

2016 年 7 月

お客様各位

SCSK 株式会社
IP ネットワークテクノロジー部

Brocade Virtual Traffic Manager 日本語参考訳のお取扱い

平素は格別のご高配を賜り、厚くお礼申し上げます。

本書のご利用に際し、以下の内容にご同意いただくことが前提になりますこと、予めご了承くださいますようお願い申し上げます。

敬具

記

1. ご利用条件

- (1) 貴社及び SCSK 株式会社（以下、SCSK）と貴社間の商流にて **Brocade Virtual Traffic Manager** ライセンスをご購入いただいたお客様のみ、本書の利用を可能とさせていただきます。
- (2) 本書の内容は **Brocade Communications Systems, Inc.**社が作成した資料の参考和訳で、齟齬がある場合は和訳元の書の内容が優先されます。
- (3) 本書に関するご質問には、SCSK でお答え出来ない場合がございます。
- (4) 本書によって貴社、或いはお客様に損害が生じた場合、SCSK は、損害の発生を予見し得たか否かに関わらず、如何なる損害につきましても一切責任を負わないものとさせていただきます。
- (5) 本書は予告なく内容を変更することがあります。変更により貴社に損害が生じたとしても SCSK は一切責任を負わないものとします。

2. 著作権

- (1) 本書の著作権は、SCSK に帰属します。貴社は **Brocade Virtual Traffic Manager** を使用する目的以外において SCSK の事前の承諾無く本書を転載、複製、変更、配布、販売、貸与等の利用をすることはできません。
- (2) 本書は、SCSK の事前の承諾無く第三者に提供することはできません。
- (3) 貴社にて本書を修正される場合は、SCSK に事前に通知をお願いいたします。また修正した内容の著作権も変わらず SCSK に帰属するものと致します

以上

53-1004324-01
29 March 2016

Brocade Virtual Traffic Manager: User's Guide

バージョン 10.4

BROCADE 

Copyright © 2016 Brocade Communications Systems, Inc. 著作権所有。

ADX、Brocade、Brocade Assurance、the B-wing symbol、DCX、Fabric OS、HyperEdge、ICX、MLX、MyBrocade、OpenScript、The Effortless Network、VCS、VDX、Vplane、および Vyatta は、米国その他の国における Brocade Communications Systems, Inc. の登録商標であり、Fabric Vision および vADX は、米国その他の国における Brocade Communications Systems, Inc. の商標です。本ドキュメントで使用されているその他のブランド、製品、またはサービス名は、他社の商標となっている場合があります。

注意: 本ドキュメントは情報提供のみを目的としており、Brocade によって提供された、または今後提供される何らかの装置、装置の機能、またはサービスに関して、明示的また暗黙的ないかなる保証も規定しません。

Brocade は、通知なしでいつでも本ドキュメントを変更する権利を留保し、その使用に対していかなる責任も負いません。この情報ドキュメントでは、現在利用できない機能が説明されていることがあります。利用可能な機能と製品については、Brocade 営業部にお問い合わせください。本ドキュメントに含まれる技術データを輸出するには、米国政府からの輸出ライセンスが必要になることがあります。

本ドキュメントの作成者および Brocade Communications Systems, Inc. は、本ドキュメントの正確性に関して、または本ドキュメントに含まれる情報もしくは本ドキュメントに付随するコンピュータプログラムに起因する損失、コスト、負債、もしくは損害に関して、個人または組織にいかなる責任も負いません。

本ドキュメントで説明する製品には、GNU 一般公衆利用許諾書またはその他のオープンソースライセンス契約の対象となる「オープンソース」ソフトウェアが含まれることがあります。Brocade 製品に含まれるオープンソースをソフトウェアを調べ、オープンソースソフトウェアに適用されるライセンス条件を表示し、プログラミングソースコードのコピーを入手するには、

