CHAPTER 1

アプリケーション アクセラレーションおよび最適化の概要

この章では、従業員の生産性を向上し、顧客維持率を増大し、オンライン収益を増加させるために、Cisco 4700 シリーズ Application Control Engine（ACE）Appliance 上でアプリケーション アクセラレーションおよび最適化を使用して、企業アプリケーションの処理を加速する方法について説明します。

ここでは、ACE の各種のアプリケーション アクセラレーションおよび最適化の機能について説明します。具体的な内容は、次のとおりです。

（注）　ACE でのアプリケーション アクセラレーションのパフォーマンスは、スループットが 50 ～ 100 Mbps です。典型的なページ サイズおよびブラウザの使用パターンでは、このスループットはおよそ 1,000 個の同時接続数に相当します。それ以降の接続では、アプリケーション アクセラレーション エンジンはバイパスされます。この制限は、アプリケーション アクセラレーション処理（FlashForward、デルタ最適化など）の対象として明示的に設定されたトラフィックだけに適用されます。アプリケーション アクセラレーション処理の対象として設定されていないトラフィックは、これらの制限を受けません。また、ACE の HTTP 圧縮はハードウェアに個別に実装されているので、これらの制限は適用されません。たとえば、アプリケーション アクセラレーションが有効なトラフィックと、アプリケーション アクセラレーションが無効なトラフィックが混在する場合、前者は制限を受けますが、後者は制限を受けません。アプリケーション アクセラレーションが有効なトラフィックが 50 Mbps の場合でも、ACE では、アプリケーション アクセラレーションが無効なトラフィックのスループットを最大 1.9 Gbps まで実現できます。

最適化トラフィック ポリシーおよび一般的な設定フロー

この章の各セクションに記載されている各種のアプリケーション アクセラレーションおよび最適化の機能を ACE に定義して実行するには、次のソフトウェア コンポーネントを設定する必要があります。

- 最適化 HTTP アクション リスト
- 最適化 HTTP パラメータ マップ
- レイヤ 7 HTTP 最適化ポリシー マップ
- レイヤ 7 サーバ ロード バランシング クラス マップおよびポリシー マップ
- レイヤ 3 およびレイヤ 4 サーバ ロード バランシング クラス マップおよびポリシー マップ
- レイヤ 3 およびレイヤ 4 最適化ポリシー マップ

次に、ACE に各種のアプリケーション アクセラレーションおよび最適化の機能を実装する場合の設定プロセスについて、一般的な手順の概要を示します。

ステップ 1　**action-list type optimization http** コマンドを使用して、最適化 HTTP アクション リストを作成します。最適化 HTTP アクション リストにより、ACE で実行する一連のアプリケーション アクセラレーションおよび最適化の各処理をグループ化します。たとえば、1 つまたは複数のアクション リストに、次の最適化機能を指定できます。

- デルタ最適化
- FlashForward
- FlashForward オブジェクト アクセラレーション
- Just-in-time オブジェクト アクセラレーション
- アダプティブ ダイナミック キャッシング
- キャッシュ最適化

詳細については、第 2 章「最適化 HTTP アクション リストの設定」を参照してください。

ステップ 2　（オプション）**parameter-map type optimization http** コマンドを使用して、最適化 HTTP パラメータ マップを作成します。最適化 HTTP パラメータ マップにより、関連アクション リストの選択内容に基づいて、いくつかの最適化テクノロジーを調整または制御するアプリケーション アクセラレーションおよび最適化のパラメータを識別します。たとえば、関連アクション リストに指定した機能に応じて、1 つまたは複数のパラメータ マップに次の最適化機能を設定できます。

- FlashForward パラメータ
- ベースファイル パラメータ
- キャッシュ パラメータ
- キャッシュポリシー パラメータ
- 標準 URL 文字列
- デルタ最適化のモードおよびパラメータ
- スクリプト言語
- クライアント ブラウザのオブジェクトのフレッシュネス期間
- リベース パラメータ
- cache-ttl 変更のロードしきい値

詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

ステップ 3　レイヤ 7 Server Load Balancing（SLB; サーバ ロード バランシング）クラス マップを作成し、レイヤ 7 ポリシー マップと関連付けます。レイヤ 7 SLB クラス マップおよびポリシー マップは、クッキー、HTTP ヘッダー、または送信元 IP アドレスなど、指定した SLB 基準に一致するトラフィックを判別するためのフィルタとして動作します。

詳細については、第 4 章「HTTP 最適化に関するトラフィック ポリシーの設定」の「SLB に関するレイヤ 7 クラス マップおよびポリシー マップの設定」（p.4-6）を参照してください。

ステップ 4　**policy-map type optimization http first-match** コマンドを使用して、レイヤ 7 最適化 HTTP ポリシー マップを作成します。最適化 HTTP ポリシー マップにより、特定のアプリケーション アクセラレーションおよび最適化の処理を設定できる

最適化 HTTP アクション リストをアクティブにします。最適化 HTTP ポリシー マップにオプションの最適化 HTTP パラメータ リストを指定して、アクション リストとパラメータ マップのアソシエーションを識別できます。この場合、最適化 HTTP アクション リストで実行する処理を定義し、最適化 HTTP パラメータ マップでアプリケーション アクセラレーション処理の実行方法の詳細を定義します。

詳細については、第 4 章「HTTP 最適化に関するトラフィック ポリシーの設定」の「SLB に関するレイヤ 7 クラス マップおよびポリシー マップの設定」（p.4-6）を参照してください。

ステップ 5　レイヤ 3 およびレイヤ 4 のクラス マップを作成します。レイヤ 3 およびレイヤ 4 のクラス マップには、Virtual IP（VIP; バーチャル IP）アドレス、プロトコル、ACE ポートなど、ACE をパススルーできるネットワーク トラフィックを分類するための一致基準が含まれます。ACE は、これらのレイヤ 3 およびレイヤ 4 トラフィック クラスを使用して、SLB を実行します。

詳細については、第 4 章「HTTP 最適化に関するトラフィック ポリシーの設定」の「SLB に関するレイヤ 3 およびレイヤ 4 クラス マップの設定」（p.4-13）を参照してください。

ステップ 6　レイヤ 3 およびレイヤ 4 のポリシー マップを作成して、VIP に関連付ける SLB 処理を指定します。また、ACE で実行するアプリケーション アクセラレーションおよび最適化サービスを設定します。これにより、関連する最適化 HTTP アクション リストに指定された機能とパラメータ マップに、特定の VIP がバインドされます。

詳細については、第 4 章「HTTP 最適化に関するトラフィック ポリシーの設定」の「SLB およびアプリケーション アクセラレーションに対応するレイヤ 3 およびレイヤ 4 ポリシー マップの設定」（p.4-14）を参照してください。

ステップ 7　ACE が受信したトラフィックのフィルタリング方法として **service-policy** コマンドを使用し、特定の VLAN インターフェイスを関連付けるか、またはすべての VLAN インターフェイスをグローバルに関連付けて、レイヤ 3 およびレイヤ 4 のポリシー マップをアクティブにします。

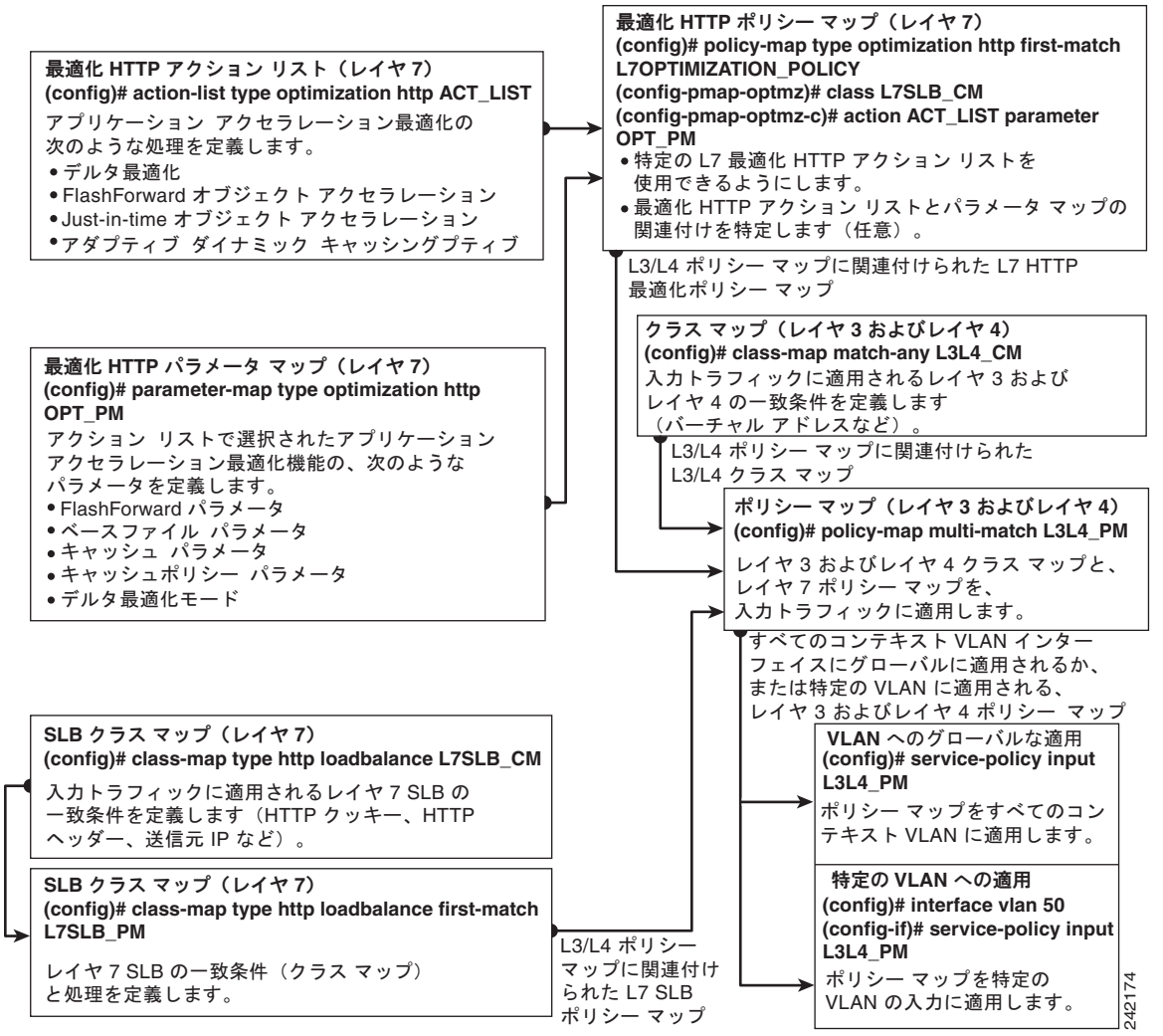
詳細については、第 4 章「HTTP 最適化に関するトラフィック ポリシーの設定」の「サービス ポリシーの適用」（p.4-18）を参照してください。

ステップ 8　**optimize** コマンドを使用して、最適化パラメータを定義します。特定の最適化パラメータをグローバル レベルで適用することにより、オプションの Cisco AVS 3180A Management Station にアプリケーション アクセラレーションおよび最適化の統計ログ情報をアップロードしたり、組み込みオブジェクト URL のホスト名の前にグローバル プレフィクスを挿入するなどの処理を実行できます。

詳細については、第 5 章「グローバル最適化値の設定」を参照してください。

図 1-1 に、レイヤ 7、レイヤ 3、およびレイヤ 4 のアクション リスト、パラメータ マップ、および ACE が SLB とアプリケーション アクセラレーションに使用するトラフィック ポリシーを構築して適用するために必要な基本的なプロセスの概要を示します。この図には、ポリシー設定の各種コンポーネントを相互に関連付ける方法も示されています。

