



Flash Media Server

エッジサーバーユーザーガイド

商標

Afterburner、AppletAce、Attain、Attain Enterprise Learning System、Attain Essentials、Attain Objects for Dreamweaver、Authorware、Authorware Attain、Authorware Interactive Studio、Authorware Star、Authorware Synergy、Backstage、Backstage Designer、Backstage Desktop Studio、Backstage Enterprise Studio、Backstage Internet Studio、Contribute、Design in Motion、Director、Director Multimedia Studio、Doc Around the Clock、Dreamweaver、Dreamweaver Attain、Drumbeat、Drumbeat 2000、Extreme 3D、Fireworks、Flash、Fontographer、FreeHand、FreeHand Graphics Studio、Generator、Generator Developer's Studio、Generator Dynamic Graphics Server、Knowledge Objects、Knowledge Stream、Knowledge Track、Lingo、Live Effects、Macromedia、Macromedia M Logo & Design、Macromedia Contribute、Macromedia Flash、Macromedia Xres、Macromind、Macromind Action、MAGIC、Mediamaker、Object Authoring、Power Applets、Priority Access、Roundtrip HTML、Scriptlets、SoundEdit、ShockRave、Shockmachine、Shockwave、Shockwave Remote、Shockwave Internet Studio、Showcase、Tools to Power Your Ideas、Universal Media、Virtuoso、Web Design 101、Whirlwind、および Xtra は、Macromedia, Inc. の商標であり、米国およびその他の国の法域で登録される場合があります。本マニュアルにおけるその他の製品名、ロゴ、デザイン、タイトル、語句は、Macromedia, Inc. またはその他の団体の商標、サービスマーク、または商号である場合があります、米国およびその他の国の特定の法域で登録されている場合があります。

サードパーティー情報

Jabber は、Jabber Software Foundation の登録商標です。



Sorenson™ Spark™ ビデオ圧縮および圧縮解除テクノロジーは、Sorenson Media, Inc. のライセンス供与によって提供されます。

本マニュアルには、Macromedia 社が管理していない、サードパーティーの Web サイトへのリンクが掲載されていますが、Macromedia 社はいかなるリンク先サイトの内容についても責任を持ちません。本マニュアルに記載されているサードパーティーの Web サイトには、自己責任においてアクセスしてください。Macromedia 社はこれらのリンクを便宜上の目的においてのみ掲載しています。リンクを掲載することにより、Macromedia 社がこれらのサードパーティーのサイトの内容について何らかの責任を持つことを示すものではありません。

Copyright © 2002-2005 Macromedia, Inc. All rights reserved. 本マニュアルの一部または全体を Macromedia, Inc. の書面による許諾を受けることなく、コピー、複写、複製、翻訳、電子的または物理的に変換することは禁じられています。

マニュアル制作スタッフ

プロジェクト管理 : Suzanne Smith

執筆 : John Norton、Suzanne Smith

編集 : Evelyn Eldridge、Mary Ferguson、Lisa Stanziano、Anne Szabla

制作管理 : Adam Barnett

メディアデザイン・制作 : Aaron Begley、Paul Benkman、John Francis、Mario Reynoso

初版 : 2005 年 10 月

Macromedia, Inc.

601 Townsend St.

San Francisco, CA 94103

マクロメディア株式会社

東京都港区赤坂 2-17-22 赤坂ツインタワー本館 13F

目次

第1章：エッジサーバーとオリジンサーバーの使用.....	5
エッジサーバーとは.....	5
エッジサーバーの動作.....	6
エッジサーバーでのデータのキャッシング.....	9
DMZ へのエッジサーバーのデプロイメント.....	9
明示プロキシと暗黙プロキシ.....	10
リバースプロキシ.....	11
ルーティング情報.....	12
エッジサーバーへの接続.....	12
プロキシサーバーの存在の検出.....	13
エッジサーバーのチェーン化.....	13
エッジサーバーの設定.....	14
エッジサーバーのクラスタのデプロイメント.....	15
エッジサーバーのクラスタへの接続.....	15
クラスタへのプロキシサーバーの登録.....	16
エッジクラスタ経由のアプリケーションへのアクセス.....	16
リバースプロキシのクラスタリング.....	17
オリジンサーバーとプロキシサーバーのセットアップ.....	18
エッジサーバークラスタの保守.....	20
エッジサーバーのキャッシュのクリア.....	20

エッジサーバーと オリジンサーバーの使用

この章では、Flash Media Server と共にエッジサーバーとオリジンサーバーをデプロイするためのさまざまな方法を説明します。

エッジサーバーとは

Flash Media Server の以前のバージョンでは、クライアントは常に、アプリケーションが稼働されているコンピュータに直接接続していました。アプリケーションの実行は、クライアントが接続しているコンピュータ上で行われており、アプリケーションは、ローカルで移動していました。Flash Media Server の今リリースでは、アプリケーションのリモート実行という概念を導入しています。Flash Media Server では、オリジンサーバーとしてローカルで、あるいはエッジサーバーとしてリモートで、アプリケーションを実行できるようになりました。エッジサーバーは、Flash Media Server の異なる種類ではなく、アプリケーションをリモートで実行できるよう構成設定したものです。

エッジサーバーのデプロイメントにはさまざまな利点がありますが、セキュリティの強化もその1つです。オリジンサーバー上で稼働するアプリケーションは、インターネットにダイレクトに公開されなくなります。Flash Media Server のサービスに対するすべての要求は、既知のセキュアな接続ポイントにルーティングされます。これらの接続ポイントは、エッジサーバーまたはプロキシサーバーと呼ばれ、これら2つの用語はどちらも同じ意味で使われます。管理者は、エッジサーバー上のトラフィックを監視することができます。各エッジサーバーのアクセスログファイルによって、管理者は接続が許可されたものであるかを検証できます。ログファイルによって、許可されなかった接続要求を特定することもできます。

