CHAPTER 2

最適化HTTP アクション リストの設定

この章では、Cisco 4700 Series Application Control Engine（ACE）アプライアンス用の最適化 HTTP アクション リストを設定する方法について説明します。アクション リストでは、特定の処理タイプに適用される、個々のアプリケーション アクセラレーションおよび最適化機能をひとまとめにします。

この章の内容は、次のとおりです。

最適化 HTTP アクション リストのコンフィギュレーション クイック スタート

表 2-1 に、最適化 HTTP アクション リストの設定に必要なステップの概要を示します。ステップごとに、CLI コマンドまたは作業に必要な手順の参照を示します。各機能および CLI コマンドに関連する全オプションの詳細については、表 2-1 の後ろの各セクションを参照してください。

表 2-1　最適化 HTTP アクション リストのコンフィギュレーション クイック スタート

作業およびコマンドの例

1. 複数のコンテキストで動作している場合は、CLI プロンプトで意図するコンテキストで動作しているかどうかを確認してください。必要に応じて、正しいコンテキストに変更してください。
`host1/Admin# changeto C1`
`host1/C1#`
この表では以後、特に指定しないかぎり、管理コンテキストを使用します。コンテキスト作成の詳細については、『*Cisco 4700 Series Application Control Engine Appliance Administration Guide*』を参照してください。

2. グローバル コンフィギュレーション モードを開始します。
`host1/Admin# config`
`host1/Admin(config)#`

3. 最適化 HTTP アクション リストを作成します。
`host1/Admin(config)# action-list type optimization http ACT_LIST1`
`host1/Admin(config-actlist-optm)#`

4. 必要に応じて、アクション リストでアプリケーション アクセラレーションおよび最適化機能を設定します。入力例を示します。
`host1/Admin(config-actlist-optm)# cache dynamic`
`host1/Admin(config-actlist-optm)# cache forward`
`host1/Admin(config-actlist-optm)# delta`
`host1/Admin(config-actlist-optm)# flashconnect-object`
`host1/Admin(config-actlist-optm)# exit`
`host1/Admin(config)#`

表 2-1　最適化 HTTP アクション リストのコンフィギュレーション クイック スタート（続き）

作業およびコマンドの例

5. レイヤ 7 HTTP 最適化ポリシー マップを作成して、既存のアクション リストとパラメータ マップを関連付け、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処理を設定します。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」を参照してください。
`host/Admin(config)# policy-map type optimization http first-match L7OPTIMIZATION_POLICY`
`host/Admin(config-pmap-optmz)# class L7SLBCLASS`
`host1/Admin(config-pmap-optmz-c)#`
`host1/Admin(config-pmap-optmz-c)# action ACT_LIST1 parameter OPTIMIZE_PARAM_MAP`
`host1/Admin(config-pmap-optmz-c)# exit`

6. （任意）フラッシュ メモリに設定変更を保存します。 `host1/Admin# copy running-config startup-config`

最適化 HTTP アクション リストの作成

アプリケーション アクセラレーションを実行するための最適化アクション マップを作成するには、グローバル コンフィギュレーション モードで **action-list type optimization http** 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。

action-list type optimization http *list_name*

list_name 引数には、最大 64 の英数字で、引用符なしのテキスト ストリングとして、一意の名前を入力します。

最適化 HTTP アクション リストを作成する場合の入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)#
```

コンフィギュレーションからアクション リストを削除する場合は、次のように入力します。

```
host1/Admin(config)# no action-list type optimization http ACT_LIST1
```

ACE のさまざまなアプリケーション アクセラレーションおよび最適化機能をイネーブルにするには、この章で紹介するコマンドを 1 つまたは複数使用します。最適化 HTTP アクション リストの完成後、最適化 HTTP パラメータ マップを作成して、特定の処理の実行方法について詳細を定義します。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。

AppScope パフォーマンス モニタリングのイネーブル化

オプションの Cisco AVS 3180A Management Station では、ACE の最適化機能に対応する Management Console が動作します。この Management Console には、データベース、管理、および AppScope レポートを含めたレポート機能が組み込まれています。オプションの Cisco AVS 3180A Management Station の使用方法および AppScope 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。

AppScope のパフォーマンス モニタリングを ACE で使用できるようにするには、アクション リスト最適化モードで **appscope** コマンドを使用します。

コマンドのシンタックスは、次のとおりです。

appscope

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# appscope
```