図 1-1　アプリケーション アクセラレーションおよび最適化の設定フロー チャート



次に、ACE の各種アプリケーション アクセラレーションおよび最適化の機能を実装する実行コンフィギュレーションとなる設定例を示します。

```
# ====================================
# Backend web server farm
# ====================================
rserver SERVER1
   ip address 192.168.10.100
   inservice

rserver SERVER2
   ip address 192.168.10.101
   inservice

serverfarm SFARM1
   rserver SERVER1 80
      inservice
   rserver SERVER2 80
      inservice

policy-map type loadbalance first-match L7_SLB_POLICY
   match any_url  http url .*
   serverfarm SFARM1

# ================================================
# Application Acceleration-Oriented Configuration
# ================================================
class-map type http loadbalance match-any L7_BYPASS_CLASS
   match http url .*\acc_post\.html
   match http url .*\acc_nopost\.html
   match http url .*\acc_dbgtrace.*.js
   match http url .*\acc_appscope.*.js

class-map type http loadbalance match-any L7_FLASHFORWARD_CLASS
   match http url .*\.gif
   match http url .*\.css
   match http url .*\.js
   match http url .*\.class
   match http url .*\.jar
   match http url .*\.cab
   match http url .*\.txt
   match http url .*\.ps
   match http url .*\.vbs
   match http url .*\.xsl
   match http url .*\.xml
   match http url .*\.pdf
   match http url .*\.swf
```

```
class-map type http loadbalance match-any L7FLASHFORWARD-IMG-OPT_CLASS
   match http url .*\.jpg
   match http url .*\.jpeg
   match http url .*\.jpe
   match http url .*\.png

class-map type http loadbalance match-any L7_CM-SLB_CLASS
   match http url .*\.html
   match http url .*\.htm

class-map type http loadbalance L7_DELTA_CLASS
   match http url .*

parameter-map type optimization http L7_FLASHFORWARD_POLICY
   cache ttl min 0
   cache ttl max 60

action-list type optimization http FLASHFORWARD-OBJ_AL
   flashforward-object

action-list type optimization http FLASHFORWARD-IMG-OPT_AL
   flashforward-object

action-list type optimization http DELTA_AL
   delta
   flashforward

action-list type optimization http FLASHFORWARD_AL
   flashforward


policy-map type optimization http first-match L7_OPTM_POLICY
   class L7_FLASHFORWARD_CLASS
      action FLASHFORWARD-OBJ_AL parameter L7_FLASHFORWARD_POLICY
   class L7FLASHFORWARD-IMG-OPT_CLASS
      action FLASHFORWARD-IMG-OPT_AL parameter L7_FLASHFORWARD_POLICY
   class L7_CM-SLB_CLASS
      action FLASHFORWARD_AL
   class L7_DELTA_CLASS
      action DELTA_AL


# ================================
# LAYER 4 CONFIGURATION
# ================================
class-map match-any L4_VIP_CLASS
   match virtual-address 172.16.2.142 any
```

```
policy-map multi-match L4_OPTIMIZE-VIP
   class L4_VIP_CLASS
      loadbalance policy L7_SLB_POLICY
      loadbalance vip inservice
      optimize http policy L7_OPTM_POLICY

#
# Bind with VLAN
#
interface gigabitEthernet 1/2
   channel-group 2
   no shutdown

interface gigabitEthernet 1/3
   channel-group 3
   no shutdown

interface port-channel 2
   switchport access vlan 2
   no shutdown

interface port-channel 3
   switchport access vlan 3
   no shutdown

access-list ACL1 line 10 extended permit ip any any

interface vlan 2
   ip address 172.16.2.141 255.255.255.0
   access-group input ACL1
   service-policy input OPTIMIZE-VIP
   no shutdown

interface vlan 3
   ip address 192.168.10.141 255.255.255.0
   access-group input ACL1
   no shutdown
```

デルタ最適化

ACE を使用すると、企業のクライアント ブラウザのキャッシュを、コンテンツの差分（デルタ）によってダイナミックに、直接アップデートできます。これにより、ページのダウンロードが速くなり、従業員の生産性が向上し、オンライン収益が増大します。

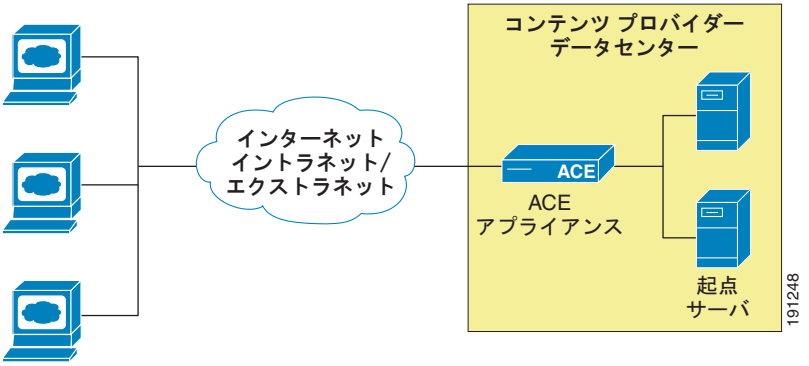
ほとんどの Web ページは、各要求に対して異なるコンテンツを生成できるようにダイナミックに作成されます。コンテンツの差分は、ページの内側および外側のペイロードの違い、またはより詳細な HTML コンテンツの変更（株価の変動、ニュースの見出しの変更など）によって発生します。ダイナミックに作成されるページは要求ごとに変化するので、従来のキャッシング ソリューションではキャッシュできません。Web キャッシングを使用しても、ユーザは文書を要求するごとに、起点サーバから完全な HTML 文書をダウンロードする必要があります。以降のダウンロードでの差分が、ページ全体のサイズに比べてわずかであっても同様です。

たとえば、ニュース サービスのホームページが 70 KB の HTML であるとします。このページは、要求ごとにペイロードが異なるので、ダイナミックに作成されます。また、その日のうちに、このページのニュースの見出しが変更されることもあります。デルタ最適化テクノロジーを使用しない場合、HTML ページの変更がわずかに 2 KB であっても、ユーザはこのページにアクセスするごとに 70 KB のページ全体をダウンロードする必要があります。

デルタ最適化テクノロジーにより、コンテンツ プロバイダーは ACE を使用して、後続のコンテンツ取得時におけるコンテンツの差分（デルタ）を、（必要に応じてユーザ単位で）ダイナミックに計算し、ダイナミック コンテンツへの後続のアクセスに対してデルタだけを送信することができます。ユーザは、ホームページへの初回アクセス時には 70 KB を取得しますが、以降のアクセスでは 2 KB のコンテンツ デルタを取得するだけで済みます。デルタ最適化を使用すると、以降に要求されたページ間の差分だけがユーザに送信されます。ダイナミック HTML によって符号化されるこれらのデルタによって、起点サーバによる従来のエッジ キャッシュのアップデートと同様に、ACE でクライアントのブラウザ キャッシュを直接アップデートできます。

図 1-2 に、ネットワーク トポロジにおける ACE の配置を示します。

図 1-2　単純な ACE トポロジ



この図で、ACE はコンテンツの起点サーバ（Web サーバ）からブラウザへのコンテンツ配信パス上の、サーバの近くに配置されています。ACE は、配信するコンテンツを観察して変更し、帯域幅を削減して、ユーザのダウンロード パフォーマンスを向上させます。

次のトピックでは、デルタ最適化における ACE のブラウザ、サーバ、およびコンテンツのサポートについて説明します。

- ブラウザのサポート（p.1-13）
- Web サーバのサポート（p.1-14）
- Web コンテンツのサポート（p.1-14）
- 設定可能なデルタ最適化モード（p.1-16）

ブラウザのサポート

ACE は、HTTP 1.0 および HTTP 1.1 プロトコルを使用するすべてのブラウザで送受信されるトラフィックをサポートします。HTTP 1.1 は、RFC 2068、「Hypertext Transfer Protocol -- HTTP/1.1」のセクション 14.40 に規定されている Transfer-Encoding: Chunked HTTP 要求ヘッダーを使用したチャンク処理をサポートしています。

ACE は、次バージョンでクッキーと JavaScript が有効であれば、Windows 98、Windows NT 4、Windows 2000、Windows ME、Windows XP 上のすべての主要ブラウザで完全な最適化をサポートします。

- Microsoft Internet Explorer バージョン 5.5、6.0 以降
- Netscape Communicator バージョン 4.8、6.2 以降（Solaris 2.8、2.9、および Redhat AS 3.0 用を含む）
- AOL ブラウザ バージョン 6.0 以降
- FireFox 2.0（Solaris 2.9 用を含む）

その他のブラウザおよび設定はすべて、デルタ最適化なしでサポートされます。

ユーザがページを要求すると、ACE はユーザのブラウザ タイプを識別し、そのブラウザでサポートされる最適化を適用します。

ACE は、ブラウザの製造元、バージョン、プラットフォーム、クッキー設定、および JavaScript 設定を自動的に検出するので、ブラウザでの設定の変更およびソフトウェアのインストールは不要です。

クッキーおよび JavaScript のサポートは、次のように検出されます。

1. ユーザのサイトへの初回アクセス時に、ACE はユーザに配信するページに JavaScript プローブ コードを挿入します。この JavaScript コードにより、ユーザのシステムに ACE クッキーが作成されます。
   - ブラウザが JavaScript をサポートしていない場合、JavaScript は無効になります。または、クッキーが無効の場合、クッキーは作成されません。
   - それ以外の場合、ACE クッキーが正常に作成されます。
2. ユーザのサイトへの 2 回めのアクセス時には、クッキーの存在によってブラウザが JavaScript とクッキーをサポートしていることが示され、ACE に対して、ユーザにデルタ最適化コンテンツを配信するように通知されます。

（注）　ACE のサポートは、ユーザが以降でブラウザを再設定して JavaScript を無効にした場合でも持続します。詳細については、「クッキーの使用方法」（p.1-34）を参照してください。

Web サーバのサポート

ACE は、すべてのプラットフォーム上で、HTTP 1.0 または 1.1 プロトコルをサポートしているすべての Web サーバおよび Web アプリケーション サーバをサポートします。HTTP 1.1 は、RFC 2068、「Hypertext Transfer Protocol -- HTTP/1.1.」のセクション 14.40 に規定されている Transfer-Coding: Chunked HTTP 要求ヘッダーを使用した HTTP 1.1 チャンク転送符号化をサポートしています。

Web コンテンツのサポート

ACE は、コンテンツまたはサーバ ソフトウェアを変更することなく、すべての Web コンテンツをサポートします。ただし、すべての Web コンテンツにデルタ最適化を適用できるわけではありません。凝縮できないコンテンツは、変更されずにパススルーされます。

次に、デルタ最適化サポートの制限について説明します。

- キャラクタ セットのサポート（p.1-15）
- サポートされない HTML 要素（p.1-15）

キャラクタ セットのサポート

ACE は 1 バイト文字だけをサポートしているので、デルタ最適化を適用できるのは ISO-8859-1 キャラクタ セットを使用している HTML コンテンツだけです。その他のタイプのコンテンツはすべて、凝縮しないでパススルーできます。

ACE は、Content-Type HTTP 応答ヘッダーを調べて、キャラクタ セットのタイプを判別します。このヘッダーは、Content-Type:text/html; charset=iso-8859-1 のように表示されます。キャラクタ セットが指定されていない場合、ACE は ISO-8859-1 であるとみなします。

サポートされない HTML 要素

特定のタイプの HTML コンテンツは、凝縮できません。インライン フレーム（IFRAME タグ）、include アクションを使用する外部スクリプト（作成者がコードを再使用できる SRC 属性を含む SCRIPT タグ）、および組み込みオブジェクト（OBJECT タグ）には、デルタ最適化を適用できません。これらの要素を含むページそのものはデルタ最適化されますが、ページ内のこれらの要素は凝縮されずにパススルーされます。

ページ内に組み込まれた JavaScript コードはすべて、凝縮されずにパススルーされ、オリジナル ページの正確な順序が保持されます。これは、JavaScript への依存と実行順序に関する問題を回避するためです。

ACE にデルタ最適化動作パラメータを設定して、デルタ最適化を適用しないキャッシュ可能オブジェクトを定義するには、パラメータ マップ最適化コンフィギュレーション モードで、**delta** コマンドを使用します。詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

設定可能なデルタ最適化モード

デルタ最適化を適用する Web ページがすべてのユーザに共通なのか、ユーザごとに個別なのかを指定するには、デルタ最適化モードを使用します。デルタ最適化モードの選択により、ACE によって生成されるページ デルタの種類が異なります。