エッジ/オリジンサーバーは柔軟にデプロイすることができます。エッジサーバーを利用することで、現行のデプロイメントを解体することなく、Flash Media Server の処理量をスケールアップしたり、サーバーへのアクセス負荷を再配分したりできます。エッジサーバーの追加も可能であるため、組織内のユーザーや拠点に合わせてエッジサーバーの場所を変更することや、拠点間でのトラフィックのフローに変更を加えることができます。

エッジサーバーのデプロイメントによって、管理者は、Flash Media Server のサービスに対する接続要求を分散させることができます。クライアントとオリジンサーバー間のトラフィックでは、既存の帯域幅とシステムリソースがより効率的に使用されます。クライアントのコンピュータとエッジサーバーの間では、ローカルのトラフィックの割合が増えます。エッジサーバーが存在しない環境では、Flash Media Server オリジンサーバーがどこに配置されている場合でも、すべてのクライアントはそのオリジンサーバーに接続する必要があります。

エッジサーバーは、組織の信頼できるネットワークとインターネットのような信頼できないネットワーク間の通信トラフィックを仲介します。エッジサーバーを使用しても、IP レベルでトラフィックを管理するファイアウォールの必要性がなくなるわけではありませんが、アプリケーションレベルのセキュリティが一層強化されます。

エッジサーバーの動作

エッジ / オリジンのサーバー構成においては、ユーザーのコンピュータから Flash Media Server のサービスに対するすべての接続要求がエッジサーバーにリダイレクトされます。エッジサーバーは、ユーザーのコンピュータからの要求を受け取りますが、アプリケーションを実行しているわけではありません。エッジサーバーは、アプリケーションが稼働している、オリジンサーバーと呼ばれる別のコンピュータへの接続を確立します。オリジンサーバーは、Flash Media Server のサービスに対する要求を処理し、アプリケーションのデータをエッジサーバーに戻します。そして、そのデータがクライアント（ユーザーのコンピュータ）に転送されます。

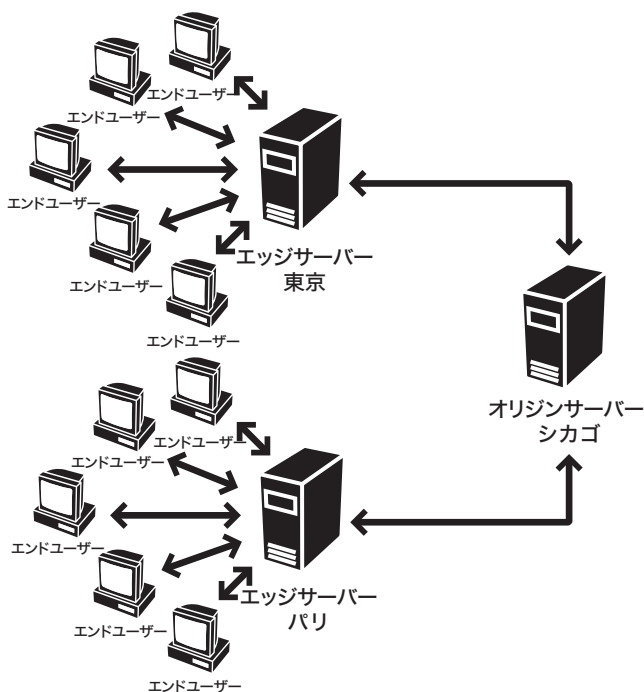
ユーザーには、アプリケーションが置かれているオリジンサーバーに自分自身が直接接続しているように見えます。エッジサーバーは、自分自身がアプリケーションを実行しているかのように動作しますが、エッジサーバーの役割は、クライアントとオリジンサーバー間の通信を仲介することです。エッジサーバーはユーザーを認証し、認証されたユーザーからの Web サービス要求を許可します。

たとえば、あるコンピュータに Flash Media Server が単体でデプロイされていて、アプリケーションインスタンスを1つだけ実行している場合には、多数のユーザーが複数の場所からほぼ同時に Flash Media Server に接続しようとする、システムリソースとネットワークリソースが足りなくなる可能性があります。会社全体の打ち合わせや会議では、このような状況が発生する可能性があります。同じアプリケーションに同時にアクセスしようとするユーザーの数が多いと、既存の帯域幅とシステムリソースを圧迫するおそれがあります。リソースの再設定あるいは再割り当てが必要であることを示すもう1つの指標は、待ち時間が長くなることです。

オリジンサーバーへの接続が発生するたびに、その接続を介した実際のデータフローとは別にリソースが消費されます。接続の数が増えれば、このような負荷が過度に大きくなり、結果としてサーバーのパフォーマンスを低下させることとなります。エッジサーバーは、接続数を集約することで、この問題を大幅に緩和します。多数のクライアントからの接続を多重化し、オリジンサーバーへの1つの接続に集約します。

単一サーバーへの負荷の増大に対応するために、Flash Media Server とネットワークの管理者は、異なるデプロイメントを検討する必要があります。その場合、いくつかの仮想ホストをエッジサーバーとして稼働させ、それ以外の仮想ホストはオリジンサーバーとして稼働させるように Flash Media Server を設定することで、システムと帯域幅のリソースの負荷を適正に配分するという方法をとることができます。