アクション リストで AppScope 機能をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no appscope
```

アプリケーション パフォーマンスを測定する AppScope パラメータおよび AppScope レポートのために要求を並べ替えるパラメータを制御できます。パラメータ マップ最適化モードで次のコマンドを設定します。

- **appscope optimize-rate-percent**
- **request-grouping-string**

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。

キャッシュ最適化のイネーブル化

対応する URL のキャッシュ最適化をイネーブルにするには、アクション リスト最適化モードで **cache** コマンドを使用します。キャッシュ機能の詳細については、第 1 章「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」を参照してください。

コマンドのシンタックスは、次のとおりです。

cache {**dynamic** | **forward** | **forward-with-wait**}

- **dynamic** — 応答の期限設定値によって、コンテンツがダイナミックであることが示されている場合を含めて、対応する URL のアダプティブ ダイナミック キャッシングをイネーブルにします。キャッシュ オブジェクトの期限は、時間またはサーバの負荷（パフォーマンス保証）に基づいた期限設定値で制御します。
- **forward** — 対応する URL のキャッシュ転送機能をイネーブルにします。**forward** キーワードを指定すると、ACE は最大キャッシュ TTL 期間がまだ経過していない場合に（パラメータ マップ最適化モードで **cache ttl** コマンドを使用して設定）、オブジェクトが期限切れであっても、自分のキャッシュ（スタティックまたはダイナミック）からオブジェクトを提供します。ACE は同時に、起点サーバに非同期要求を送信し、オブジェクトのキャッシュを更新させます。

（注） スタティック キャッシングを使用するのか、それともダイナミック キャッシングを使用するのかを特定するために、**cache forward** コマンドを指定する前に、**cache dynamic** コマンド（ダイナミック キャッシング）または **flashforward-object** コマンド（スタティック キャッシング）を指定する必要があります。

- **forward-with-wait** — 対応する URL の待機付きキャッシュ転送機能をイネーブルにします。オブジェクトが期限切れでも、最大キャッシュ TTL 期間がまだ経過していない場合（パラメータ マップ最適化モードで **cache ttl** コマンドを使用して設定）、ACE は起点サーバにオブジェクトを要求します。このページを要求する他のユーザには、この間も引き続き、キャッシュからコンテンツが与えられます。新しいオブジェクトが返されると、それが要求元ユーザに送信され、キャッシュも更新されます。

（注）　**forward-with-wait** キーワードの機能は、**forward** キーワードと同様ですが、要求を満たすには、オブジェクトが更新されるまで、個々のユーザが待機しなければならない点が異なります。**forward-with-wait** キーワードは、アプリケーションに処理のコンテキストが必要でありながら、非同期更新処理が適切ではないために、**forward** キーワードを指定できない場合に便利です。

対応する URL のキャッシュ転送機能をイネーブルにする場合の入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# cache dynamic
host1/Admin(config-actlist-optm)# cache forward
```

アクション リストでキャッシュ機能をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no cache forward
```

パラメータ マップ最適化モードで次のコマンドを設定することによって、ACE キャッシュ オブジェクト キー、キャッシュの鮮度、キャッシュ要求 / 応答ポリシーの設定値を定義します。

- **cache key-modifier**
- **cache parameter**
- **cache ttl**
- **cache-policy request**
- **cache-policy response**

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。

デルタ最適化のイネーブル化

対応する URL を凝縮するために、デルタ最適化をイネーブルにするには、アクション リスト最適化モードで **delta** コマンドを使用します。デルタ最適化の詳細については、第 1 章「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」を参照してください。

コマンドのシンタックスは、次のとおりです。

delta

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# delta
```

アクション リストでデルタ最適化をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no delta
```

パラメータ マップ最適化モードで次のコマンドを設定することによって、デルタ最適化モードおよびパラメータを定義します。

- **delta all-user**
- **delta cacheable-content**
- **delta exclude**
- **delta first-visit**
- **delta page-size**
- **delta per-user**

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 ジャストインタイム オブジェクト アクセラレーション

ジャストインタイム オブジェクト アクセラレーションを使用すると、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り、その結果、アプリケーションの応答時間が短縮されます。この機能によって、ユーザは要求ごとにオブジェクトをダウンロードする必要がなくなります。対応する URL のジャストインタイム オブジェクト アクセラレーションをイネーブルにするには、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。ジャストインタイム オブジェクト アクセラレーションの詳細については、第 1 章「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」を参照してください。

コマンドのシンタックスは、次のとおりです。

dynamic etag

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# dynamic etag
```