ACE は、2 種類のデルタ最適化モードをサポートします。

- all-user（全ユーザ）モード
- per-user（ユーザ別）モード

all-user モードでは、URL の全ユーザが共有する単一のベース ファイルに対して、デルタが生成されます。all-user デルタ最適化モードは、ダイナミックな個別化コンテンツであっても、ページ構造が全ユーザに共通であれば、ほとんどの場合に使用できます。ディスク スペースのオーバーヘッドは最小限です（ディスク スペース要件は、ユーザ数ではなく凝縮するページ数に基づきます）。

per-user モードでは、特定のユーザが URL を要求すると、そのユーザ用に特別に作成されたベース ファイルに対して、応答のデルタが生成されます。per-user デルタ最適化モードは、ページのコンテンツ（レイアウト要素を含む）がユーザごとに異なる場合に使用します。このモードでは、最高レベルのデルタ最適化が実行されます。ただし、各ユーザに配信されるベース ページのコピーを ACE キャッシュに保管しておく必要があるので、ACE キャッシュ用のディスク スペース要件は増大します。per-user デルタ最適化モードでは、ベース ページがユーザ間で共有されないので、コンテンツのプライバシー保護に適しています。

ACE のデルタ最適化モードを制御するには、パラメータ マップ最適化コンフィギュレーション モードで **delta** コマンドを使用します（第 3 章「最適化 HTTP パラメータ マップの設定」を参照）。

アダプティブ ダイナミック キャッシング

アダプティブ ダイナミック キャッシングでは、アプリケーション サーバおよびデータベースをオフロードして、ACE でダイナミックな情報または個別化された情報の要求を満たすことができます。アダプティブ ダイナミック キャッシングを使用すると、アプリケーションの応答時間が著しく改善され、サーバの負荷が削減されて、同時ユーザ数が増大します。これにより、スケーラビリティが向上し、サーバのアップグレード コストが削減されます。パフォーマンスが保証されたキャッシング ポリシーにより、ACE でサーバ負荷をリアルタイムで監視し、インテリジェントなクローズド ループ方式でコンテンツの有効期限を決定することにより、トラフィック ロードのピーク時でもサイトのパフォーマンスとハードウェア リソースを最大限に活用できます。

アダプティブ ダイナミック キャッシングにより、ACE は、Active Server Page（ASP）スクリプトまたはアプリケーション サーバによって作成されたコンテンツなど、ダイナミックなコンテンツをキャッシュできます。ダイナミックなコンテンツは、生成されたコンテンツに最新データが含まれていることがあるので、通常はキャッシュされません。ただし、フレキシブルな設定とアルゴリズムに基づく処理により、ACE では、これらのコンテンツを特定の時間だけキャッシュしておくことができます。

アダプティブ ダイナミック キャッシングには、次の機能があります。

- キャッシュのパラメータ化 — URL およびクエリー パラメータだけでなく、より多くの要素で応答を差別化します。クッキー値、HTTP ヘッダー値、使用 HTTP 方式など、その他のパラメータが参照されます。この機能により、ACE では、指定されたキャッシュ パラメータに応じて、同じ URL の複数の応答をキャッシュできます。キャッシュのパラメータ化は、スタティック（FlashForward）およびダイナミックの両方のキャッシングに適用されます。
- 拡張有効期限ルール — 時間に基づくか、またはパフォーマンス保証によるロードベース有効期限により、キャッシュされたコンテンツの有効期限を自動的に設定します。
- デルタ キャッシュ — オリジナル HTML コンテンツがダイナミック キャッシュにあり、リベースが実行されなかった場合、デルタ コンテンツをキャッシュ（メモリ専用オブジェクト）に保管します。

アダプティブ ダイナミック キャッシングには、各種の状況に自動的に適用されるデフォルトの設定はありません。この機能は、Web サイトまたはアプリケーションごとに特別に設定する必要があります。アダプティブ ダイナミック

キャッシングを HTTP 最適化パラメータ マップに設定し、最適化 HTTP アクション リストに関連付けることにより、レイヤ 7 サーバ ロード バランシング ポリシーに指定した各種 URL セットに、それぞれ異なるキャッシングを適用できます。

ダイナミック キャッシングの使用方法、および実装と設定に必要な手順については、「ダイナミック キャッシングの設定」（p.1-18）を参照してください。

ダイナミック キャッシングの設定

ACE を配置する前に、トラフィック パターンを分析して、ACE の各種機能（FlashForward やアダプティブ ダイナミック キャッシングなど）の適合性を判別することを推奨します。最適な設定はさまざまな方法で判別できますが、ダイナミック キャッシングを設定する場合には、次のガイドラインに従ってください。

- ダイナミック キャッシングのガイドライン（p.1-18）
- ダイナミック キャッシングの設定（p.1-20）

対応する URL のキャッシュ最適化をイネーブルにする方法については、第 2 章「最適化 HTTP アクション リストの設定」を参照してください。

パラメータ マップのダイナミック キャッシング設定の詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

ダイナミック キャッシングのガイドライン

ダイナミック キャッシングは、次の状況で使用します。

- ダイナミック　キャッシングは、たとえ短い時間であっても、一定時間内に応答を再使用できる場合に限り、有効です。
- ダイナミック　キャッシングは、次のように、キャッシュした応答を、その応答が期限切れになる前に 2 回以上使用できる場合に限り、有効です。
  - 同じ応答を一定時間内に複数のユーザに使用できる場合、つまり、コンテンツを共有できる場合。この機能は、たとえキャッシュの有効期限が短くても、その期限内に複数のユーザが同じ応答にアクセスする状況であれば使用できます。たとえば、階層ディレクトリのブラウザ アプリケーション（カタログ ストアまたは文書管理アプリケーションなど）で、ディレクトリがダイナミックに生成されても全ユーザに同じ表示が提

供されるような場合です。ダイナミック キャッシングを使用すると、各ユーザに対してディレクトリ ページを再生成する必要がありません。

- 応答を使用できるのが（個別化された）単一ユーザであっても、そのユーザが複数回にわたってアクセスする場合。同じ応答が、そのユーザの複数のブラウザ セッションで使用される状況です。たとえば、1 回のセッションで生成された応答を 2 回めのセッションで変更せずに使用できる場合、または応答が特定のセッションに連携していて単一ブラウザ セッション内で再使用される場合などです。

- ダイナミックにキャッシュされる応答は、次の 1 つまたは複数の方法で識別されます。
  - URL が正規表現と一致している
  - 特定のクエリー パラメータが存在する
  - 要求内に特定のクッキーが存在する
  - 要求内に特定の HTTP ヘッダーが存在する

  たとえば、特定のアプリケーションが各種の動作に同じ URL を使用している場合、その一部だけをキャッシュできます。アプリケーションの URL が、次のような場合、

  1. http://xyz.com/doit.jsp?action=login&username=xyz
  2. http://xyz.com/doit.jsp?action=browse&level=1
  3. http://xyz.com/doit.jsp?action=browser&level=2

  URL はすべて同じですが、（1）への応答はキャッシュできません。（2）および（3）への応答はキャッシュできます。

次の状況では、ダイナミック キャッシングを使用しないでください。

- 応答が、配信後すぐに古くなる場合。たとえば、次のような場合です。
  - 応答に設定されるクッキーがそのセッションに固有である。たとえば、ログイン ページへの応答は特定のセッションに固有のものです。
  - 応答に、セッションの前の動作に固有のデータが含まれている。たとえば、実行したばかりのトランザクションの確認番号はキャッシュできません。
- 応答の有効期限を判別できない場合。応答に、以降の動作に基づいてアップデートされるデータが含まれている場合です。たとえば、証券会社のユーザのポートフォリオ ページは、そのユーザが取り引きを実行しないと変更されません。

- 同じ応答のバージョン違いは、URL、URL クエリー パラメータ、または要求内のクッキーでは識別できません。たとえば、応答にユーザ名などの個人設定が含まれていても、URL クエリー パラメータまたはクッキーでは直接ユーザを識別できない場合です。起点サーバで個別化された応答（セッション クッキーなど）を生成するには、要求によってユーザを識別できる必要があります。ユーザ セッションごとに異なるページのバージョンは、ダイナミックにキャッシュできます。ただし、この状況でダイナミック キャッシングが有効になるのは、ユーザが同じブラウザ セッション中にそのページに 2 回以上アクセスする場合だけです。

ダイナミック キャッシングの設定

ページにダイナミック キャッシングを適用できることを確認したら、ここで説明する手順に従って、ACE に適切なダイナミック キャッシングを設定します。

ACE にダイナミック キャッシングを設定する手順は、次のとおりです。

ステップ 1　レイヤ 7 サーバ ロード バランシング クラス マップの URL と一致する正規表現により、1 つ以上の URL を識別します。この手順で URL の短いリストを作成すると、次のような目的に使用できます。

- Accelerometer または他のパフォーマンス テスト プログラムを使用して、LAN 上でパフォーマンス テストを実行します。LAN を使用するのは、ネットワーク遅延の問題ではなく、サーバ遅延の問題を判別するためです。
- エラー ログを調べて、起点サーバの応答時間が長いエントリを見つけます。応答時間の長さは、エンド ユーザによるページのダウンロード時間を意味します。たとえば、LAN 上でのページのダウンロードに 10 秒かかる場合、起点サーバの応答時間が 1 秒以上である個々のコンポーネントのダウンロードを調べる必要があります。

ステップ 2　URL と一致する正規表現を実行して、ダイナミックにキャッシュできる URL のインスタンスを識別します。正規表現が多数の応答と一致する場合、ダイナミックにキャッシュできる応答と、できない応答があります。キャッシュできる応答とキャッシュできない応答を区別できるかどうかを判別します。特定のクエリー パラメータ、クッキー、または HTTP ヘッダーの有無を確認してください。

ステップ 3　応答のバージョンを識別します。ダイナミックにキャッシュできるインスタンスを識別したら、キャッシュのパラメータ化を使用して、キャッシュする応答のバージョン数を推測します。要求パラメータと応答コンテンツを調べ、応答のコンテンツを定義できるものを判別します。次の要求パラメータを調べます。

- URL およびそのコンポーネント
- クエリー パラメータ
- クッキー
- HTTP ヘッダー値

ステップ 4　応答のコンテンツを個別に判別できる最小限のパラメータ セットを決定します。この応答に基づいて、次のように、この URL の最適化パラメータ マップに **cache key-modifier** コマンドおよび **cache parameter** コマンドを定義し、キャッシュ キーを修正します（第 3 章「最適化 HTTP パラメータ マップの設定」を参照）。

- 応答を、URL およびすべてのパラメータによって個別に判別できる場合。これはデフォルトのキャッシュ キーなので、何も実行する必要はありません。
- 応答を、一部のパラメータだけで判別できる場合。**cache parameter** コマンドを使用して、これらのパラメータを指定します。たとえば、サイトの URL が次のような場合、

  ```
  http://xyz.com/dosomething.asp?action=browse&dir=x&session=13345.
  ```

  応答のコンテンツを、**action** および **dir** クエリー パラメータによって判別できます。**session** クエリー パラメータは、応答には影響しません。したがって、**cache parameter** コマンドを次のように指定します。

  ```
  host1/Admin(config-parammap-optmz)# cache parameter
  $http_query_param(action)
  host1/Admin(config-parammap-optmz)# cache parameter
  $http_query_param(dir)
  ```

- 応答を、いくつかのクッキー値で判別できる場合。次のように、最適化パラメータ マップにこれらの値を追加します。

  ```
  host1/Admin(config-parammap-optmz)# cache parameter
  $http_cookie(foo)
  ```

- URL の一部を無視する必要がある場合。たとえば、一部のサイトの URL にはセッション ID が組み込まれていますが、応答はセッション ID には依存しません。この場合、最適化パラメータ マップの **cache key-modifier** コマンドで、URL のサブ表現を使用できます。たとえば、サイトの URL が次の形式の場合、

  ```
  http://xyz.com/sess12345/dosomething.asp?action=browse&dir=x.
  ```

  sess12345 が応答と無関係であれば、次のように（レイヤ 7 SLB クラス マップに）URL 一致の正規表現を指定し、**cache key-modifier** コマンドを使用して、キャッシュ キーから（sess*NNNNN*）の文字列を除外します。

  ```
  host1/Admin(config)# class-map type http loadbalance match-any
  Example_Classmap
  host1/Admin(config-cmap-http-lb)# match http url
   (.*)/(sess.*)/(dosomething.asp)
  host1/Admin(config-cmap-http-lb)# exit
  host1/Admin(config)# parameter-map type optimization http
  OPTIMIZE_PARAM_MAP1
  host1/Admin(config-parammap-optmz)# cache key-modifier $(1)/ $(3)
  ```