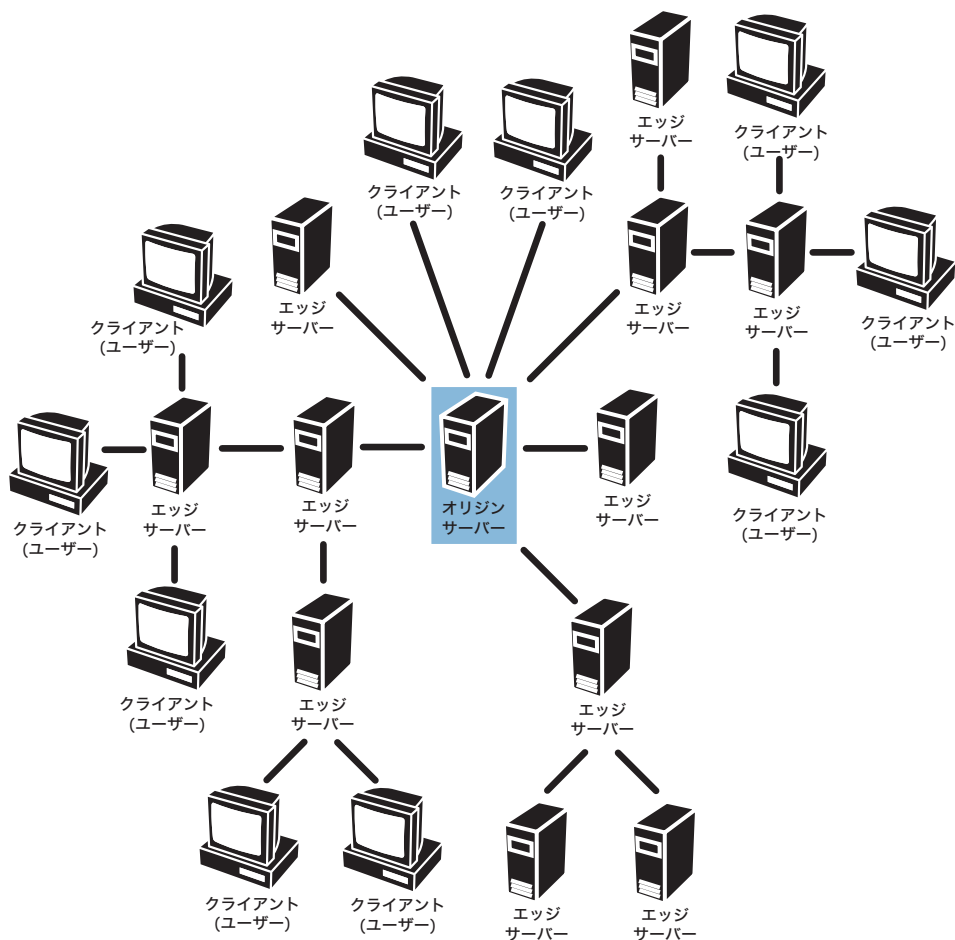
それぞれの要求をオリジンサーバーに転送する処理ではリソースが消費されるため、エッジサーバーでは、多数のクライアントからの要求をオリジンサーバーへの1つの接続に集約します。エッジサーバーとオリジンサーバーとの間の通信は、ユーザーには透過的に発生します。



ネットワークリソースおよびシステムリソースの需要を分散させるには、管理者が特定のエッジサーバーをいずれかの地域や組織的な機能の領域に割り当てます。たとえば、あるエッジサーバーは東京のユーザーからの要求を集約して転送し、別のエッジサーバーはパリからの要求を集約して転送するように設定します。パリと東京のエッジサーバーはそれぞれのクライアントからの要求を集約し、たとえばシカゴのような別のセキュアな場所に置かれているオリジンサーバーにそれを転送します。

パリと東京にいるユーザーは、それぞれの地域に割り当てられたエッジサーバーを必ず経由して、オリジンサーバーにアクセスします。これらのエッジサーバーはオリジンサーバーからの応答を受け取って、それぞれの地域、すなわち東京またはパリのクライアント（ユーザーのコンピュータ）にそれを配信します。さらに、エッジサーバーは、オリジンサーバーから受け取ったデータをキャッシュに保存して、そのエッジサーバーに接続している他のクライアントもそのデータを使用できるようにします。このようなデータの再利用も、エッジサーバーによる効率的なリソース使用のもう1つの方法です。静的なコンテンツをキャッシュすることで、オリジンサーバーにおける全体的な負荷が緩和されます。

ネットワーク化された環境への Flash Media Server のデプロイメントでは、複数のエッジサーバーを設定することになりますが、それらのエッジサーバーは、個別にデプロイすることも、クラスタ化することもできます。また、エッジサーバーをチェーン構成にし、あるエッジサーバーが他のエッジサーバーとそれらのクライアントからの接続要求を集約してからオリジンサーバーにその要求を送るという方式にすることもできます。



エッジサーバーでのデータのキャッシング

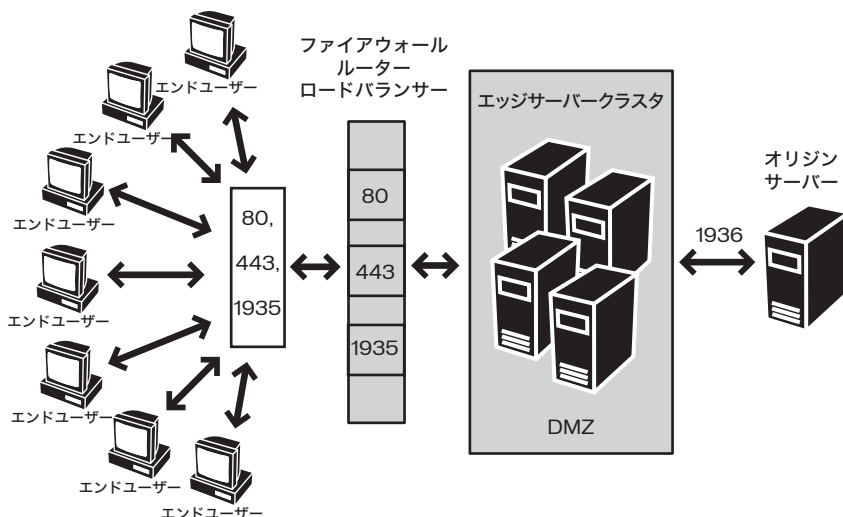
エッジサーバーは、Flash Media Server のサービスに対する特定のゾーンのユーザーからの要求をインターセプトして、それらの要求を収集、集約した上でオリジンサーバーに送ります。オリジンサーバーがエッジサーバーに結果を返すと、そのデータはユーザーのクライアントコンピュータに送り返されます。さらに、エッジサーバーはこの情報を自分自身のキャッシュに保存するため、同じエッジサーバーに割り当てられている他のユーザーやクライアントもこの情報にアクセスすることができます。このような処理方法によって、オリジンサーバーに転送されるサービス要求の数が削減されます。