アクション リストでジャストインタイム オブジェクト アクセラレーションをディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no dynamic etag
```

高速リダイレクトのイネーブル化

高速リダイレクトは、起点サーバから 302 の応答を代行受信し、クライアントに代わってリダイレクト URL を求める二次要求を行い、リダイレクト URL を取得してクライアントに送信します。これが適用されるのは、同一ドメイン内のリダイレクトだけです。高速リダイレクトをイネーブルにするには、**fast-redirect** コマンドを入力します。

URL のスマート リダイレクトを設定するには、アクション リスト最適化モードで **meta refresh-to-302** コマンドを使用します（「URL のスマート リダイレクトのイネーブル化」を参照）。

このコマンドのシンタックスは、次のとおりです。

fast-redirect

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# fast-redirect
```

アクション リストで高速リダイレクトをディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no fast-redirect
```

FlashConnect のイネーブル化

FlashConnect を使用すると、ACE は帯域幅の使用率を引き下げ、HTML ページに組み込まれているオブジェクトのダウンロードを高速化します。FlashConnect はプレフィクスを追加し、ホスト名を変更することによって、組み込みオブジェクトの名前を動的に変更し、すべてのオブジェクトが単一ホスト上にある場合も含めて、各オブジェクトが異なるホストに配置されているように見せかけます。FlashConnect はブラウザがオブジェクトごとに、起点サーバに対して別々の接続を開始するようにします。その結果、オブジェクトが 1 つずつ取得されるのではなく、並行して取得されるので、ネットワーク パフォーマンスが向上します。FlashConnect の詳細については、第 1 章「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」を参照してください。

（注） この機能を使用するには、書き換えられたオブジェクト URL に対するすべての要求が解決されて（最初に書き換えた）ACE に戻るように、DNS を設定する必要があります。第 1 章「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」を参照してください。

対応する URL の FlashConnect をイネーブルにするには、アクション リスト最適化モードで **flashconnect** コマンドを使用します。

コマンドのシンタックスは、次のとおりです。

flashconnect

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# flashconnect
```

アクション リストで FlashConnect をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no flashconnect
```

対応する組み込みオブジェクトの URL に対して、FlashConnect パフォーマンス アクセラレーションをイネーブルにするには、アクション リスト最適化モードで **flashconnect-object** コマンドを使用します。FlashConnect オブジェクト ポリシーと一致する組み込みオブジェクトがトランスフォームの対象になります。

このコマンドのシンタックスは、次のとおりです。

flashconnect-object

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# flashconnect-object
```

アクション リストで FlashConnect パフォーマンス アクセラレーションをディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no flashconnect-object
```

次のコマンドを使用することによって、FlashConnect の動作パラメータを定義します。

- **prefix flashconnect** — FlashConnect 機能でホスト名を変形させるときに、組み込みオブジェクト URL のホスト名の前に追加する、グローバル プレフィクスを指定します。第 5 章「グローバル最適化値の設定」を参照してください。
- **flashconnect limit** — FlashConnect 機能に使用させる人工ホストの数を制限します。4 ホストがデフォルトの限度です。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 のイネーブル化

FlashForward オブジェクト アクセラレーションは、ACE の帯域幅使用率引き下げおよびダウンロード高速化の利点を HTML ページに組み込まれたオブジェクトに拡大します。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 リネームを結合して、親 HTML ページ内のオブジェクトの鮮度を維持します。FlashForward の詳細については、第 1 章「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」を参照してください。

対応する URL の FlashForward をイネーブルにして、組み込みオブジェクトのトランスフォームを行うには、アクション リスト最適化モードで **flashforward** コマンドを使用します。コマンドのシンタックスは、次のとおりです。

flashforward

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# flashforward
```

アクション リストで FlashForward をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no flashforward
```

対応する URL の FlashForward スタティック キャッシングをイネーブルにするには、アクション リスト最適化モードで **flashforward-object** 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。

flashforward-object

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# flashforward-object
```