ステップ 5　コンテンツを考慮して、応答をキャッシュしておく時間を識別します。たとえば、新しいアイテムは 3 時間後には古くなることがあります。

cache ttl タイミング関連最適化パラメータ マップ コマンドを使用して、次のように、Time-To-Live（TTL; 存続可能時間）の最小値と最大値を秒数で設定できます。

- 最小 TTL — コンテンツをキャッシュしておく最小時間を指定します。これが、コンテンツのライフタイムになります。新しいアイテムを 3 時間有効にしておく場合、この値は 3 × 60 × 60 = 10800 秒です。
- 最大 TTL — オブジェクトの最小 TTL の経過後、ACE による処理方法を設定します。

オブジェクトの有効期限は、パフォーマンス保証によるロードベース有効期限を使用して設定することもできます。この場合、起点サーバからの現在の応答時間（短いタイム ウィンドウの平均値）が平均応答時間（長いタイム ウィンドウの平均値）よりも大きい場合、しきい値によって TTL 設定をダイナミックに増分できます。同様に、この逆の状況では、TTL 値がダイナミックに減少します。

キャッシュ有効期限の設定の詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

FlashForward オブジェクト アクセラレーション

FlashForward オブジェクト アクセラレーションは、ACE アプライアンスの使用帯域幅削減とダウンロード アクセラレーションの利点を、HTML ページ内に組み込まれているオブジェクトにまで拡張して適用します。この機能は、ローカル オブジェクト ストレージと組み込みオブジェクトのダイナミックなリネームを統合することにより、親 HTML ページ内のオブジェクトのフレッシュネスを保持します。

FlashForward は、ACE 内でスタティック オブジェクトをリネームしてキャッシュし、組み込みオブジェクトの HTTP ヘッダーを変更してブラウザ キャッシュ内での有効性を延長し、リネームされたオブジェクトが参照されるように親 HTML コードの URL 参照を変更して、クライアントが HTML 要求の時点で最新オブジェクトまたはアップデートされたオブジェクトだけを要求できるようにします。この技術により、クライアントのページ ダウンロード時間が著しく短縮され、オブジェクト検証要求に関連するアップストリーム トラフィックが削減されます。

FlashForward は、イメージ、スタイル シート、JavaScript ファイルなど、組み込み Web オブジェクトに関連するネットワーク遅延を削減します。ACE を使用しない場合、ユーザがグラフィック イメージのあるページをロードすると、各オブジェクトについてユーザが最新バージョンを所有しているかどうかを確認する必要があるので、遅延が発生します。オブジェクト検証により、20 KB 以上の不要なアップストリーム トラフィックが使用されます。各検証には、クライアントからサーバへの HTTP 要求が含まれます。FlashForward は、サーバ側での組み込みオブジェクトのバージョン管理を強制します。すべてのオブジェクトの有効性情報が親 HTML ドキュメントの 1 回のダウンロードで伝送されるので、不要な検証要求は発生しません。

Web の現在のオブジェクト フレッシュネス検証メカニズムの場合、以降のセッションでは、サーバからクライアントにオブジェクトの有効性が明示的に伝送されるまで、クライアントのブラウザにキャッシュされているオブジェクトはすべて無効（古い）とみなされます。このアプローチでは、以前にキャッシュしたページへの以降のアクセス時に、クライアントが各オブジェクトについて検証要求を発行する必要があるので、ページのロードに著しい遅延が発生します。クライアントとサーバ間の往復伝送が完了するまではオブジェクトをレンダリングできないので、複数のオブジェクトが組み込まれているページの場合、ロード遅

延はきわめて大きくなります。また、このアプローチではアップストリーム帯域幅が大幅に消費されます。

FlashForward は、オブジェクトのフレッシュネス検証の責任を、クライアントではなく ACE に配置し、プロセスを逆にすることによって、効率を改善します。FlashForward では、クライアントは最新オブジェクトだけを要求し、ACE によって有効と判別されたブラウザ キャッシュ内のオブジェクトについては、検証要求を発行しません。デルタ最適化（「最適化トラフィック ポリシーおよび一般的な設定フロー」[p.1-3] を参照）との併用により、小さなデルタ ページによって新規オブジェクトを参照する情報が配信されます。FlashForward を使用すると、クライアントはブラウザにキャッシュされているオブジェクトのフレッシュネスを起点サーバに対して検証する必要がないので、ページのダウンロードが著しく加速され、オブジェクト検証要求に関連するアップストリームおよびダウンストリームの両方のトラフィックが削減されます。

FlashForward の動作の詳細については、「FlashForward の動作」（p.1-44）を参照してください。

FlashForward を設定するには、アクション リスト最適化モードで、**flashforward** コマンドまたは **flashforward-object** コマンドを使用します（第 2 章「最適化 HTTP アクション リストの設定」を参照）。

Just-in-time オブジェクト アクセラレーション

Just-in-time オブジェクト アクセラレーションでは、キャッシュできない組み込みオブジェクトのアクセラレーションが可能になるので、アプリケーションの応答時間が改善されます。この機能により、ユーザは各要求で、これらのオブジェクトをダウンロードする必要がなくなります。代わりに、ACE によって各オブジェクトのフレッシュネスがリアルタイムで自動的に追跡されます。要求されたオブジェクトが変更されていなければ、ACE はクライアントに対して、キャッシュされているオブジェクトのバージョンを使用するように指示します。オブジェクトが変更されている場合には、ACE からクライアントに最新オブジェクトが配信されます。ACE は、オブジェクトが変更されている場合に限りオブジェクトを配信するので、すべてのユーザに対して最適なアプリケーション応答時間が保証されます。

イメージなどのスタティック コンテンツは ACE の FlashForward によって処理され、ダイナミックな HTML は ACE のデルタ最適化によって処理されます。Just-in-time オブジェクト アクセラレーションは、次の状況のように、デルタ最適化で処理できないダイナミック コンテンツに対して効果的です。

- HTML コンテンツがダイナミックで、凝縮可能な最大ページ サイズ（250 KB）を超えている
- コンテンツが起点サーバによって期限切れ、またはキャッシュ不能としてマークされている

Just-in-time オブジェクト アクセラレーションは HTTP 圧縮と併用することを推奨します（『*Cisco 4700 Series Application Control Engine Appliance Server Load-Balancing Configuration Guide*』を参照）。

Just-in-time オブジェクト アクセラレーションは、次のように機能します。

1. ブラウザがオブジェクトを要求し、ACE がブラウザの要求を代行してオブジェクトを取得します。
2. ACE が、コンテンツの MD5 ハッシュを使用して ETag（エンティティ タグ）ヘッダーを構築して挿入し、オブジェクトをブラウザに送信します。ETag（エンティティ タグ）は、同じオブジェクトの異なるバージョンを識別するために使用する要求ヘッダーです。

3. ブラウザからの以降の要求にはすべて、この ETag ヘッダーが含まれています。ACE は最新の起点サーバ コンテンツの MD5 ハッシュを再計算してヘッダーと比較し、次のように処理します。
   - コンテンツが変更されていない場合、ACE は 304（Not Modified）応答を戻し、データは配信しません。
   - コンテンツが変更されている場合、ACE は新しいコンテンツを取得して、ETag を再設定します。

Just-in-time オブジェクト アクセラレーションを設定するには、アクション リスト最適化モードで **dynamic etag** コマンドを使用します（第 2 章「最適化 HTTP アクション リストの設定」を参照）。

ベース ファイルの管理

ACE は、パフォーマンスを最適化するためにキャッシング メカニズムを使用します。ベース ページが自動的かつ透過的にキャッシュされ、キャッシュが満杯になると、最も使用されていないページが廃棄されます。

ベース ファイル選択ポリシーは、標準 URL 機能（「標準 URL」[p.1-30] を参照）と同様ですが、URL グループで共有するベース ファイルをより柔軟に指定できます。ベース ファイル選択ポリシーにより、URL の一般化方法を定義する正規表現を識別できます。たとえば、Amazon.com は次のような URL を使用します。

http://www.amazon.com/exec/obidos/tg/browse/-/289728/ref=k_kh_ln_bp_2/105-3394538-7390300

http://www.amazon.com/exec/obidos/tg/browse/-/289737/105-3394538-7390300

これらの URL は、一般的な疑問符（?）文字ではなく、パス内の ID 番号によってパラメータ化されます。この例の共通 URL を http://www.amazon.com/exec/obidos/tg/browse/-/ として定義し、このベース URL 用のベース ファイルが作成されるように、ベース ファイル選択ポリシーを設定できます。同様の要求はすべて、同じベース ファイルを共有します。

匿名ベース ファイル

ACE には、ユーザのプライバシーに対処するための匿名ベース ファイル機能が組み込まれています。この機能は、all-user デルタ最適化モード オプション（「設定可能なデルタ最適化モード」[p.1-16] を参照）の 1 つで、ACE ノードを使用して、オンラインでの商取引、バンキング、業務会計など、個人の機密情報を配信できます。ユーザは通常、SSL と組み合わせてこの機能を使用することによって、セキュリティとプライバシーが保護された凝縮型コンテンツを配信できるようになります。

多数のユーザに共通する情報は、一般的に機密情報ではなく、特定ユーザの情報でもありません。逆に、特定ユーザに固有の情報や、少数ユーザだけに共通する情報は、機密情報またはユーザ固有の情報になります。この機能を使用すると、ACE は、多数のユーザに共通する情報だけが含まれているベース ファイルを作成して、配信します。特定のユーザ（または少数のユーザ グループ）に固有の情報は、匿名ベース ファイルに含まれません。匿名ベース ファイルでは、ベース ファイル内のデータのコンテキストが除外されるので、情報を特定のユーザに関連付けることは不可能です。匿名ベース ファイルの場合、特定の URL への初回アクセス時には、初期情報（潜在的な機密情報）は表示されません。

たとえば、**m** および **n** の 2 つの数値があり、**m** は匿名性レベル、**n** はベース ファイルのサンプル サイズを表しているとします。匿名ベース ファイル機能は、**n** ベース ファイル（およびユーザ）から **m** だけに共通する情報を含む共有ベース ファイルを作成します。たとえば、**m** = 4、**n** = 20 の場合、匿名ベース ファイルには 20 のユーザ特定ベース ファイルのうち、最低 4 つだけに共通するコンテンツが含まれます。20 のベース ファイルのいずれか 1 つに固有のコンテンツは、匿名ベース ファイルには含まれません。また、4 つより少ないベース ファイル数の共通コンテンツも含まれません。これらの 20 のベース ファイルは、この機能をイネーブルにするためだけに作成される per-user タイプのファイルです（これらの per-user ベース ファイルはコンテンツの凝縮には使用されません）。ベース ファイル サンプル サイズ内のベース ファイルは、固有ブラウザからの特定の URL への最初の **n** の固有要求として選択されます。

匿名性レベル（**m**）として、**m**= 0（どの匿名レベルもイネーブルにしない）を設定できます。この場合、ACE は、**n** = 最大値（5、3**m**）を選択し、高度な匿名ベース ファイルを作成します。つまり、**n** は 5 または 3 × **m** の大きい方の値に設定されます。

匿名ベース ファイル機能は、次のパラメータ マップ最適化モード コマンドを使用して設定できます。

- **basefile anonymous-level — m** に相当する追加のベース ファイル匿名レベルを指定します。
- **delta all-user —** all-user デルタ最適化モードを指定します。

これらのコマンドの詳細は、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

クラスベースの凝縮

クラスベースの凝縮では、URL クラスと呼ばれる複数の URL 間で、共通ベース ファイルを共有できます。このモードは、凝縮する各 URL について個別のベース ファイルが保持される標準 URL-based モードとは異なります。

クラスベースの凝縮では、同様のレイアウトまたはコンテンツを持つ Web ページのクラスを定義できます。特定のページのクラスを作成すると、クラス内の各ページは、そのクラスの全ページに共通する 1 つのマスター ベース ページに対して凝縮されます。ある文書を、URL ベースの凝縮のように、以前にダウンロードした同じ文書のバージョンに対して凝縮するのではなく、以前に取得した同様の文書に対して凝縮できます。