<http://www.brocade.com/en/support/support-tools/oscd.html> にアクセスしてください。

Brocade Communications Systems, Incorporated

Corporate and Latin American Headquarters
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
電話: 1-408-333-8000
Fax: 1-408-333-8101
電子メール: info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems China HK, Ltd.
No. 1 Guanghua Road
Chao Yang District
Units 2718 and 2818
Beijing 100020, China
電話: +8610 6588 8888
Fax: +8610 6588 9999
電子メール: china-info@brocade.com

European Headquarters
Brocade Communications Switzerland Sàrl
Centre Swissair
Tour B - 4ème étage
29, Route de l'Aéroport
Case Postale 105
CH-1215 Genève 15
Switzerland
電話: +41 22 799 5640
Fax: +41 22 799 5641
電子メール: emea-info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems Co., Ltd. (Shenzhen WFOE)
Citic Plaza
No. 233 Tian He Road North
Unit 1308 – 13th Floor
Guangzhou, China
電話: +8620 3891 2000
Fax: +8620 3891 2111
電子メール: china-info@brocade.com

目次

Brocade Virtual Traffic Manager: User's Guide	1
前書き.....	17
このガイドについて.....	17
対象読者.....	17
表記上の慣例.....	17
ドキュメントとリリースノート.....	18
トラフィックマネージャのドキュメント.....	19
トラフィックマネージャのオンラインヘルプ.....	19
トラフィックマネージャのオンライン情報.....	19
テクニカルサポートの利用またはエラーのレポート.....	20
Web アクセス.....	20
電子メールと電話によるアクセス.....	20
CHAPTER 1 トラフィックマネージャの概要.....	21
トラフィックマネージャの紹介.....	21
一般的なデプロイ.....	22
トラフィックマネージャ製品のバリエーション.....	22
開発者モード.....	23
対応プラットフォーム.....	24
サポートされるクラスタの組み合わせ.....	24
各章の概要.....	26
CHAPTER 2 ネットワークレイアウト.....	30
ネットワーク構成の基本コンポーネント.....	30
専用管理ネットワーク.....	31
クラスタのサイジング.....	32
フロントエンドサーバ.....	33
IP トランスペアレンシ.....	34
ルーティング設定.....	36
ローカルルーティングの制限事項.....	37
IP トランスペアレンシとトラフィックマネージャクラスタ.....	37
トラフィック IP アドレスとトラフィック IP グループ.....	37
トラフィック IP アドレスのモード.....	38
構成例.....	39
クラスタでの IP トランスペアレンシの使用.....	43
ルートヘルスインジェクションおよびネットワーク.....	46
IPv6 について.....	51
トラフィックマネージャでの IPv6 の主な機能.....	51
技術的制約.....	52
CHAPTER 3 初期設定.....	54
アーキテクチャコンセプト.....	54
最初のサービスの管理.....	56
ウィザードを使用した仮想サーバとプールの作成.....	56

手動でのプールと仮想サーバの作成	58
クラスタの作成	59
クラスタへの組み込み	59
トラフィック IP グループが設定されたクラスタへの組み込み	63
CHAPTER 4 仮想サーバ	64
ルールの適用	66
SSL 復号化	68
サービスプロテクションクラス	69
帯域制御クラス	69
サービスレベルモニタリングクラス	70
HTTP コンテンツキャッシング	70
Web Accelerator	71
HTTP コンテンツ圧縮	72
コンテンツ圧縮のコントロール	72
コネクション分析	73
設定オプション	73
コネクション分析の使用	75
Request Logging (リクエストロギング)	75
ファイルへのリクエストロギング	75
リモートリクエストロギング	76
リクエストロギングのコントロール	76
コネクション管理	77
エラーの処理	82
コネクション用メモリ制限	83
CHAPTER 5 プール	85
負荷分散	85
局所認識リクエスト分散(LARD)	87
セッション維持	88
帯域制御	89
ヘルスモニタリング	89
SSL 暗号化	90
コネクション管理	91
プールコネクション制限	91
はじめに	91
プールコネクション制限	92
クラスタ環境でのコネクション制限	92
コネクションキューイング	92
考慮事項	92
プールコネクション制限の有効化	93
コネクション制限のトラッキング	94
コネクション制限のテスト	94
バックエンドのフォールトトレランス	95
ノードのドレインと無効化	98
Drain a Node (ノードのドレイン)および Disable a Node (ノードの無効化)ウィザードの 使用	99
オートスケーリング	99
はじめに	99
オートスケーリングの動作	100