アクション リストで FlashForward スタティック キャッシングをディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no flashforward-object
```

パラメータ マップ最適化モードで次のコマンドを使用することによって、ベースを変更する FlashForward リフレッシュ ポリシーおよびしきい値制御を定義します。

- **flashforward refresh-policy**
- **rebase flashforward-percent**

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。

画像最適化のイネーブル化

画像の最適化によって、画像品質の最適化と同時に、JPEG および PNG 画像ファイルの容量が縮小されます。その結果、画像のダウンロード時間が短縮され、ページのレンダリングが高速になり、帯域幅がいっそう効率よく利用されるようになります。画像最適化の詳細については、第 1 章「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」を参照してください。

ACE では、画像最適化はデフォルトでディセーブルです。画像最適化をイネーブルにするには、アクション リスト最適化モードで **image** 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。

image {**advanced** | **standard**}

キーワードは次のとおりです。

- **advanced** — 拡張画像最適化をイネーブルにします。拡張モードでは、ACE が標準の設定値を上書きし、個々の最適化オプションを制御します。
- **standard** — 標準画像最適化をイネーブルにします。標準モードでは、ACE が画像を最適化し、ノイズが少なくなるように、必要に応じてスムージングを行います。

画像品質が高くなるように（最適化をあまり行わないように）指定する場合の入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# image advanced
```

アクション リストで画像最適化をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no image advanced
```

次のパラメータ マップ最適化モード コマンドを定義することによって、JPEG および PNG 画像に適用する圧縮の度合いを制御できます。

- **image grayscale**
- **image high**
- **image ignore-thumbnails**
- **image progressive**
- **image smooth**

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。

URL のスマート リダイレクトのイネーブル化

スマート リダイレクトを使用すると、ACE は HTML メタタグ リダイレクトをより効率的な HTTP ヘッダーベース リダイレクトに自動的かつ透過的に変換できます。この機能により、不要な鮮度検証要求が排除され、その結果、ページの応答時間が大幅に短縮されます。その際、多くの企業アプリケーションで実現される HTML メタタグ リダイレクトの柔軟性が損なわれることはありません。

スマート リダイレクトの詳細については、第 1 章「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」を参照してください。

（注）　HTML メタタグ リダイレクトの目的が、別のページへのリダイレクトではなく、単にページをリロードするだけの場合は、スマート リダイレクトを使用しないでください。

URL のスマート リダイレクトをイネーブルにするには、アクション リスト最適化モードで **meta refresh-to-302** コマンドを使用します。

コマンドのシンタックスは、次のとおりです。

meta refresh-to-302

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# meta refresh-to-302
```

アクション リストで URL のスマート リダイレクトをディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no meta refresh-to-302
```

URL マッピングのイネーブル化

URL マッピングは、起点サーバとクライアント ブラウザ間で、データ ストリームの URL およびその他のコンテンツを変更する、ACE の能力に関連します。URL のフォーマットは、次のとおりです。

