリバース プロキシ サーバとして動作します（図 15-4 を参照）。

図 15-4　リバース プロキシ サーバによる IP スプーフィング



スタンドアロン Content Engine（リバース プロキシ サーバ）と WCCP Version 2 対応ルータで IP スプーフィングを設定する手順は、次のとおりです。

ステップ 1　Content Engine（リバース プロキシ サーバとして動作）で WCCP Version 2 を有効にします。

```
ContentEngine(config)# wccp version 2
```

ステップ 2　Content Engine にルータ リストを設定します。このケースでは、ルータ リスト番号 1 に 1 つのルータ（10.10.20.1 の IP アドレスを持つ WCCP Version 2 対応ルータ）しかありません。

```
ContentEngine(config)# wccp router-list 1 10.10.20.1
```

ステップ 3　ポート 80 を介してポート リスト 1 を WCCP サービスに関連付けます。

```
ContentEngine(config)# wccp port-list 1 80
```

ステップ 4　Content Engine でユーザ定義の WCCP サービス（サービス 96）を設定します。この WCCP サービス用に、ルータ リスト、ポート リスト、ハッシング パラメータを送信先 IP アドレスと送信元ポートに関連付けます。

```
ContentEngine(config)# wccp service-number 96 router-list-num 1 port-list-num 1
application cache hash-destination-ip match-source-port
```

（注） Content Engine クラスタを使用していて、このクラスタに重み付け割り当てを使用している場合は、発信パケットと着信パケットの両方に対して、IP スプーフィングに割り当てられたサービス グループに対する重み付け割り当てがすべての Content Engine 上で等しいことを確認して、TCP 接続の中断を回避する必要があります。必要に応じて、これらのサービス グループに対して重み付けを割り当てるには、**wccp service-number** *servnumber* **router-list-num** *num* **port-list-num** *port* **application cache weight** *percentage* コマンドを使用します。デフォルトでは、Content Engine クラスタは、IP スプーフィングをオンにして適切なハッシュを行っており、サービス グループに対する重み付けの割り当ては不要です。

ステップ 5　Content Engine（リバース プロキシ サーバとして動作）が HTTP リバース プロキシ トラフィックを受け入れていることを、指定のルータ リスト（たとえば、ルータ リスト番号 1）内の WCCP Version 2 対応ルータに知らせます。

```
ContentEngine(config)# wccp reverse-proxy router-list-num 1
```

ステップ 6　Content Engine のクライアント IP スプーフィングを有効にします。

```
ContentEngine(config)# wccp spoof-client-ip enable
```

ステップ 7　グローバル コンフィギュレーション モードを終了します。

```
ContentEngine(config)# exit
```

ステップ 8　実行コンフィギュレーションを不揮発性メモリに書き込みます。

```
ContentEngine# write memory
```

ステップ 9　次のようにして、IP スプーフィング用にルータを設定します。

a. リバース プロキシ サービス（サービス 99）を有効にします。

```
Router(config)# ip wccp 99
```

b. 設定するインターフェイスを指定し、インターフェイス コンフィギュレーション モードに入ります。

```
Router(config)# interface type number
```

c. クライアントに接続しているインターフェイス上の WCCP サービス番号 96（送信先 IP アドレス上でハッシュするサービス）用にリダイレクションを設定します。

```
Router(config-if)# ip wccp 96 redirect out
```

d. オリジン サーバに接続しているインターフェイス上のリバース プロキシ サービス（サービス 99）用にリダイレクションを設定します。

```
Router(config-if)# ip wccp 99 redirect out
```

ステップ 10 Content Engine（リバース プロキシ サーバとして動作）に接続しているルータ インターフェイス上の WCCP リダイレクションを無効にします。

```
Router(config-if)# ip wccp redirect exclude in
```

ステップ 11 グローバル コンフィギュレーション モードを終了します。

```
Router(config-if)# exit
```