設定.....	101
DNS を用いたオートスケーリング	109
CHAPTER 6 トラフィック IP グループとフォールトトレランス	111
フォールトトレランス.....	111
トラフィック IP アドレスとトラフィック IP グループ	111
トラフィック IP グループ内でのトラフィックの分配.....	112
トラフィック IP アドレスの選択	113
トラフィック IP グループの作成	113
トラフィックの分配.....	114
パッシブマシン	116
トラフィック IP グループの無効化	116
インターフェースとサブネットのマッピング(トラフィック IP ネットワーク).....	116
フォールトトレランスの設定.....	117
フォールトトレランスの設定	118
トラフィックマネージャのフォールトトレランスチェック	120
ヘルスブロードキャスト	121
クラスタのヘルス判定	121
フェールオーバー.....	122
トラフィック IP アドレストランスファ(シングルホストモード).....	122
トラフィック IP アドレストランスファ(シングルホストモード).....	123
トラフィック IP アドレストランスファ(RHI モード).....	123
障害からの復旧	123
フォールトトレランスのデバッグとモニタリング	124
BGP 接続性の設定.....	124
BGP ルータ ID の設定.....	125
BGP ネイバーの管理	125
OSPFv2 の接続性の設定.....	127
OSPFv2 の IP アドレスの設定	128
隣接モニタリングの設定	128
CHAPTER 7 トラフィックマネージャ管理インターフェースの主要機能	130
ホームページ.....	130
Services (サービス) > Configuration Summary (設定概要).....	131
カタログ	131
ライセンス管理.....	133
ライセンスキーの追加と削除	133
System (システム) > Global Settings (グローバル設定)	134
System (システム) > Backups (バックアップ).....	136
System (システム) > Backups (バックアップ) > Partial Backups (部分バックアップ)	138
アクティビティモニタリング	142
Activity (アクティビティ) > Current Activity (現在のアクティビティ).....	143
Activity (アクティビティ) > Historical Activity (ヒストリカルアクティビティ)	148
Plot Data (データのプロット)をクリックし、現在の設定を使用してグラフを 表示します。データを .tsv (タブ区切り変数)ファイルとしてダウンロードして、 解析に利用することもできます。Activity (アクティビティ) > Map (マップ).....	149
Activity (アクティビティ) > Connections (コネクション).....	149
Activity (アクティビティ) > Draining Nodes (ノードのドレイン)	151
Activity (アクティビティ) > View Logs (ログ表示).....	152
クラウドクレデンシャル.....	152

Amazon EC2 クレデンシャルの IAM ロール	154
Web Application Firewall	154
概要	155
Application Firewall の有効化	155
トラフィックマネージャの管理 UI で利用できる Application Firewall 機能	156
System (システム) > Application Firewall ページ	157
Enforcer と Decider	159
Enforcer ルール	159
ユーザ管理	161
ソフトウェアの更新	162
CHAPTER 8 TrafficScript ルール	163
概要	163
TrafficScript の例	164
TrafficScript に関するドキュメント	166
トラフィックマネージャでのルールの使用	166
トラフィックマネージャでのルールの使用	167
カタログでのルールの作成	167
カタログへのルールのアップロード	171
仮想サーバへのルールの適用	171
ルールの例	172
コンテンツタイプによるルーティング	172
時刻に基づくアクセス制限	172
カスタムのプライオリティ設定	173
サービスレベルの管理	174
XML トラフィックに基づくルーティング	174
CHAPTER 9 TrafficScript での認証のサポート	177
概要	177
認証コードの設定	177
TrafficScript ルールの設定	181
仮想サーバの設定	182
CHAPTER 10 Java Extensions	183
Java について	183
Java Extension の呼び出し	183
トラフィックマネージャで Java を使用するための設定	184
必要条件	184
Java Extension のコンパイル	184
トラフィックマネージャへの Java Extensions のロード	185
トラフィックマネージャの Java Extension Runner の設定	185
CHAPTER 11 プロトコルのサポート	187
基本的な TCP プロトコル	187
Server First プロトコル	187
Client First プロトコル	188
「サーババナー」を使った Server First	189
Generic Streaming プロトコル	190