```
protocol:// host[: port]/ path
```

データ ストリームの URL またはその他のコンテンツをどこで変更するかを定義することによって、**urlmap scope** コマンドで、これらの URL の任意の部分を変更します。

ここで扱うトピックは、次のとおりです。

- データ ストリームの URL およびその他のコンテンツの変更
- パターン置換ディレクティブの使用方法

データ ストリームの URL およびその他のコンテンツの変更

ACE では、URL マッピングはデフォルトでディセーブルです。起点サーバとクライアント ブラウザ間で、データ ストリームの URL を変更できます。

アクション リスト最適化モードで **urlmap scope** コマンドを使用すると、置換ディレクティブの含まれたストリングを指定して、入力ストリームの任意の部分を変更できます。この変更が適用されるのは、パラメータ マップ最適化モードで **urlmap non-html** コマンドを指定した場合を除き、HTML ファイルだけです（第 3 章「最適化 HTTP パラメータ マップの設定」を参照）。

コマンドのシンタックスは、次のとおりです。

urlmap scope {**all** | **cookie** | **content** | **header** | **html**} {*replacementDirective*}

キーワードおよび引数は、次のとおりです。

- **all** — どこで検出されたかを問わず、URL を変更します（デフォルト）。
- **cookie** — cookie のドメイン セクションを変更します。
- **content** — パターンに基づいて（URL だけではなく）任意のコンテンツを変更します。

- **header** — Location 応答ヘッダー フィールド内の URL だけを変更します（ステータス コード 302 の応答の場合と同様）。
- **html** — META HTTP-EQUIV タグの URL アトリビュート内および次の HTML タグの SRC アトリビュート内の URL だけを変更します。BASE、HREF、IMG、LINK、SCRIPT、および STYLE です。
- *replacementDirective* — URL の変更方法。表 2-2 に、使用できる値を示します。

表 2-2　replacementDirective の値

replacementDirective	説明
host *src* **to** *dst*	ホスト部分がストリング *dst* になるように、ホスト部分に *src* が設定されている URL のホスト部分を変更します。ホスト ストリングの最大長は、英数字で 64 文字です。
pattern *regex_string* **to** *url_pattern_string*	入力ストリームの任意の部分を変更します。パターン *regex_string* は、変更する入力ストリーム内のサブ表現を定義する正規表現です。ストリング *url_pattern_string* で、置換パターンを指定します。ストリング *regex_string* および *url_pattern_string* の最大長は、英数字で 64 文字です。 詳細については、「パターン置換ディレクティブの使用方法」を参照してください。
port *src* **to** *dst*	ポートがストリング *dst* になるように、ポートに *src* が設定されている URL のポート部分を変更します。ポート番号の値は 0 ～ 65535 です。
protocol {**http** | **https**} **to** {**http** | **https**}	URL のプロトコルを HTTP から HTTPS に、または HTTPS から HTTP に変更します。

HTML パターンを変更する入力例を示します。

```
host1/Admin(config-actlist-optm)# urlmap scope html pattern
(.*)(https:)(//.*) to  $urlmap_pattern(1)http:$urlmap_pattern(3)
```

変更した HTML パターンをアクション リストから削除するには、次のように入力します。

```
host1/Admin((config-actlist-optm)# no urlmap scope html pattern
(.*)(https:)(//.*) to $urlmap_pattern(1)http:$urlmap_pattern(3)
```

パターン置換ディレクティブの使用方法

パターン置換ディレクティブのシンタックスは、次のとおりです。

pattern *regex_string* **to** *url_pattern_string*

引数 *regex_string* では、正規表現のシンタックスを使用して、変更する入力ストリーム内のサブ表現を定義します。サブ表現は、カッコ () を使用して区切ります。サブ表現の番号は 1 から始まり、これが左カッコ「(」の番号で、左から数えます。

入力ストリーム内の 3 つのサブ表現を定義するパターン例を示します。