URL クラスは、クラス内のすべての URL と一致する正規表現を指定することによって定義します。たとえば、http://host/thisdir/* という表現を指定すると、このパス内の全ファイルが 1 つのクラスとしてグループ化されます。このパスに、http://host/thisdir/first.html および http://host/thisdir/second.html という 2 つのファイルが含まれている場合、これらは共通のベース ファイルを共有します。

標準 URL

ACE では、標準 URL 関数を使用して、パラメータ化された要求を変更し、疑問符（?）とその後ろに続く、URL の汎用部分を特定する文字を排除します。その後、この汎用 URL を使用して、ベース ファイルを作成します。ACE では、標準 URL を使用して、複数のパラメータ化された URL を 1 つの標準 URL にマッピングします。

ベース ファイルの選択を指定するには、パラメータ マップ最適化モードで **canonical-url** コマンドを使用し、標準 URL の正規表現を含む文字列を指定します。この文字列は、各種の実 URL を一致させるために使用する正規表現です。一致したすべての URL で 1 つのベース ファイルを共有します。**canonical-url** コマンドには、文字列を評価するパラメータ拡張関数を使用できます。詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。表 3-3 に、使用できるパラメータ拡張関数が示されています。

たとえば、次の 2 つの URL は、いずれも http://www.servers.com/books という URL に短縮されます。

http://www.servers.com/books?id=235

http://www.servers.com/books?id=576

その結果、両方のパラメータ化 URL は、http://www.servers.com/books という標準 URL に基づく同じベース ファイルを共有します。ただし、元の 2 つの URL が共通するコンテンツまたはレイアウトがほとんどない 2 ページを参照している場合、凝縮レベルは低くなります（大型のデルタ ファイルは、コンテンツまたはレイアウトを共有しないパラメータ化 URL の要求で配信することもできます）。

次に、標準 URL の使用例を示します。カタログ Web サイトに、次のページが含まれているものとします。

http://server/catalog/business?category=pencils

http://server /catalog/business?category=erasers

http://server /catalog/consumer?category=pencils

http://server /catalog/consumer?category=erasers

この例で、business および consumer セクションの pencils ページには多数の共通コンテンツが含まれています。erasers ページの場合も同様です。これらのカテゴリでは、ベース ページを共有するほうが効率的です。つまり、次の 2 つのページのベース ページは共通になります。

http://server/catalog/business?category=pencils

http://server /catalog/consumer?category=pencils

次の **canonical-url** コマンドを使用すると、ACE は 4 つのすべての URL を一致させます。

```
host1/Admin(config-parammap-optmz)# canonical-url $(1)/
$http_query_param(category)
```

たとえば、要求が次の URL の場合、

http://server/catalog/business?category=pencils

(1) は http://server/catalog 文字列になり、http_query_param(category) により pencils 文字列に拡張されます。結果の標準 URL は、次のようになります。

http://server/catalog/pencils

次の URL の場合も同じです。

http://server/catalog/consumers?category=pencils

したがって、これらの 2 つの URL は同じベース ファイルを共有します。

同様に、eraser URL の標準 URL は、次のようになります。

http://server/catalog/erasers

これらも、同じベース ファイルを共有します。

Smart Rebasing

Smart Rebasing は、デルタ生成のために使用するベース ファイルをアップデートする機能です。サイトのベース コンテンツは一定期間中に変更されることが多いので、生成されるデルタのサイズはかなり大きくなります。デルタ最適化プロセスの効率を保持するために、必要に応じてベース ファイルを自動更新します。

Smart Rebasing により、ACE は必要に応じて URL を瞬時にリベースできます。以降に要求があった場合に備えて、古いベース ページのコピーが保持されます。

パラメータ マップ最適化モードの **rebase flashforward-percent** コマンドでは、応答の FlashForward URL のパーセンテージに基づいてリベースできる、しきい値を設定できます。パラメータ マップ最適化モードで **rebase delta-percent** コマンドを使用すると、デルタ応答サイズが、しきい値（ベース ファイル サイズのパーセンテージとして指定）を超えた時点で、リベースが実行されます。また、**rebase flashforward-percent** コマンドを使用すると、デルタ応答の FlashForward URL とベース ファイルのパーセンテージの差分がしきい値を超えた時点で、リベースが実行されます。

Smart Rebasing を使用すると、ACE は、取得したデルタ応答数、デルタ サイズが **rebase delta-percent** コマンドのしきい値を超えている応答数、FlashForward URL が多すぎる応答数を追跡します。一定のサンプル サイズ（10 など）に到達すると、ACE は、rebase delta percent および rebase FlashForward percent に設定されたしきい値を超えているデルタ応答のパーセンテージを調べます。デフォルトでは、全体のパーセンテージが 50% を超えると、リベースが自動的に実行されます。Smart Rebasing では、ACE が既存のベース ファイルでは大部分の要求に対して最小サイズのデルタ応答を戻すことができないと判別した時点で、ページが自動的にリベースされます。

Smart Rebasing では、リベースの実行時でもデルタ最適化が確実に実行されるので、ACE のパフォーマンスおよびコンテンツ アクセラレーションが全般的に改善されます。

MIME タイプの除外

コンテンツ プロバイダーによっては、Web クローラーを防ぐために、HTTP エンティティ ヘッダー フィールドに、text/HTML コンテンツではなく、nontext/HTML Multipurpose Internet Mail Extension（MIME）タイプ メッセージのラベルを付加することがあります。MIME タイプ除外機能により、MIME タイプを明示的に凝縮不可として設定できます。たとえば、JavaScript が MIME タイプ アプリケーション /x-text として報告されるように、Web サーバを設定したとします。この機能により、この MIME タイプを凝縮不可として指定すると、ACE はこの MIME タイプの応答を凝縮しません。この機能をディセーブルにすると、ACE はこの MIME タイプの応答を凝縮します。

mimetypes.conf ファイルにデルタ最適化を適用しない MIME タイプを指定するには、パラメータ マップ最適化モードで **delta exclude mime-type** コマンドを使用します。このファイルに何も指定しない場合、またはファイルが存在しない場合、すべての MIME タイプが凝縮可能および圧縮可能とみなされます。

MIME タイプ除外の設定の詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

クッキーの使用方法

ブラウザがクッキーおよび JavaScript をサポートしているかどうかを判別するために、ACE は、サイトへのユーザの初回アクセス時に、ユーザに戻されるページに JavaScript コードを挿入します。このコードがクライアントのブラウザによって実行されると、ランダムに生成される 128 ビット ユーザ ID のクッキーが作成されます。クライアントのブラウザが JavaScript をサポートしていない場合、スクリプトは無視されるので、ACE クッキーは作成されません。

クライアントが 2 回めの要求を発行すると、ACE は ACE クッキーを検索し、クライアントが JavaScript およびクッキーをサポートしているかどうかを確認します。ACE が ACE クッキーを検出すれば、クライアントは JavaScript をサポートしているとみなされます。ACE クッキーが検出されない場合、クライアントは JavaScript またはクッキーをサポートしていない（またはクライアントからの初回要求である）とみなされ、このユーザの要求にはデルタ最適化は適用されません。

ACE クッキーには、表 1-1 に示すアトリビュートがあります。

表 1-1　クッキーのプロパティ

アトリビュート	値
名前	ACCCDN
有効期限	30 日
ドメイン	ターゲット サイトと同じ
パス	“/”（完全なサイト）
値	ACE がユーザを個別に識別するための 128 ビットのランダム生成ユーザ ID

ACE は、ユーザが JavaScript を有効にしてアクセスしたあと、以降で JavaScript を無効にした場合の問題を ACE で処理できるように、クッキーの自動期限切れをサポートしています。このサポートがないと、ACE から（JavaScript でラップされている）凝縮されたコンテンツを取得した場合、ブラウザにブランク ページが表示されることになります。この状況を処理するために、ACE はクライアントへの応答に <NOSCRIPT> タグを挿入します。JavaScript が有効である初回の要求後にブラウザの JavaScript が無効にされた場合、クライアントがこの HTML コードを実行すると、クライアントの ACE クッキーが強制的に期限切れになり、（JavaScript が再び有効になるまで）以降のページは凝縮されずに取得されます。

JavaScript が無効である場合、クライアントは <NOSCRIPT> タグ内の HTML コードを実行し、http://www.example.com/?cisco=removeCookie などの URL を取得します。ACE はこの要求を検出すると、クライアントに対して HTTP 302 Temporary Redirect を発行し、要求されたオリジナル ページにクライアントをリダイレクトします。この応答には、クライアントの ACE クッキーをただちに期限切れにする Set-Cookie HTTP ヘッダーが含まれています。

Cisco AVS 3180A Management Station を使用した AppScope パフォーマンス モニタリング

オプションの Cisco AVS 3180A Management Station で実行する Management Console は、ACE の最適化機能のための AppScope レポート機能をはじめ、一連のデータベース機能、管理機能、およびレポート機能を備えています。

AppScope Performance Monitor は、企業がアプリケーション パフォーマンスを効率的に追跡できるブラウザ ベースのレポート機能を提供します。AppScope のレポーティング エンジンは、ドリルダウン レポートにより詳細なパフォーマンス モニタリングの結果をグラフィカルに表示します。

AppScope Performance Monitor のデータはすべて、完全独立のリレーショナル Postgres SQL データベースに保管されるので、企業は Crystal Reports などの独自のレポーティング ツールを使用したり、独自のカスタム パフォーマンス モニタリング レポートを作成できます。

AppScope パフォーマンス モニタリングの使用方法およびレポート生成の詳細については、付録 A「オプションの Cisco AVS 3180A Management Station によるレポート作成」を参照してください。

ACE/ クライアントの相互関係の詳細

ここでは、各クライアントから Web ページへの 2 回のアクセスにおける、ACE とクライアント ブラウザ間の相互関係について説明します。

（注）　この例では、all-user デルタ最適化方式を使用しています。

ここで説明する内容は、次のとおりです。

- 初回アクセス — 新規クライアント（p.1-37）
- 2 回めのアクセス — 既存クライアント（p.1-39）
- キャッシュ制御ヘッダー（p.1-43）

初回アクセス — 新規クライアント

このプロセスは、ACE の配置後、クライアントから Web ページへの初回アクセスによって開始されます。

このアクセスでは、ACE は起点サーバの透過プロキシとして動作し、単純にクライアントから要求された Web ページを戻します。クライアントが JavaScript をサポートしていることがわかっている場合には、ACE は戻されたページに一部の JavaScript コードを組み込みます（詳細は、「初回アクセス時の JavaScript の例」[p.1-38] を参照）。この JavaScript は、クライアントのシステム上で ACE クッキーの作成を試みます。JavaScript が正常に実行されると、ACE は、クライアントが実際に JavaScript をサポートし、JavaScript が有効に設定されていることを確認します。このクッキーの存在により、ACE は、このクライアントが同じページ（または同じクラスのページ）を以降で要求したときに、ACE から凝縮したコンテンツを戻せることを認識します。

初回アクセス時の JavaScript の例

次に、www.foo.com への初回アクセス時にクライアントに送信されるページの例を示します。ACE クッキーをインストールする JavaScript コードは、一番上に表示されています。ACE は、この JavaScript を既存の HTML ページ コンテンツに付加します。

```
<SCRIPT LANGUAGE="JavaScript">
//<!--
document.cookie="ACCCDN=5.0-f98f1ec8-7b57-402f-b23b-77f708a9a26b;
path=/;expires=Sat, 16 Jun 2007 18:50:05 GMT";
//-->
</SCRIPT>
<html><head><title>Foo!</title><base href=http://www.foo.com/>
<meta http-equiv="PICS-Label"
content='(PICS-1.1 "http://www.rsac.org/ratingsv01.html"
l gen true for "http://www.foo.com" r (n 0 s 0 v 0 l 0))'>
</head><body><center><form action=http://search.foo.com/bin/search>
<map name=m><area coords="11,0,73,52" href=r/a1>
<area coords="74,0,142,52" href=r/p1>
<area coords="143,0,212,52" href=r/m1>
<area coords="462,0,531,52" href=r/wn>
<area coords="532,0,600,52" href=r/i1>
<area coords="601,0,665,52" href=r/hw>
</map><img width=674 height=53 border=0 usemap="#m"
src=http://us.a1.yimg.com/us.yimg.com/i/ww/m5v4.gif alt=Foo><br>
<table border=0 cellspacing=0 cellpadding=3 width=640><tr><td
align=center width=205>
etc...
</body></html>
```