エッジサーバーは、ビデオストリームや共有オブジェクトなどのデータをキャッシュに保存します。ユーザーから要求されたデータがキャッシュに見つかると、エッジサーバーは、オリジンサーバーへの呼び出しを行うことなく、そのデータを要求元のクライアント（ユーザーのコンピュータ）に返します。このような迂回は、ユーザーには透過的に行われます。オリジンサーバーへの接続を回避することで、エッジサーバーは帯域幅の節約を図ります。この処理方法では、オリジンサーバーには何も要求されません。

DMZ へのエッジサーバーのデプロイメント

エッジサーバーが企業ネットワークの DMZ（非武装地帯）にデプロイされる場合もあります。DMZ とは、組織の信頼できるネットワークとインターネットの信頼できないネットワークの間に存在する、隔離されたネットワークです。DMZ へのデプロイメントにおいては、エッジサーバーは、企業のネットワークを通過するすべての RTMP (Real Time Messaging Protocol) トラフィックに対するプロキシサーバーとして機能します。DMZ にエッジサーバーをデプロイすることで、ユーザーのインターネット接続とオリジンサーバーの間の防衛が一層追加されます。インターネットからの Flash Media Server を宛先とするトラフィックはすべて、エッジサーバーを通過することになります。

DMZ に置かれているエッジサーバーにおいては、ネットワークオペレータはポート 1936 へのアクセスを開くことができ、システムのオーバーヘッドと HTTP トンネリングのリスクを回避できます。エッジサーバーがストリームの分割とストリームのキャッシングを実行するため、エッジサーバーはオリジンサーバーへの接続を効率的に使用し、インターネットの帯域幅コストが削減されます。また、ユーザーにとっても、接続の信頼性が高まるというメリットがあります。



明示プロキシと暗黙プロキシ

エッジサーバーは、設定ファイルの値を使って定義されます。明示プロキシと暗黙プロキシ（匿名プロキシとも呼ばれます）のどちらも、オリジンサーバー上で稼動しているアプリケーションに対するクライアントの要求をインターセプト（傍受）し、集約します。明示エッジサーバーのアドレスには、オリジンサーバーの URI (Uniform Resource Identifier) が接頭辞として付加されます。このような設定によって、クライアント（ユーザーのコンピュータ）にプロキシを認識させます。明示プロキシの URI では、クライアントからオリジンサーバーへの接続要求をインターセプトするエッジサーバーを指定します。

暗黙プロキシの ID (IP アドレスとポート番号) は、クライアントからは隠されます。暗黙プロキシは、到着する URI に含まれているルーティング情報を置換あるいは更新することなく、クライアントをオリジンサーバーに接続します。エッジサーバーは、FPAD (Flash プロキシ自動検出プロセス) を介して、暗黙プロキシとしても定義されます。

Flash Media Server 管理者とネットワーク管理者は、Flash Media Server をデプロイし、1つ以上のエッジサーバーを介して、到着する接続要求をオリジンサーバーにルーティングすることができます。クライアントはプロキシを認識しないため、これらの暗黙エッジサーバーはクライアントからは透過です。暗黙エッジサーバーは、割り込み (Interseption) プロキシと呼ばれることもあります。このようなサーバー構成は、ネットワークを通過する RTMP トラフィックのフローを最適化したいと考える ISP や通信事業者にとって有効です。

オリジンサーバーに対する接続要求を自動的にエッジサーバーまたはプロキシサーバーにルーティングするよう、管理者が明示的に指定することができます。たとえば、アプリケーションが `fms.foo.com` で稼動している場合には、ユーザーのコンピュータからの接続要求を、`fmsproxy.foo.com` という名前でリモートモードで実行するよう構成されている別のサーバー（仮想ホスト）にリダイレクトすることができます。

アプリケーションに対する通常の接続には次のような接続ストリングを使用します。

```
rtmp://fms.foo.com/app/inst
```

上記の接続ストリングの代わりに、既存の URI にプロトコルとホスト名を前に付加し、次のように指定することで、クライアントはエッジサーバー経由にリダイレクトされます。

```
rtmp://fmsproxy.foo.com/?rtmp://fms.foo.com/app/inst
```

この接頭辞に含まれるのは、プロトコルとホスト名、およびオプションでポート番号だけです。この URI は必ず閉じスラッシュで終了し、プロキシの接頭辞とメインの URI を疑問符 (?) で区切ります。

リバースプロキシ

クライアントからの Flash Media Server サービスへの接続要求をエッジサーバーまたはプロキシサーバー経由で送るように設定できない場合もあるでしょう。そのような場合には、1つ以上のプロキシサーバーをセットアップして、オリジンサーバーに近いところに置きます。通常、リバースプロキシは、組織の DMZ 内に置かれ、インターネット経由でファイアウォールの背後に置かれているオリジンサーバーに接続しようとするクライアントからのアクセスを制御します。明示プロキシと匿名プロキシは外部への接続要求をリダイレクトしますが、プロキシサーバーは内部への接続要求をリダイレクトします。