HTTP	191
SSL	193
SSL ハンドシェイクの保護	194
SSL コネクションの再ネゴシエーション保護	194
SMTP (Simple Mail Transport Protocol)	196
FTP	197
FTP ソースポート	198
SSL-wrapped FTP (FTPS)	199
SSL-wrapped FTP の使用例	200
Real Time Streaming プロトコル	201
RTSP サービスの設定	203
Session Initiation Protocol	203
SIP の機能	203
トラフィックマネージャと SIP プロトコル	204
プロキシサーバでトラフィック管理をサポートするための設定	205
トラフィックマネージャでの SIP サービスの設定	206
トラフィックマネージャでの SIP 動作モード	206
SIP のその他の設定	208
UDP ベースの SIP サーバとの通信	210
CHAPTER 12 セッション維持	212
セッション維持とは	212
セッション維持の設定	213
セッション維持の有効化	213
セッション維持手法の選択	214
ノードへのセッション維持マップの名前解決	219
ノード障害発生時のオプション	220
コネクションのドレイン	221
セッション維持キャッシュのサイジング	222
セッション維持とマルチホストトラフィック IP アドレスの併用	223
UDP プロトコルでのセッション維持	224
例	224
ユニバーサル PHP セッション維持	224
CHAPTER 13 SSL 暗号化	226
SSL の概要	226
サーバ認証	226
クライアント認証	227
暗号化データ転送	228
トラフィックマネージャの SSL 機能	228
復号化と暗号化	228
SSL 証明書カタログ	228
SSL 復号化ウィザード	229
SSL 証明書の設定	230
自己署名 SSL 証明書の新規作成	230
証明書データの管理	232
証明書署名リクエストの作成	232
新規 SSL 証明書のインポート	233
中間証明書の使用方法	234
認証局の証明書と CRL ファイルの管理	235

SSL 復号化.....	236
SSL 復号化の設定	236
1 台の仮想サーバを使用した複数のサイトへの対応	239
暗号および TLS バージョンの設定	240
クライアント証明書	242
OCSP の設定	243
SSL セッション ID キャッシュ	247
OCSP Stapling キャッシュ	247
SSL 暗号化.....	248
SSL 転送における IP アドレスの維持.....	249
SSL 暗号処理専用デバイスの使用	250
SSL デバイスを使用するためのトラフィックマネージャの設定	251
SSL デバイスの正常な動作の検証.....	254
Connect to Microsoft Azure Key Vault (Microsoft Azure Key Vault に接続)ウィザードの 使用.....	255
Connect to NetHSM ウィザードの使用	258
セキュアなデバイスに保存された鍵と証明書の識別	261
CHAPTER 14 ヘルスモニタリング	262
どのノードをモニタリングするか.....	262
複数プールでのノードの使用	262
パッシブヘルスモニタリング	263
失敗したリクエストの再試行	264
ノード障害.....	265
パッシブモニタリングの有効化と無効化	265
ヘルスモニタの概要.....	266
モニタカタログ	266
内蔵ヘルスモニタ	267
カスタムヘルスモニタ	269
ノード単位のモニタとプール全体のモニタ	271
ヘルスモニタの使用.....	271
プールへのモニタの適用	271
外部プログラムモニタ	271
トラフィックマネージャへのモニタのアップロード	272
Perl でのモニタの作成.....	273
CHAPTER 15 サービスプロテクション	274
リスクの分類.....	274
DoS(サービス妨害).....	274
Web ワームとウイルス	274
DDoS(分散型サービス妨害)攻撃	275
不正形式の HTTP 攻撃	275
ファイアウォールおよびその他のセキュリティ対策	275
保護機能	275
ネットワークアクセス制限	276
コネクション制限.....	276
不正形式の HTTP のフィルタ	276
ルールベースの保護.....	276
サービスプロテクションの有効化.....	277
サービスプロテクションクラスの追加.....	277