2 回めのアクセス ― 既存クライアント

このプロセスは、クライアントの 2 回めの Web ページ アクセスに続きます。この例は、クライアントの初回アクセス後、同じ Web ページへの 2 回め以降のすべてのアクセスを示しています。

このアクセスでは、ACE は初回アクセス時に作成された ACE クッキーの存在を確認して、クライアントが JavaScript をサポートしているかどうかを判別します。ACE は、クッキーを検出すると、JavaScript が有効で、クライアントがデルタ ページを処理できることを認識します。ACE は、デルタ ページを生成して、クライアントに配信します。デルタ ページのコンテンツの詳細については、「2 回めのアクセス時の JavaScript の例」（p.1-40）を参照してください。

（注）　ACE からの初回のデルタ ページの配信時には、ベース ページも配信する必要があるので、クライアントから ACE への 2 つの要求が必要になります。この特殊なベース ページには、以降のアクセスでのデルタ適用に使用する JavaScript が含まれています。このベース ページは、以降のデルタを計算できるように、ACE にも保管されます。以降のアクセス時には、リベースが必要にならないかぎり、小さなデルタ ページだけが配信されます。

以降の各アクセス時にも、ACE は必ず、ACE クッキーの存在を確認します。ACE クッキーが存在しない場合、ACE は初回アクセスと同じ処理を行い、新しいクッキーの作成を試みます。

デルタ ページが参照するベース ページをブラウザで使用できない場合、すなわち、ブラウザのキャッシュにも、Web キャッシュにも、ACE にも存在しない場合には、ACE から戻されたデルタ ファイルは無効になります。ベース ページが存在しない場合、クライアントは存在しないベース ページに対して ACE デルタの適用を試みるので、ブラウザは ACE から配信されたコンテンツにアクセスできません。この状況に対処するため、ACE は、ベース ページを使用できない場合、ブラウザにキャッシュを迂回させてベース ページを強制的にリロードさせる JavaScript コードを、クライアントへの応答に組み込みます。

2 回めのアクセス時の JavaScript の例

次に、www.foo.com への 2 回めのアクセス時にクライアントに送信されるページの例を示します。

```
<script type="text/javascript">var isACCBaseCorrect=false;</script>
<script type="text/javascript"
src="/4grv3hs41d2bkt4wj41kcotmlh/986848530/ausr/_acc_http_//www.foo.co
m/">
</script>
<noscript><META HTTP-EQUIV="Refresh" CONTENT="0;
URL=http://www.foo.com/?ciscoacc=removeCookie">
Please click on <a href="http://www.foo.com/?ciscoacc=removeCookie">
this url</a> if the page is not refreshed automatically in a few
seconds
</noscript>
<script type="text/javascript">
if (!isACCBaseCorrect)
document.location.reload(true);
</script>
<script type="text/javascript" >
/*
Portions Copyright (c) 2005-2007 by Cisco Systems, Inc. All rights
reserved.
SendDelta
*/
document.open('text/html', 'replace');
fgn_b(0, 861);
fgn_o('84022.657463.2834087');
fgn_b(881, 31);
fgn_o('528320');
fgn_o('art2');
.
.
.
fgn_o(' - Shop until midnight, Dec. 20');
fgn_b(7539, 11017);
fgn_flush();
document.close();

fgn_flush();
</script>
```

次に、ファイルの各セクションについて詳細に説明します。

```
<script type="text/javascript">var isACCBaseCorrect=false;</script>
```

このコード セグメントは、デフォルトの ACCBaseCorrect 変数を false に定義します。この変数は、凝縮されたページが正常に処理されると、true に設定されます。この変数は、デルタ ページ内で参照されるベース ファイルをブラウザで使用できない（ブラウザ キャッシュ、Web キャッシュ、および ACE ノード上に存在しない）という潜在的な問題が生じた場合、ACE がベース ファイルを回復して対処するときに使用されます。この機能がない場合、ブラウザは存在しないベース ページに対して ACE デルタの適用を試みるので、ACE から配信されたコンテンツにアクセスできません。この機能は、ベース ページを取得できない場合にブラウザに（キャッシュを迂回させて）ページを強制的にリロードさせ、ACE から新しいベース ページを取得させる JavaScript コードを ACE 応答内に組み込むことによって、この問題に対応しています。

```
<script type="text/javascript"
src="/4grv3hs41d2bkt4wj41kcotmlh/986848530/ausr/_fgn_http_//www.foo.co
m/">
</script>
```

このコード セグメントは、クライアントに、要求したコンテンツのベース ファイルを参照させます。ベース ファイルがブラウザのキャッシュに存在しない場合、ネットワーク キャッシュまたは ACE からベース ファイルを取得できます。このベース ファイルには、要求したオリジナル コンテンツと、クライアントがコンテンツ デルタから以降のページを構築するために使用する追加の関数定義が含まれています。ベース ファイル名は、要求したページの元のファイル名とは異なります。したがって、ベース ファイルを取得できるのは、ACE を使用しているクライアントだけです。

この例で、ベース ファイル名の規則を確認してみましょう。最初の文字列である 4grv3hs41d2bkt4wj41kcotmlh は、このベース ファイルを生成した ACE を個別に識別する ACE ID です。これは、ACE ノードの MAC アドレス、IP アドレス、およびポートに基づく一方向ハッシュです。2 番めの文字列である 986848530 は、ベース ファイルの更新タイム スタンプで、これにより ACE はベース ファイルのバージョン変更を検出できます。

```
<noscript><META HTTP-EQUIV="Refresh" CONTENT="0;
URL=http://www.foo.com/?ciscoacc=removeCookie">
Please click on <a href="http://www.foo.com/?ciscoacc=removeCookie">
this url</a> if the page is not refreshed automatically in a few
seconds
</noscript>
```

このコード セグメントは、クライアントの JavaScript が無効にされた場合、ACE でコンテンツを確実に配信できるよう、クッキーの自動期限切れ機能をイネーブルにします。これは、初回アクセス時に JavaScript が有効であったクライアントが、以降で JavaScript サポートを明示的に無効にした場合を想定しています。この機能がないと、ACE から凝縮されたコンテンツ（JavaScript）を取得した場合、ブラウザにブランク ページが表示されることになります。対処方法として、クライアントへの ACE 応答に、<NOSCRIPT> タグが組み込まれます。JavaScript が有効である初回の要求後にブラウザの JavaScript が無効にされた場合、クライアントがこの コードを実行すると、自動的にクライアントの ACE クッキーが強制的に期限切れになり、以降のページは凝縮されずに（ACE が生成した JavaScript コードなしで）取得されます。

```
<script type="text/javascript">
if (!isACCBaseCorrect)
document.location.reload(true);
</script>
```

このコードは、ベース ファイルを使用できない場合、ブラウザにキャッシュを迂回させて、凝縮されていないページをリロードさせます。このフェールオーバー メカニズムにより、クライアントは、ベース ファイルを使用できない場合でも、必ずコンテンツを取得することができます。

```
<script type="text/javascript">
/*
Portions Copyright (c) 2005-2007 by Cisco Systems, Inc. All rights
reserved.
SendDelta
*/
document.open('text/html', 'replace');
fgn_b(0, 861);
fgn_o('84022.657463.2834087');
fgn_b(881, 31);
fgn_o('528320');
fgn_o('art2');
.
.
.
fgn_o(' - Shop until midnight, Dec. 20');
fgn_b(7539, 11017);
fgn_flush();
document.close();

fgn_flush();
```

このコード セグメントは、コンテンツのデルタ情報を表しています。ベース ファイルに定義されている単純な文字列処理 JavaScript 関数により、クライアントは取得済みのベース ファイルから、新たに要求したページを構築できます。

キャッシュ制御ヘッダー

ACE からクライアントに送信されるデルタ ページは、起点サーバから提供されたオリジナル ページと同じ HTTP キャッシュ制御ヘッダーを使用しています。これにより、ネットワーク キャッシュで、オリジナル ページと同様にデルタ ページをキャッシュできます。

デフォルトでは、ACE からクライアントに送信されるベース ページには、ネットワーク エッジ キャッシュを効率的に使用するために、キャッシュ期間が 30 日に制限されたキャッシュ制御ヘッダーが使用されます。

FlashForward の動作

ここでは、ACE での FlashForward 機能の動作について説明します。

FlashForward の概要

FlashForward オブジェクト アクセラレーションは、オブジェクト検証要求に関連するアップストリームおよびダウンストリームの両方のトラフィックを削減することにより、組み込みイメージおよびオブジェクトの配信を加速します。

最終更新日および有効期限の日付が含まれている組み込みオブジェクトは、ブラウザ キャッシュ内で有効期限が切れていないオブジェクトであることを意味します。新しいブラウザ セッションで、このオブジェクトを参照する HTML ページにアクセスした場合、このオブジェクトに対する HTTP GET 要求は生成されません。FlashForward により、アップストリーム HTTP 要求トラフィックが削減され、ページの配信が加速されます。

FlashForward およびデルタ最適化のオプション

FlashForward およびデルタ最適化は、独立したメカニズムです。最適なアクセラレーションで帯域幅を節減するには、デルタ最適化と FlashForward を同時に使用する設定を推奨しますが、FlashForward だけ、またはデルタ最適化だけを設定することもできます。

最適な結果を得るには、「デルタ最適化を使用した場合の FlashForward の動作」（p.1-46）で説明しているように、デルタ最適化と FlashForward の両方を使用することを推奨します。

少数の HTML ページに多数のキャッシュ可能な組み込みオブジェクトが含まれているサイトの場合には、デルタ最適化を使用せずに FlashForward だけを設定できます（「デルタ最適化を使用しない場合の FlashForward の動作」[p.1-51] を参照）。このタイプの設定では、リピート アクセスによる HTML 配信のアクセラレーションはサポートされません。この設定でサポートされるのは、通常、複数のブラウザ セッションでのリピート アクセスによる組み込みオブジェクト配信のアクセラレーションです。このタイプの設定では、クライアント ブラウザが、クッキーまたは JavaScript ベースの Dynamic HTML（DHTML）をサポートしている必要はありません。FlashForward だけの場合、すべてのブラウザが自動的にサポートされますが、デルタ最適化を使用する場合には、ブラウザが DHTML をサポートしている必要があります。

多数の HTML ページがあり、キャッシュ可能な組み込みオブジェクト数が少ないサイトでは、FlashForward を使用せずにデルタ最適化を設定できます。このタイプの設定では、同じブラウザ セッション内または複数のブラウザ セッション間での HTML 配信のアクセラレーションがサポートされますが、複数のブラウザ セッション間での組み込みオブジェクト配信のアクセラレーションはサポートされません。デルタ最適化を使用する ACE 設定はすべて、クライアント ブラウザによるクッキーおよび JavaScript ベース DHTML のサポートが必要になります。

FlashForward およびリピート アクセス

ほとんどのユーザは、特定のページへの 2 回めのアクセス時に、FlashForward アクセラレーションの利点を得ることができます。2 回めのアクセスとは、実際には、ACE キャッシュが準備されたあとのクライアントの初回アクセスになります。

ここで説明する例は、ACE をインストールした直後を想定しているので、（デルタ最適化を使用するかどうかに関係なく）FlashForward オブジェクト用の ACE キャッシュの準備に必要な初期手順が含まれています。この例は URL への本当の初回アクセスを示しているので、クライアントが FlashForward の利点を得るのは、2 回め以降のリピート アクセス時です。

この 3 回のアクセス要件が適用されるのは、起点サーバからの*新規*（変更されていない）オブジェクト用の ACE キャッシュを実際に準備するクライアント（つまり、新規オブジェクトを要求した最初のクライアント）だけです。その他のク

ライアントは、初回のリピート アクセスの時点で、FlashForward の利点を自動的に得ることができます。他のクライアントは、初回のアクセス時に FlashForward オブジェクト、すなわちクッキーが付加されたページ（デルタ最適化が設定されている場合）またはプレーン HTML ページ（デルタ最適化が設定されていない場合）を取得できるという利点があります。これらのクライアントには、ACE キャッシュ内の FlashForward オブジェクトを参照する変換後の URL が適用されます。