リバースプロキシは、ファイアウォールの外側に置かれているエッジサーバーと同じようにクライアントから到着するすべての接続要求をインターセプトし、それらの要求を認証してからオリジンサーバーに転送します。オリジンサーバーは常に、ファイアウォールの内側に置かれます。リバースプロキシによって、信頼できるネットワーク上のオリジンサーバーへのアクセスを制限します。リバースプロキシは、設定で許可されているものを除くすべてのトラフィックをブロックします。また、信頼できるネットワーク上の他のサーバーとそのリソースへのアクセス試行もブロックします。

リバースエッジサーバーは、オリジンサーバーの存在と場所を隠します。管理者は、Flash Media Server への接続要求がリバースエッジサーバーあるいはプロキシサーバーに自動的にルーティングされるよう、明示的に指定することができます。RTMPS を使用して、セキュアではない接続要求用のポート 80 とセキュアな要求用の 443 をリバースプロキシがリッスンするように設定することができます。

たとえば、アプリケーションが `fms-secure.foo.com` で稼動している場合には、ユーザーのコンピュータからの接続要求を `fms.foo.com` という名前のプロキシサーバー（または仮想ホスト）に送り、そのプロキシサーバーが、アプリケーションが稼動している `fms-secure.foo.com` への接続を確立するようにすることも可能です。クライアントは、他のサーバーに要求が送られることを認識しません。

ルーティング情報

Flash Media Server 管理者またはネットワーク管理者は、`Vhost.xml` 設定ファイルの `RouteEntry` タグを使用して、プロキシサーバーまたはエッジサーバーのルーティング情報を指定することができます。`Vhost.xml` ファイルを使用すると、目的とする宛先への接続をどこへどのようにルーティングするかを設定することができます。

`RouteEntry` タグの `protocol` 属性には、外部への接続のプロトコルを指定します。この属性は、`"rtmp"`（セキュアではない接続）か `"rtmps"`（セキュアな接続）のどちらかに設定する必要があります。

`RouteTable` コンテナタグは、すべての `RouteEntry` タグを保持しています。たとえば、`RouteTable` コンテナに格納されている可能性があるタグとしては、RTMP を使用したセキュアな外部への接続を指定するための `RouteEntry` タグや、セキュアではない RTMP 接続を指定するための `RouteEntry` タグがあります。プロトコルが指定されていないと、外部への接続では内部への接続と同じプロトコルが使用されます。

これらのルーティング用タグの詳細は、『Flash Media Server 管理ガイド』の「`Vhost.xml` ファイル」を参照してください。

エッジサーバーへの接続

特定のエッジサーバーを使用できることがわかっているならば、クライアントはそのエッジサーバーに明示的に接続することができます。ラップされたフォーマットの URI を使用して、明示プロキシへの接続を確立します。

```
rtmp://edge/?rtmp://origin/app
```

プロキシのチェーンに接続することもできますが、その場合には、外部への接続要求をルーティングする一連のプロキシを明示的に特定します。

```
rtmp://edge1/?rtmp://edge2/?rtmp?://edge3/?rtmp://edge4/?rtmp://origin/app
```

チェーン内の次のエッジサーバーに接続が移行すると、Flash Media Server はストリング内の最初のトークンを削除します。`edge1` への接続が確立されると、接続ストリングは次のようになります。

```
rtmp://edge2/?rtmp?://edge3/?rtmp://edge4/?rtmp://origin/app
```

プロキシサーバーの存在の検出

Flash Player 8 は、近接するエッジサーバーを自動的に検出します。エッジサーバーが利用可能になると、Flash Player は、クライアントからの接続をプロキシサーバー経由でオリジンサーバーに自動的にルーティングします。一般的にはクライアントはプロキシ経由で Flash Media Server とやり取りするということを認識していないため、これらのエッジサーバーは暗黙プロキシとして定義されています。暗黙プロキシのこのような使い方によって、現行のアプリケーションを変更することなく、引き続き動作させることができます。Flash Player は、NetConnection オブジェクトに対する読み取り専用のプロパティだけを使用して、このプロキシ情報をクライアントに公開します。

NetConnection クラスの詳細については、『クライアントサイド ActionScript リファレンスガイド』を参照してください。

メモ	以前のバージョンの Flash Player は、近接するエッジサーバーを自動的に検出できません。
----	---

エッジサーバーのチェーン化

オリジンサーバーへの接続の確立時に、任意の数のエッジサーバーをチェーン化することができます。以下の URI は、2 つの明示プロキシをチェーン化して、オリジンサーバーへの要求を送る方法を示しています。

```
rtmp://proxy1/?rtmp://proxy2/?rtmp://origin/app/inst
```

プロキシをチェーン化するための URL の記述では疑問符 (?) を使用するため、Flash Player 7 以前のバージョンでは、共有オブジェクトに問題が発生する可能性があります。

Flash Player 7 以前のバージョンを使用するクライアントのためにこの問題を解決する方法がありません。URI を共有オブジェクトに渡す前に、以下に示す関数 `escape` を使用することで、URI の疑問符に関するこの問題を解決することができます。

```
function escapeURI(uri) {  
    index = uri.indexOf('?');  
    if (index == -1) return uri;  
    prefix = uri.substring(0, index);  
    uri = uri.substring(index);  
    return prefix += escape(uri);  
}
```

共有オブジェクトに渡す前に URI に対してこの関数を呼び出し、その結果を元の URI の代わりに使用します。Flash Player 8 では、Flash Media Server のこの問題は解決されています。

2 つ目の問題は、URI の中で RTMPT を指定する場合に発生します。このプロトコルは、最初の接頭辞だけに使用するものです。次のような URI は有効です。

```
rtmpt://foo/?rtmp://bar/app/inst  
rtmpt://foo:443/?rtmp://bar/app/inst
```

次のような URI は無効です。

```
rtmpt://foo/?rtmpt://bar/app/inst  
rtmp://foo/?rtmpt://bar/app/inst
```