```
(.*)(fast)(.*)
```

最初のサブ表現は、「fast」という単語の前にあるすべてです。2 番めのサブ表現は、「fast」という単語です。3 番めのサブ表現は、「fast」という単語の後ろにあるすべてです。

ストリング *url_pattern_string* では、置換パターンを指定します。*src* で定義されたストリング全体が *dst* で定義されたストリングによって置き換えられます。ストリング *dst* では、表 2-3 に記載されている任意のパラメータ拡張関数、1 つまたは複数の urlmap_pattern（*number*）変数を使用できます。この変数は、元の入力ストリームに戻って特定のサブ表現を参照します。

$urlmap_pattern(0) は、入力ストリーム全体と一致します。$urlmap_pattern(1) は、最初のサブ表現と一致します。$urlmap_pattern(2) は 2 番めのサブ表現と一致します（以下同様）。

入力ストリームに指定されたサブ表現がない場合、変数は空ストリングとして評価されます。

（注） 表 2-3 に記載されたパラメータ拡張関数が適用されるのは、クライアントからの HTTP 要求のコンテキストに限られます。クライアントへの応答として送信されるデータ ストリームには適用されません。たとえば、関数 $http_query_string() は、要求 URL のクエリー ストリングとして評価されます。

表 2-3 に、使用できるパラメータ拡張関数を示します。

表 2-3　パラメータ拡張関数

変数	説明
`$(number)`	URL パターンの（*number* で）一致する対応サブ表現に拡張します。URL パターン内のサブ表現は、カッコ () を使用して表します。サブ表現の番号は 1 から始まり、これが左カッコ「(」の番号で、左から数えます。番号には任意の正の整数を指定できます。$(0) は URL 全体と一致します。たとえば、URL パターンが ((http://server/.*)/(.*)/)a.jsp の場合、一致した URL は次のとおりです。 http://server/main/sub/a.jsp?category=shoes&session=99999　です。したがって、下記は有効です。 $(0) = http://server/main/sub/a.jsp $(1) = http://server/main/sub/ $(2) = http://server/main $(3) = sub 入力ストリームに指定されたサブ表現がない場合、変数は空ストリングとして展開されます。
`$http_query_string()`	URL のクエリー ストリング全体の値に展開されます。たとえば、次の URL の場合、 http://myhost/dothis?param1=value1&param2=value2 下記は有効です。 $http_query_string() = param1=value1&param2=value2 この関数は、GET 要求と POST 要求の両方に適用されます。

表 2-3　パラメータ拡張関数（続き）

変数	説明

`$http_query_param(query-param-name)`
 	
この廃止されたシンタックスもサポートされます。
`$param(query-param-name)`
	名前で指定されたクエリー パラメータ（大文字と小文字が区別される）の値に展開されます。たとえば、次の URL の場合、
	http://server/main/sub/a.jsp?category=shoes&session=99999
	下記は有効です。
	$http_query_param(category) = shoes
	$http_query_param(session) = 99999
	クエリーに指定されたパラメータがない場合、変数は空ストリングとして展開されます。この関数は、GET 要求と POST 要求の両方に適用されます。

`$http_cookie(cookie-name)`	指定された cookie の値に展開されます。$http_cookie(cookiexyz) が 1 例です。cookie 名は大文字と小文字が区別されます。
`$http_header(request-header-name)`	指定された HTTP 要求ヘッダーの値に展開されます。複数の値を持つヘッダーの場合、HTTP 仕様で指定された単一表現になります。$http_header(user-agent) が 1 例です。HTTP ヘッダー名は、大文字と小文字が区別されません。
`$http_method()`	GET、POST など、要求に使用した HTTP メソッドとして評価されます。

ブール関数
 	
`$http_query_param_present (query-param-name)`
`$http_query_param_notpresent (query-param-name)`
`$http_cookie_present(cookie-name)`
`$http_cookie_notpresent(cookie-name)`
`$http_header_present(request-header-name)`
`$http_header_notpresent(request-header-name)`
`$http_method_present(method-name)`
`$http_method_notpresent(method-name)`
	ブール値として評価されます。つまり、要求における要素の有無に応じて、真または偽です。要素は特定のクエリー パラメータ（query-param-name）、特定の cookie（cookie-name）、特定の要求ヘッダー（request-header-name）、または特定の HTTP メソッド（method-name）です。識別情報は、HTTP 要求ヘッダー名を除き、すべて大文字と小文字の区別があります。

次の **urlmap scope** コマンドを例にします。

```
host1/Admin(config-actlist-optm)# urlmap scope content pattern
(.*)(fast)(.*) to $urlmap_pattern(1)faster$urlmap_pattern(3)
```

入力ストリームは次のとおりです。

```
Cisco makes accessing the web fast now.
```

結果として、次のストリームになります。

```
Cisco makes accessing the web faster now.
```

コンテンツの logos/dna.gif ストリングを http://www.google.com/logos/dna.gif に変更する例を示します。

```
host1/Admin(config)# class-map type http loadbalance match-any
Example1_Classmap
host1/Admin(config-cmap-http-lb)# match http url
.*google.com/index.html
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# flashforward
host1/Admin(config-actlist-optm)# urlmap scope content pattern
(.*)(logos/dna.gif)(.*) to
$urlmap_pattern(1)http://www.google.com/logos/dna.gif$urlmap_pattern(3)
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP1
host1/Admin(config-parammap-optmz)# flashforward refresh-policy direct
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY1
host/Admin(config-pmap-optmz)# class Example1_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST1 parameter
OPTIMIZE_PARAM_MAP1
```

入力ストリームのすべての URL について、プロトコルを HTTPS から HTTP に変更する例を示します。

```
host1/Admin(config)# class-map type http loadbalance match-any
Example2_Classmap
host1/Admin(config-cmap-http-lb)# match http url .*t1a\.asp
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)#  action-list type optimization http ACT_LIST2
host1/Admin(config-actlist-optm)# urlmap scope all protocol https to
http
host1/Admin(config-actlist-optm)# delta
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP2
host1/Admin(config-parammap-optmz)# delta all-user
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY2
host/Admin(config-pmap-optmz)# class Example2_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST2 parameter
OPTIMIZE_PARAM_MAP2
```

入力ストリームのすべての URL について、ポートを 911 から 80 に変更する例を示します。