デルタ最適化を使用した場合の FlashForward の動作

ここでは、デルタ最適化を使用した場合の、クライアント ブラウザのページへの 3 回のアクセスにおける FlashForward の動作について説明します。

初回アクセス：キャッシュの準備

ここでは、インストールされたばかりの ACE 上でキャッシュを準備するプロセスについて説明します（キャッシュは空です）。ACE キャッシュには、ベース ファイルまたはオブジェクトは、まだ含まれていません。ACE キャッシュを準備する場合、特定の URL への（クライアント単位の初回アクセスではなく）全体での初回の要求は、次のように処理されます。

1. クライアントが URL を要求します。ACE が、起点サーバへのアクセスをプロキシします。
2. 起点サーバから ACE に HTML ページが配信されます。
3. ACE は、ページを取得すると、HTML ページを解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトが現在のローカル キャッシュに存在するかどうかを確認します。この場合は初回のアクセスで、組み込みオブジェクト用の ACE キャッシュがまだ準備されていないので、HTML で参照されている組み込みオブジェクトは ACE にキャッシュされていません。ACE は、JavaScript により先頭に ACE クッキーを付加してページを圧縮し、そのほかは変更せずに配信します（FlashForward 機能は適用されず、標準のクッキー付加ページが作成されて、配信されます）。
4. これはユーザによる URL への初回アクセスなので、ACE は通常どおり、ベース ページを作成します。

5. クライアントは、圧縮されたクッキー付加ページを取得すると、HTML を解析して組み込みオブジェクトへの参照を調べ、HTTP GET 要求を発行して、起点サーバに直接、組み込みオブジェクトを要求します。
6. ACE はプロキシなので、HTTP GET は ACE 経由で起点サーバに渡され、起点サーバからの以降のオブジェクト応答（HTTP [200 OK]）も ACE 経由でクライアントに戻されます。ACE は、すべてのキャッシュ可能オブジェクトを、オブジェクトのパススルー時にキャッシュし、キャッシュを準備します。ACE は、複雑なキャッシュ実装機能を使用するのではなく、クライアントの HTTP GET 要求を使用して、キャッシュを実装します。このプロセスは、サーバからクライアントへの新規オブジェクトの初回配信時にのみ実行されます（ACE が、HTML 要求時に、オリジナルの HTML 参照オブジェクトがキャッシュに存在しないと判断した場合だけです）。これで、ACE キャッシュの準備は完了です。
7. クライアント ブラウザは、起点サーバから配信されたかのように、HTML 内で参照されているオリジナル オブジェクトを取得して、キャッシュします。

初回のリピート アクセス：FlashForward 変換

ここでは、初回のリピート アクセス時のプロセスについて説明します。

1. 新しいブラウザ セッションで、クライアントが同じ（または同様の）URL を要求します。
2. ACE はクッキーを調べ、クライアントが最適化された応答をサポートできることを認識します。
3. ACE が、起点サーバへの要求をプロキシします。
4. 起点サーバで HTML ページが作成され、ACE に配信されます。
5. ACE は、HTML ページを解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトがローカル キャッシュに存在するかどうかを確認します。存在する場合、ACE はキャッシュされているオブジェクトに FlashForward を適用できるかどうかを確認します（FlashForward オブジェクト タイプとして設定されているかどうかを判別します）。オブジェクトがキャッシュ可能で、FlashForward を適用できる場合、ACE は起点サーバに HTTP IMS 要求を発行して、キャッシュされているオブジェクトがまだ有効かどうかを判別します。

（注） ACE は、すべてのクライアント要求について If-Modified-Since（IMS）要求を発行するわけではありません。つまり、クライアントから要求を取得するごとに、起点サーバに対してオブジェクトのフレッシュネスを確認するわけではありません。ACE は、最適化パラメータ マップのキャッシュ フレッシュネス設定を使用して、IMS 要求を発行する頻度を判別します。たとえば、キャッシュの有効期限を 10 分間に設定した場合、ACE はそのオブジェクト タイプについて、10 分ごとにだけ IMS 要求を発行します。

   a. 起点サーバから HTTP 304 [Not Modified] ステータス コードの応答が戻されると、ACE はオブジェクト名（URL）にバージョンを付加してキャッシュ済みのオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。ACE は、304 応答情報を使用することにより、オブジェクトのフレッシュネスを検証し、クライアントからの IMS 要求の発行を不要にします。このプロセスにより、IMS/304 トラフィックに関連する WAN 往復時間が削減されるので、クライアントによるページのダウンロードが加速されます。

   b. 起点サーバから HTTP 200 [OK] ステータス コードの応答（および変更されたオブジェクト）が戻されると、ACE は変更されたオブジェクトをキャッシュし、オブジェクト名（URL）にバージョンを付加して新たにキャッシュしたオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。

6. 次に、ACE は、組み込みオブジェクトの参照（URL）が起点サーバ上のオリジナル オブジェクトではなく、新しい FlashForward 変換名（ACE キャッシュ内のオブジェクトを表すバージョンが付加された名前）をポイントするように、起点サーバから配信された HTML を書き換えます。

7. ACE は、書き換えた HTML とベース ページを比較して、デルタ ページを作成します（デルタ最適化は、書き換えられた HTML 上で実行されます）。デルタ最適化により、FlashForward プロセスの結果である HTML コンテンツ内の変更および URL 参照の変更が配信されます。これは、FlashForward とデルタ最適化の併用による重要な処理です。

8. ACE は、デルタ ページを圧縮して、クライアントに配信します。

9. クライアントは、デルタ ページを取得し、デルタ ページとベース ページから書き換えられた HTML を再構築します。
10. クライアント ブラウザは、HTML を解析して、組み込みオブジェクトへの参照を検索します。これらの参照は、この時点では、ACE にキャッシュされている FlashForward オブジェクトをポイントしています。
11. クライアントは ACE に FlashForward オブジェクトを要求し、ACE はクライアントに FlashForward オブジェクトを配信します。
12. クライアント ブラウザが、FlashForward オブジェクトを取得して、キャッシュします。

2 回めのリピート アクセス：利点の活用

ここでは、2 回めのリピート アクセス時のプロセスについて説明します。

1. 新しいブラウザ セッションで、クライアントが同じ（または同様の）URL を要求します。
2. ACE はクッキーを調べ、クライアントが凝縮された応答をサポートできることを認識します。
3. ACE が、起点サーバへの要求をプロキシします。
4. 起点サーバで HTML ページが作成され、ACE に配信されます。
5. ACE は、HTML を解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトがローカル キャッシュに存在するかどうかを確認します。存在する場合、ACE はキャッシュされているオブジェクトに FlashForward を適用できるかどうかを確認します（ACE に、そのオブジェクト タイプの FlashForward が設定されているかどうかを判別します）。オブジェクトがキャッシュ可能で、FlashForward を適用できる場合、ACE は起点サーバに HTTP IMS 要求を発行して、キャッシュされているオブジェクトがまだ有効かどうかを判別します。
   a. 起点サーバから HTTP 304 [Not Modified] ステータス コードの応答が戻されると、ACE は、キャッシュ内にあるリネーム後の現在の FlashForward が適用されたオブジェクトがまだ有効であると判断します。
   b. 起点サーバから HTTP 200 [OK] ステータス コードの応答（および変更されたオブジェクト）が戻されると、ACE は変更されたオブジェクトをキャッシュし、オブジェクト名（URL）にバージョンを付加して新たにキャッシュしたオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。

6. 次に、ACE は、組み込みオブジェクトの URL が FlashForward 変換名をポイントするように、起点サーバから配信された HTML を書き換えます。オブジェクトが変更されていなければ、オブジェクト名は初回のリピート アクセス時に参照されたオブジェクト名と同じになります（クライアントのブラウザにキャッシュされている名前と同じです）。オブジェクトが変更されている場合には、新しいバージョンが付加された名前になります（クライアント ブラウザのキャッシュに存在しないオブジェクトを示します）。
7. ACE は、書き換えた HTML とベース ページを比較して、デルタ ページを作成します（デルタ最適化は、書き換えられた HTML 上で実行されます）。デルタ最適化により、FlashForward プロセスの結果である HTML コンテンツ内の変更および URL 参照の変更が配信されます。
8. ACE は、デルタ ページを圧縮して、クライアントに配信します。
9. クライアントは、デルタ ページを取得し、デルタ ページとベース ページから書き換えられた HTML を再構築します。
10. クライアント ブラウザは、HTML を解析して、組み込みオブジェクトへの参照を検索します。これらの参照は、この時点では、ACE にキャッシュされている FlashForward オブジェクトをポイントしています。
11. ブラウザ キャッシュ内の組み込みオブジェクトには Expires ヘッダーが付加されているので、ブラウザは、HTML 内で参照され、ブラウザにキャッシュされていないオブジェクトに対してのみ、HTTP GET 要求を発行します（つまり、変更されたオブジェクトだけに HTTP GET 要求が発行されます）。すでにキャッシュされている FlashForward オブジェクトに対しては、HTTP GET 要求は発行されません。これらのオブジェクトは、長期有効の Expires が設定されているので、現在も有効であることがわかっているからです。FlashForward により、ACE は組み込みオブジェクトのフレッシュネスをダイナミックに判別し、クライアントにその情報を明示的に通知できるので、クライアントはオブジェクトのフレッシュネスを検証するために要求を発行して、貴重な時間と帯域幅を浪費する必要がなくなります。また、ACE により有効であると判断されたオブジェクトについては、IMS 要求がすべて除外されるので、ページのダウンロードが加速されます。

次のセクションでは、デルタ最適化を使用しない場合の FlashForward の動作について説明します。

デルタ最適化を使用しない場合の FlashForward の動作

ここでは、デルタ最適化を使用しない場合の、クライアント ブラウザのページへの 2 回のアクセスにおける FlashForward の動作について説明します。

初回アクセス：Application Appliance キャッシュの準備

ここでは、インストールされたばかりの ACE 上でのプロセスについて説明します（キャッシュは空です）。ACE キャッシュを準備する場合、特定の URL への（クライアント単位の初回アクセスではなく）全体での初回の要求は、次のように処理されます。

1. クライアントが URL を要求します。ACE が、起点サーバへのアクセスをプロキシします。
2. 起点サーバから ACE に HTML ページが配信されます。
3. ACE は、ページを取得すると、HTML ページを解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトが現在のローカル キャッシュに存在するかどうかを確認します。この場合は、組み込みオブジェクト用の ACE キャッシュがまだ準備されていないので、HTML で参照されている組み込みオブジェクトは ACE にキャッシュされていません。ACE は、通常どおりページを圧縮し、そのほかは変更せずに配信します（FlashForward 機能は適用されず、標準のクッキー付加ページが作成されて、配信されます）。デルタ最適化は設定されていないので、ページにはクッキー付加 JavaScript は追加されません。
4. クライアントは、圧縮された HTML ページを取得すると、HTML ページを解析して組み込みオブジェクトへの参照を調べ、HTTP GET 要求を発行して、起点サーバに直接、組み込みオブジェクトを要求します。
5. ACE はプロキシなので、HTTP GET は ACE 経由で起点サーバに渡され、起点サーバからの以降のオブジェクト応答（HTTP [200 OK]）も ACE 経由でクライアントに戻されます。ACE は、すべてのキャッシュ可能なオブジェクトを、パススルー時にキャッシュします。ACE は、複雑なキャッシュ実装機能を使用するのではなく、クライアントの HTTP GET 要求を使用して、キャッシュを実装します。このプロセスは、サーバからクライアントへの新規オブジェクトの初回配信時にのみ実行されます（HTML 要求時に、オリジナルの HTML 参照オブジェクトがキャッシュに存在しない場合だけです）。これで、ACE キャッシュの準備は完了です。
6. クライアント ブラウザは、起点サーバから配信されたかのように、HTML 内で参照されているオリジナル オブジェクトを取得して、キャッシュします。