エッジサーバーの設定

Flash Media Server のすべての機能、すなわち、ライブストリーミング、オンデマンドストリーミング、メッセージング、共有オブジェクトの処理、およびスクリプト処理は、アプリケーションインスタンスとして発生します。アプリケーション開発者は、スクリプトを記述することで Flash Media Server アプリケーションを作成し、そのインスタンスは Flash Media Server によって配布されます。エッジサーバーでは、接続の集約、ストリームの分割、ストリームのキャッシング、およびインテリジェントな状態管理を実行する最小規模のアプリケーションが稼働します。作業をこのように別のアプリケーションに分離することで、オリジンサーバーで稼働しているメインのアプリケーションからかなりの部分の処理をエッジサーバーにオフロードしています。

Flash Media Server 管理者とネットワーク管理者は、さまざまな方法で機能を設定することができます。XML 設定ファイルでは、エッジサーバーがクライアントをどのようにオリジンサーバーに接続するかを定義します。これらの設定によって、着信する URL がエッジサーバーとオリジンサーバー間でどのようにルーティングされるかが決まります。1台のコンピュータ上で Flash Media Server を混在モードで稼働させ、そのシステム上のいくつかの仮想ホストはアプリケーションをローカルで実行し、それ以外のアプリケーションはリモートで実行するということができます。

Flash Media Server を設定するには、設定ファイルの XML タグを編集または変更します。代表的な例をいくつか紹介します。

- Vhost.xml ファイルの `Anonymous` タグを使用すると、エッジサーバーを暗黙 (透過) プロキシと明示プロキシのどちらにするかを設定することができます。
詳細については、『Flash Media Server 管理ガイド』の「Anonymous」を参照してください。
- Vhost.xml ファイルの `Mode` タグを使用すると、オリジンサーバーまたはエッジ (プロキシ) サーバーとして実行するように Flash Media Server を設定することができます。
詳細については、『Flash Media Server 管理ガイド』の「Mode」を参照してください。
- Vhost.xml ファイルの `LocalAddress` タグを設定することで、管理者は、送受信されるトラフィックを個々のネットワークインターフェイスに分離する方法でネットワークトラフィックを制御することができます。
詳細については、『Flash Media Server 管理ガイド』の「LocalAddress」を参照してください。
- Application.xml ファイルの `Scope` タグを使用すると、アプリケーションを実行するプロセススコープを確定することができます。アプリケーションとプロキシが `inst` モードで実行するようにこのタグを設定します。それぞれのアプリケーションと仮想ホストには、それぞれのプロセスが存在します。
詳細については、『Flash Media Server 管理ガイド』の「Scope」を参照してください。

設定ファイルの例、タグ構造、および関連するタグへの相互参照やシンタックス、例といったタグの詳細については、『Flash Media Server 管理ガイド』の第 3 章の「設定ファイル」を参照してください。

エッジサーバーのクラスタのデプロイメント

接続されているエッジサーバーの集まりをクラスタとしてデプロイすることもできます。クラスタ内の各エッジサーバーは、オリジンサーバーにアクセスすることができます。クラスタ化によって、オリジンサーバーで稼動しているアプリケーションに対するすべての接続をクラスタ内の多数のエッジサーバーに分散させることができます。オリジンサーバーは常にエッジサーバーの背後にあるため、セキュリティが確保されます。

たとえば、組織 A では、内部ネットワークにエッジサーバーのクラスタを配備します。エッジサーバーは明示プロキシとして定義され、特定のユーザーだけのサービスを実行します。オリジンサーバーに接続したいクライアントまたはユーザーは、最初に、クラスタ内のいずれかのエッジサーバーに接続します。この段階で、接続要求は許可または拒否されます。要求が有効であれば、エッジサーバーは、アプリケーションが稼動しているオリジンサーバーへの接続を確立します。オリジンサーバーがエッジサーバーに結果を返し、それが要求したクライアントに渡されます。さらに、エッジサーバーはその結果をキャッシュに保存して、他のサーバーとクライアントもそれを使用できるようにします。リソースの拡大や再配置に合わせて、この組織では、クラスタにさらにエッジサーバーを追加することや、明示的にクライアントを異なるエッジサーバーに割り当て直すことができます。

組織 B では、エッジサーバーのクラスタを使用して、ネットワークの外に置かれているクライアントからのインバウンドのトラフィックを処理します。このトラフィックは、オリジンサーバー上で稼動している Flash Media Server アプリケーションに送られます。このアプリケーションのユーザーがオリジンサーバーに接続しようとする、クラスタの前に置かれているロードバランサーがクライアントの要求をいずれかのエッジサーバーにルーティングし、そのエッジサーバーは、アプリケーションが稼動しているオリジンサーバーへの接続を確立します。この段階でも、クライアントは、エッジサーバーの存在を認識しません。オリジンサーバーがエッジサーバーに結果を返し、それが要求したクライアントに渡されます。さらに、エッジサーバーはその結果をキャッシュに保存して、他の外部クライアントもそれを使用できるようにします。

このような処理の流れでは、アプリケーションへの接続がクラスタ内の多数のエッジサーバーに分散されます。クラスタ内にあるエッジサーバーがアプリケーションにアクセスできない場合には、そのクラスタ内の別のエッジサーバーがその接続を処理します。オリジンサーバー上で稼動するアプリケーションがインターネット経由で入ってくる接続要求にダイレクトに公開されることはありません。

組織 A の例が外部への接続の説明であったのに対して、組織 B の例は、リバースプロキシを使用した内部への接続がどのように処理されるかを説明したものです。

エッジサーバーのクラスタへの接続

エッジサーバーがクラスタ化されている場合、クライアントには、エッジサーバーが存在し、接続要求を処理しているということはおそらくわかりません。そのため、この方法は、オリジンサーバーのアドレスを隠すことができる、安全かつ賢明な手段であると言えます。Flash Media Server をそのように設定するには、さらに別の方法をとります。