```
host1/Admin(config)# class-map type http loadbalance match-any
Example3_Classmap
host1/Admin(config-cmap-http-lb)# match http url .*t1b\.asp
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# action-list type optimization http ACT_LIST3
host1/Admin(config-actlist-optm)# urlmap scope all port 911 to 80
host1/Admin(config-actlist-optm)# delta
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP3
host1/Admin(config-parammap-optmz)# delta all-user
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY3
host/Admin(config-pmap-optmz)# class Example3_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST3 parameter
OPTIMIZE_PARAM_MAP3
```

この機能を使用して、コンテンツに含まれる任意の HTML タグを変更する例を示します。この例では、コンテンツの各 H1 タグを B タグに変更します。

```
host1/Admin(config)# class-map type http loadbalance match-any
Example4_Classmap
host1/Admin(config-cmap-http-lb)# match http url .*t4\.asp
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# action-list type optimization http ACT_LIST4
host1/Admin(config-actlist-optm)# urlmap scope content pattern
"(^.*)(\<H1\>)(.*$)" to $urlmap_pattern(1)B$urlmap_pattern(3)
host1/Admin(config-actlist-optm)# delta
host1/Admin(config-actlist-optm)# exit
host1/Admin(config)# parameter-map type optimization http
OPTIMIZE_PARAM_MAP4
host1/Admin(config-parammap-optmz)# delta all-user
host1/Admin(config-parammap-optmz)# exit
host/Admin(config)# policy-map type optimization http first-match
L7OPTIMIZATION_POLICY4
host/Admin(config-pmap-optmz)# class Example4_Classmap
host1/Admin(config-pmap-optmz-c)# action ACT_LIST4 parameter
OPTIMIZE_PARAM_MAP4
```

XSLT マージのイネーブル化

ACE は、XML ソース ドキュメントに XSL スタイルシート トランスフォーメーションを適用し、生成された HTML ドキュメントを要求元に戻すことができます。ACE は XML のトランスフォーメーション後、結果を要求元に戻すまでの間に、その他の最適化を適用します。

XML ソース ドキュメント、対応する XSL スタイルシート、さらに XSL スタイルシートを適用した XML ドキュメントのトランスフォーメーション後に ACE によって戻される HTML ドキュメントの例を示します。

XML ドキュメント例の foo.xml は、次のとおりです。

```
<?xml version="1.0"?>
<doc>Hello</doc>
```

XML ソース ファイルに XSLT マージ機能を使用します。その結果、ACE は XSLT スタイルシートを適用することによって、XML ソースのトランスフォーメーションを実行し、生成されたドキュメントを要求元に戻します。XSLT スタイルシートは、XML ドキュメントで指定するか、または XsltUseStylesheet ディレクティブを使用して指定します。

XSLT マージ機能をイネーブルにするには、アクション リスト最適化モードで **xslt merge** コマンドを使用します。

コマンドのシンタックスは、次のとおりです。

xslt merge

入力例を示します。

```
host1/Admin(config)# action-list type optimization http ACT_LIST1
host1/Admin(config-actlist-optm)# xslt merge
```

アクション リストで XSLT マージ機能をディセーブルにするには、次のように入力します。

```
host1/Admin(config-actlist-optm)# no xslt merge
```

XSLT マージ デバッグ機能を指定できます。その場合、XSLT スタイルシートの完全修飾パスを指定して事前トランスフォーメーションを実行するか、またはパラメータ マップ最適化モードで次のコマンドを設定することによって、XSLT スタイルシートの完全修飾パスを指定し、XSLT マージを強制的に実行します。

- **xslt merge-debug**
- **xslt pretransformer**
- **xslt stylesheet**

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。

次の作業

第 3 章「最適化 HTTP パラメータ マップの設定」に進み、ACE 上で最適化 HTTP パラメータ マップを設定します。このパラメータ マップを使用すると、アクション リストで指定されたアプリケーション アクセラレーションおよび最適化機能を調整したり制御したりできます。

次の作業