初回のリピート アクセス：FlashForward 変換

ここでは、初回のリピート アクセス時のプロセスについて説明します。

1. 新しいブラウザ セッションで、クライアントが同じ（または同様の）URL を要求します。
2. ACE が、起点サーバへの要求をプロキシします。
3. 起点サーバで HTML ページが作成され、ACE に配信されます。
4. ACE は、HTML ページを解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトがローカル キャッシュに存在するかどうかを確認します。存在する場合、ACE はキャッシュされているオブジェクトに FlashForward を適用できるかどうかを確認します（FlashForward オブジェクト タイプとして設定されているかどうかを判別します）。オブジェクトがキャッシュ可能で、FlashForward を適用できる場合、ACE は起点サーバに HTTP IMS 要求を発行して、キャッシュされているオブジェクトがまだ有効かどうかを判別します。
   a. 起点サーバから HTTP 304 [Not Modified] ステータス コードの応答が戻されると、ACE はオブジェクト名（URL）にバージョンを付加してキャッシュ済みのオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。ACE は、304 応答情報を使用することにより、オブジェクトのフレッシュネスを検証し、クライアントからの IMS 要求の発行を不要にします。このプロセスにより、IMS/304 トラフィックに関連する WAN 往復時間が削減されるので、クライアントによるページのダウンロードが加速されます。
   b. 起点サーバから HTTP 200 [OK] ステータス コードの応答（および変更されたオブジェクト）が戻されると、ACE は変更されたオブジェクトをキャッシュし、オブジェクト名（URL）にバージョンを付加して新たにキャッシュしたオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。
5. 次に、ACE は、組み込みオブジェクトの参照（URL）が、起点サーバ上のオリジナル オブジェクトではなく、新しい FlashForward 変換名をポイントするように、起点サーバから配信された HTML を書き換えます。
6. ACE は、書き換えた HTML ページを圧縮して、クライアントに配信します。

7. クライアント ブラウザは、書き換えられた HTML を取得して解析し、組み込みオブジェクトへの参照を検索します。これらの参照は、この時点では、ACE にキャッシュされている FlashForward オブジェクトをポイントしています。
8. クライアントは ACE に FlashForward オブジェクトを要求し、ACE はクライアントに FlashForward オブジェクトを配信します。
9. クライアント ブラウザが、FlashForward オブジェクトを取得して、キャッシュします。

2 回めのリピート アクセス：利点の活用

ここでは、2 回めのリピート アクセス時のプロセスについて説明します。

1. 新しいブラウザ セッションで、クライアントが同じ（または同様の）URL を要求します。
2. ACE が、起点サーバへの要求をプロキシします。
3. 起点サーバで HTML ページが作成され、ACE に配信されます。
4. ACE は、HTML ページを解析して、組み込みオブジェクトへの参照を調べ、参照されているオブジェクトがローカル キャッシュに存在するかどうかを確認します。存在する場合、ACE はキャッシュされているオブジェクトに FlashForward を適用できるかどうかを確認します（FlashForward オブジェクト タイプとして設定されているかどうかを判別します）。オブジェクトがキャッシュ可能で、FlashForward を適用できる場合、ACE は起点サーバに HTTP IMS 要求を発行して、キャッシュされているオブジェクトがまだ有効かどうかを判別します。
   a. 起点サーバから HTTP 304 [Not Modified] ステータス コードの応答が戻されると、ACE は、キャッシュ内にあるリネーム後の現在の FlashForward が適用されたオブジェクトがまだ有効であると判断します。
   b. 起点サーバから HTTP 200 [OK] ステータス コードの応答（および変更されたオブジェクト）が戻されると、ACE は変更されたオブジェクトをキャッシュし、オブジェクト名（URL）にバージョンを付加して新たにキャッシュしたオブジェクトをリネームし、長期有効の [Expires] HTTP 応答ヘッダーを付加します（デフォルトの有効期限は 20 年以上です）。このオブジェクトは、これで FlashForward 変換されます。

5. 次に、ACE は、組み込みオブジェクトの URL が FlashForward 変換名をポイントするように、起点サーバから配信された HTML を書き換えます。オブジェクトが変更されていなければ、オブジェクト名は初回のリピート アクセス時に参照されたオブジェクト名と同じになります（クライアントのブラウザにキャッシュされている名前と同じです）。オブジェクトが変更されている場合には、新しいバージョンが付加された名前になります（クライアント ブラウザのキャッシュに存在しないオブジェクトを示します）。
6. ACE は、書き換えた HTML を圧縮して、クライアントに配信します。
7. クライアント ブラウザは、書き換えられた HTML を取得して解析し、組み込みオブジェクトへの参照を検索します。これらの参照は、この時点では、ACE にキャッシュされている FlashForward オブジェクトをポイントしています。
8. ブラウザ キャッシュ内の組み込みオブジェクトには Expires ヘッダーが付加されているので、ブラウザは、HTML 内で参照され、ブラウザにキャッシュされていないオブジェクトに対してのみ、HTTP GET 要求を発行します（つまり、変更されたオブジェクトだけに HTTP GET 要求が発行されます）。すでにキャッシュされている FlashForward オブジェクトに対しては、HTTP GET 要求は発行されません。これらのオブジェクトは、長期有効の Expires が設定されているので、現在も有効であることがわかっているからです。FlashForward により、ACE は組み込みオブジェクトのフレッシュネスをダイナミックに判別し、クライアントにその情報を明示的に通知できるので、クライアントはオブジェクトのフレッシュネスを検証するために要求を発行して、貴重な時間と帯域幅を浪費する必要がなくなります。また、ACE により有効であると判断されたオブジェクトについては、IMS 要求がすべて除外されるので、ページのダウンロードが加速されます。

JavaScript 内で参照され、HTML 内で参照されていない組み込みオブジェクト

組み込みオブジェクトへの参照を検索する場合、ACE は HTML 内の img、script、href、および background タグを解析します。ACE は、JavaScript 内の組み込みオブジェクトへの参照は検索しません。

ACE は、準備段階ですべてのキャッシュ可能な組み込みオブジェクトをキャッシュするので、キャッシュ可能な JavaScript 組み込みオブジェクトにも FlashForward が適用され、名前は変更されずに Expires ヘッダーが付加されます。Expires ヘッダーに関連する日付と時刻は、パラメータ マップ最適化モードの **cache ttl** コマンドにより、ACE キャッシュ フレッシュネス設定を使用して定義します。

クライアント ブラウザ内のオブジェクトの有効期限を制御するには、パラメータ マップ最適化モードで **expires-setting** コマンドを使用します。

詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

CDN URL を使用した場合の FlashForward の動作

ACE では、Content Delivery Network（CDN; コンテンツ配信ネットワーク）によって変換されたオブジェクトおよび URL に、FlashForward オブジェクト アクセラレーションを適用できます。CDN エッジ キャッシュが、初回の要求時に起点サーバからオブジェクトを識別して取得するには、CDN URL 内にオリジナル名が組み込まれている必要があります。

次の例は、Akamai によって変換された一般的な URL です。

```
http://a484.g.akamai.net/f/484/868/1h/www.hilton.com/en/hi/media/images/tabs/tab_0.gif
```

また、次の例は、Speedera によって変換された一般的な URL です。

```
http://gateway.speedera.net/www.gateway.com/images/cp/banners/homepage_bnr_broadband01.gif
```

どちらの場合も、CDN で変更された URL は、CDN ID 部分と、それに続くオリジナル URL で構成されています。次の例では、太字の部分が CDN ID です。

```
http://a484.g.akamai.net/f/484/868/1h/www.hilton.com/en/hi/media/images/tabs/tab_0.gif
```

```
http://gateway.speedera.net/www.gateway.com/images/cp/banners/homepage_bnr_broadband01.gif
```

CDN で変更された URL には起点サーバの URL が組み込まれているので、ACE は、FlashForward オブジェクト検証に必要なオリジナル URL を抽出できます。

FlashForward が適用されたコンテンツに、Akamai または Speedera が使用されているかどうかは、ユーザ側で識別できます。この機能により、ACE は CDN URL を識別して解析し、オリジナル URL を抽出して、FlashForward での必要性に応じて、起点サーバに組み込みオブジェクト検証要求を実行できます。FlashForward

変換は通常どおりに実行され、ACE により、CDN URL に独自の ID が付加されます。FlashForward 変換された CDN URL は、オブジェクトが変更されるごとに更新されます。ID はオブジェクトの MD5 ハッシュに基づいているので、オブジェクトが変更されると、ハッシュが変更され、URL も変更されます。

Akamai の場合、FlashForward 変換された CDN URL は、次のようになります。

```
http://a484.g.akamai.net/f/484/868/1h/www.hilton.com/en/hi/media/images/tabs/
tab_0_vtmmi14xg2fvmlkxsxuk0ty1xd_acc_V01.gif
```

この例では、tab_0.gif が、FlashForward 変換されたオブジェクト名に置換されています。

```
tab_0_vtmmi14xg2fvmlkxsxuk0ty1xd_ACC_V01.gif
```

同様に、Speedera の場合には、FlashForward 変換された URL は、次のようになります。

```
http://gateway.speedera.net/www.gateway.com/images
/cp/banners/homepage_bnr_broadband01_5f1fdnjgsaigblyav4f42wnb1g_ACC_V0
1.gif
```

この例では、homepage_bnr_broadband01.gif が、FlashForward 変換されたオブジェクト名に置換されています。

```
homepage_bnr_broadband01_5f1fdnjgsaigblyav4f42wnb1g_ACC_V01.gif
```

ACE では、Akamai および Speedera によって変換されたオブジェクトに FlashForward を適用し、これらの CDN で、FlashForward 変換されたオブジェクトを配信およびキャッシュできます。その結果、クライアントは CDN から FlashForward オブジェクトを取得できるので、以降のセッション要求時に CDN に対して IMS 要求を発行する必要はありません。

キャッシュ キーを変更するには、パラメータ マップ最適化モードで **cache key-modifier** コマンドを使用します。これは、使用される標準 URL 定義と同じです。**cache key-modifier** コマンドは、正規表現に基づいています。詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。

次に、Akami の設定例を示します。

```
host1/Admin(config)# class-map type http loadbalance match-any
AkamaiModifier_Classmap
host1/Admin(config-cmap-http-lb)# match http url
.*akamai\.net.*www(.*\.jpg)
host1/Admin(config-cmap-http-lb)# match http url
.*akamai\.net.*www(.*\.gif)
host1/Admin(config-cmap-http-lb)# match http url
.*akamai\.net.*www(.*\.jpeg)
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# no delta
host1/Admin(config-actlist-optm)# flashforward-object
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP1
host1/Admin(config-parammap-optmz)# cache key-modifier http://www$(1)
host1/Admin(config-parammap-optmz)# cache-policy request override-all
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY
host/Admin(config-pmap-optmz)# class AkamaiModifier_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST1 parameter
OPTIMIZE_PARAM_MAP1
```

AkamaiModifier_Classmap クラス マップは、URL が Akamai によって変換されているイメージと一致します（レイヤ 7 HTTP ロードバランス クラスマップ定義を参照）。この例のカッコ内には、一致したときに **cache key-modifier** コマンドで使用されるサブ表現が定義されています。“( )” 表現は、それぞれ番号（1 から開始されるインデックス）が付けられ、表現（*番号*）により任意の方法で使用できます。サブ表現の使用に関する詳細については、第 3 章「最適化 HTTP パラメータ マップの設定」を参照してください。表 3-3 に、使用できるパラメータ拡張関数が示されています。

この例では、f1==f2 であることを確証し、FlashForward が適正に動作するように、キャッシュ キーが変更され、URL の Akamai 部分が除去されています。ただし、CDN が長時間にわたって ACE からオブジェクトを取得しない場合には、FlashForward は無効になります。

次に、Speedera の同様の設定例を示します。

```
host1/Admin(config)# class-map type http loadbalance match-any
SpeederaModifier_Classmap
host1/Admin(config-cmap-http-lb)# match http url
.*speedera\.net.*www(.*\.gif)
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# action-list type optimization http ACT_LIST2
host1/Admin(config-actlist-optm)# no delta
host1/Admin(config-actlist-optm)# flashforward-object
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP2
host1/Admin(config-parammap-optmz)# cache key-modifier http://www$(1)
host1/Admin(config-parammap-optmz)# cache-policy request override-all
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY
host/Admin(config-pmap-optmz)# class SpeederaModifier_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST2 parameter
OPTIMIZE_PARAM_MAP2
```

この設定は、このクラスに一致する URL が異なることを除けば、Akamai の設定例と同様です。

次の作業

第 2 章「最適化 HTTP アクション リストの設定」に進み、最適化 HTTP アクション リストを設定します。アクション リストにより、特定タイプの動作に適用する一連のアプリケーション アクセラレーションおよび最適化の機能をグループ化します。