プロキシのクラスタが存在する環境でクライアントが Flash Media Server に接続しようとする場合は、`NetConnection.connect()` メソッドの呼び出しで UDP (User Datagram Protocol) を使用し、FPAD メッセージのローカルブロードキャストを起動します。ネットワーク上でこの FPAD メッセージを受け取る他のエッジサーバーが、クライアントに応答します。クライアントは、接続するエッジサーバーを自動的に選択します。次に、そのエッジサーバーは、オリジンサーバーへの接続を確立します。以上の処理の流れは、明示プロキシのクラスタが、オリジンサーバーへの外部への接続をどのように処理するかの概要を説明したものです。

クラスタへのプロキシサーバーの登録

クラスタ内のそれぞれのプロキシサーバーまたはエッジサーバーには、起動時に数字の ID が動的に割り当てられます。エッジサーバーは、自分自身の存在を知らせるためのメッセージをブロードキャストします。クラスタ内で実行中のすべてのエッジサーバーがこのメッセージを受け取ります。各エッジサーバーは、自分自身の ID を示すメッセージを返します。この ID は、0、1、2、...、N-1 (N は、そのクラスタ内で許されているエッジの数) というように連番で割り当てられます。

起動中のエッジサーバーは、ブロードキャストに対する応答が他のすべてのエッジサーバーから返されるのを待機してから、それらの応答に含まれていない最初の ID を自分自身に割り当てます。停止時には、自分自身がクラスタから抜けることを知らせるメッセージをブロードキャストします。

エッジクラスタ経由のアプリケーションへのアクセス

ネットワークと帯域幅のリソースを可能な限り節約するために、Flash Media Server は、1 つのアプリケーションに対するクライアントからのすべての接続要求を、クラスタ内の同じエッジサーバーに送ります。クラスタ内のすべてのエッジサーバーは、クライアントが接続しようとする URI を基に、アフィニティ値を自動的に計算します。使用できるエッジサーバーを検出するための自動検出処理メッセージをクライアントがブロードキャストすると、アフィニティ (Affinity = 親和性) 値を含むメッセージがそのクライアントに返されます。クラスタ内のすべてのアクティブなエッジサーバーから応答を受け取った後で、クライアントは、アフィニティ値が最も低いエッジサーバーを接続先として自動的に選択します。

アフィニティ値は、接続要求の作業負荷の増減に連動して変化し、クラスタ内のエッジサーバー間で作業負荷を分散させるために使用されます。たとえば、edge0、edge1、edge2 という 3 つのエッジサーバーからなるクラスタではどのようにアフィニティ値が適用されるかを考えてみましょう。edge0 の作業負荷がピークに近づく、このエッジサーバーの接続を edge1 に分割します。両方のアフィニティ値は 1 と計算されます。さらに edge1 の負荷が増えると、今度はその接続を edge2 に分割します。両方のアフィニティ値は 2 と計算されます。さらに edge2 の負荷が増えると、今度はその接続を edge0 に分割します。

エッジサーバーの負荷が下がれば、そのアフィニティ値も下がります。クライアントは常に最もアフィニティ値が低いエッジサーバーに接続しようとするため、このエッジサーバーは、既に次のエッジサーバーに渡していた接続を取り戻すという処理を開始します。この処理の目的は、使用可能なリソースへの接続要求の数を均衡化し、同じアプリケーションインスタンスへのすべての接続が再び同じエッジサーバーを経由するようにすることです。

この場合にも、クライアントのロジックは単純で、最もアフィニティ値が低いエッジサーバーに接続します。クライアントが認識しておく必要があるのは、適切なエッジサーバーに接続するための、クラスタ内のそれぞれのエッジサーバーのアフィニティ値だけです。

リバースプロキシのクラスタリング

明示プロキシと匿名プロキシは外部への接続要求をリダイレクトしますが、リバースプロキシは内部への接続要求をリダイレクトするように設定されます。目的とするアプリケーションへのクライアントからの接続要求は、まず、ロードバランサーを経てからリバースプロキシのクラスタを通り、オリジンサーバーに到達します。リバースプロキシのサーバー構成では、接続要求のためのクライアントのブロードキャストがエッジサーバーまたはオリジンサーバーのどちらにも直接到達することはできず、目的とするアプリケーションにも到達しません。また、プロキシクラスタまたはオリジンサーバーのURIは、クライアント上には見つかりません。このため、リバースプロキシでは、クライアントとアプリケーションを接続するために別の方法が必要になります。

クラスタリングが効果的に動作するのは、同じアプリケーションインスタンスに対して要求するすべてのクライアントがクラスタ内の同じエッジサーバーに接続されていて、そのアプリケーションがそのエッジサーバーにキャッシュされている場合です。リバースプロキシにおいてこのような効果を達成するためには、複数の手順が必要です。

1. クライアントは、オリジンサーバーへの `XML.load` 呼び出しが含まれる HTTP 要求を送信します。
`http://origin[:<port>]/fms/fpad?uri=<uri>`
2. リモートネットワーク上のロードバランサーは、この要求をインターセプトし、それをクラスタ内のいずれかのエッジサーバーにルーティングします。
3. 次に、このエッジサーバーは FPAD クライアントとして動作し、FPAD メッセージをブロードキャストします。
4. クラスタに登録されているエッジサーバーは、自分自身のアフィニティ値を返します。
5. ブロードキャストを実行したエッジサーバーは、アフィニティ値が最も低いエッジサーバーを選択します。
6. このエッジサーバーは、要求された情報をクライアントに送り返します。

```
<?xml version="1.0" encoding="utf-8" ?>
<fpad>
  <proxy>10.133.192.85:1935</proxy>
  <timestamp>627539012</timestamp>
  <userdata></userdata>
</fpad>
```

`<proxy>` タグには、接続先のエッジサーバーの `host:ip address` が含まれます。

7. 次に、クライアントはこの情報を使用して、最初にエッジサーバーに接続し、その後、ラップされた URI を使用してオリジンサーバーに対して明示的な接続を確立します。

```
nc = new NetConnection();
nc.onStatus = function(info)
{
    trace(info.code);
}
uri = "rtmp://yourcompany.com/app/inst";
myXML = new XML();
myXML.onLoad = function(success)
{
    if (success)
    {
        var proxy = this.childNodes[1].childNodes[1].firstChild.nodeValue;
        uri = "rtmp://" + proxy + "/" + uri;
    }
    nc.connect(uri);
}
myXML.load("http://yourcompany.com/fcs/fpad?uri=" + uri);
```

XML.load() の URI に含まれている /fcs/fpad? というコードは、指定されている uri という名前の URI を使用して FPAD メッセージをブロードキャストする必要があることをエッジサーバーに通知します。

オリジンサーバーとプロキシサーバーのセットアップ

オリジンサーバーとプロキシサーバーのインストールと設定にあたっては、以下のガイドラインに従ってください。これらのガイドラインを順守することで、満足のいくパフォーマンスと結果が得られます。

- Macromedia のライセンスでプロキシサーバーまたはプロキシクラスタの使用が許可されていることを確認します。
- すべてのプロキシサーバーとオリジンサーバーを同じ種類のコンピュータにデプロイします。
- ファイル名の不一致を回避するために、同じオペレーティングシステム (Linux または Windows) をすべてのコンピュータで使用します。
Linux ではファイル名の大文字と小文字が区別されますが、Windows においては区別されません。
- 1 台目のサーバーに Flash Media Server をインストールします。
- この Flash Media Server インスタンスを、オリジンサーバーとして設定します。
- ライセンスで許可されている範囲内で、すべてのエッジサーバーまたはプロキシサーバーに Flash Media Server をインストールし、設定します。

- それぞれのオリジンサーバーとプロキシサーバーごとに fms.ini ファイルがカスタマイズされていることを確認します。
fms.ini ファイルは、マシンごとに固有です。
- オリジンサーバーは、1つのポートをリッスンするように設定します。プロキシサーバーは、複数のポートをリッスンするように設定することができます。
- すべてのプロキシがオリジンサーバーにアクセスできることを確認します。
- オリジンサーバーを設置し、同じサブネット上の最も近い場所にそれらのエッジサーバーを設置します。
- クラスタを配備できるライセンスがある場合には、クラスタ内のすべてのプロキシの設定が同じになっていることを確認します。
 - 1つのプロキシで conf ディレクトリを設定し、
 - その conf ディレクトリを他のプロキシの同じ場所にコピーします。
- プロキシを複数デプロイする場合には、ロードバランサーを使用します。
 - ロードバランサーは、クライアントとプロキシの間に置きます。
 - ラウンドロビンモードでプロキシにアクセスするよう、ロードバランサーを設定します。
 - オリジンサーバーのすぐ前にロードバランサーを置かないでください。

エッジサーバーのクラスタをセットアップする手順は以下のとおりです。

1. 最初の Flash Media Server をインストールし、設定します。

Flash Media Server のインストールでは、毎回、同じシリアル番号とライセンスファイルを使用します。

メモ	特別なクラスタライセンスファイルが必要です。詳細については、Macromedia の担当者にお問い合わせください。
----	---

2. この Flash Media Server インスタンスが正しく動作していることを確認します。
 3. Flash Media Server インスタンスを、オリジンサーバーとして設定します。
 4. クラスタ内の次の Flash Media Server をインストールし、設定します。
 5. この Flash Media Server インスタンスを、エッジサーバーとして設定します。
 6. このエッジサーバーがオリジンサーバーを指していることを確認します。
 7. ライセンスで許可されている範囲内で、各エッジサーバーに対して 5 ~ 7 の手順を繰り返します。
- ここまでの作業によって、Flash Media Server のクラスタリングが完了しました。クラスタの能力を拡張するには、ライセンスで許可されている範囲内で新しいエッジサーバーをクラスタに追加し、上記の手順に従って、設定します。

エッジサーバークラスタの保守

エッジサーバーは、その起動時に自分自身の存在をクラスタ内の他のすべてのエッジサーバーにブロードキャストします。停止時には、自分自身がクラスタから離れることをブロードキャストします。あるエッジサーバーがダウンすると、クラスタ内の他のすべてのエッジサーバーは、ダウンしたエッジサーバーからはキーブライブメッセージが送られて来ないことで、この状態の変化を検知します。残ったエッジサーバーは、それぞれが保持しているクラスタのビューを自動的に更新し、そのビューから停止したエッジサーバーを削除します。ユーザーが行うべきことはエッジサーバーの開始と停止だけであり、クラスタ内にどのようなエッジサーバーが登録されているかについてはエッジサーバーが自動的に判断します。

エッジサーバーのキャッシュのクリア

エッジサーバーのキャッシュをクリアするタスクを毎週スケジュールすることをお奨めします。

キャッシュをクリアするスケジュールを設定する手順は以下のとおりです。

1. cache ディレクトリを空にするための cache.bat file ファイルを作成します。

cache.bat ファイルに次のように記述します。

```
del /Q /S <cache_directory>\*.*
```

2. cache.bat ファイルを実行し、cache ディレクトリ内のファイルが削除されていることを確認します。
 - ディレクトリ構造は残りますが、これは想定された動作です。
 - エッジサーバーによってロックされているファイルは削除されませんが、これも想定された動作です。
3. [コントロール パネル]-[タスク]-[スケジュールされたタスクの追加] を選択します。
4. 実行する新しいファイルとして cache.bat を選択します。

日曜日の早朝のように、週の中で処理があまり行われていない時間帯にタスクをスケジュールするのがよいでしょう。

この手順を各エッジサーバーに対して繰り返します。