基本設定	277
同時接続	278
接続レート	279
アクセス制限	279
HTTP 固有の設定	280
サービスプロテクションルール	281
仮想サーバへのサービスプロテクションクラスの適用	281
サービスプロテクション性能	281
CHAPTER 16 帯域制御	282
帯域制御とは	282
帯域制御の設定	283
カタログへの帯域クラスの追加	283
仮想サーバへの帯域クラスの割り当て	284
プールへの帯域クラスの割り当て	284
TrafficScript を使用した帯域クラスの選択	285
CHAPTER 17 リクエストレートシェイピング	286
リクエストレートシェイピングとは	286
リクエストレートシェイピングクラス(レートクラス)の設定	287
カタログへのレートクラスの追加	287
レートクラスの使用	288
レートキュー	288
選択的レートシェイピング	289
さらにきめ細かいレートシェイピング	289
レートシェイピング Web スパイダ	290
リクエストレートシェイピングのグラフ化	291
CHAPTER 18 サービスレベルモニタリング	292
サービスレベルモニタリングについて	292
サービスレベルモニタリングクラス(SLM クラス)の設定	293
カタログへの SLM クラスの追加	293
仮想サーバへの SLM クラスの適用	294
TrafficScript による SLM クラスの適用	294
SLM クラスの TrafficScript の例	295
「FrontPage スクリプトのみ」のサービスレベルモニタリング	295
サービスレベルモニタリングによるリソースのプライオリティ設定	295
SLM クラス適合レートのグラフ化	296
CHAPTER 19 コンテンツキャッシング	297
はじめに	297
コンテンツキャッシングの設定	298
仮想サーバへのコンテンツキャッシングの適用	298
ライフタイムの設定	298
Web キャッシュのメモリ使用量の設定	300
キャッシュアクティビティのモニタリング	301
ディスクベースのキャッシングの設定	302
キャッシングポリシー	303

リクエスト	303
レスポンス	303
バリエーション	304
バイトレンジ	304
ETag	304
TrafficScript を使用したコンテンツキャッシングのコントロール	305
HTTP リクエストの処理	305
HTTP レスポンスの処理	305
TrafficScript キャッシュコントロール関数	305
キャッシュ内の古いコンテンツの削除	307
キャッシングされているコンテンツの手動削除	307
キャッシングされているコンテンツのプログラムによる削除	308
 CHAPTER 20 Web コンテンツを最適化するための Brocade Web Accelerator の使用	309
はじめに	309
動作モード	309
サービス用の Web Accelerator の設定	310
Web Accelerator ウィザード	311
アプリケーションスコープ	311
Web Accelerator プロファイル	312
Web Accelerator による変化の測定	317
Web Accelerator がアクティブであることのチェック	317
Stealth モードを使用して Web Accelerator をテスト	318
Web ページの速度の測定	318
ツール	319
カスタムアクセラレーションプロファイルについて	320
アクセラレーション設定	323
最適化手法について	328
Web ページの速度の原則	328
リソースの指定と URL のバージョンング	332
コンテンツ分配ネットワークの使用	336
Web Accelerator のトラブルシューティング	336
予想外の動作の コントロール	336
その他のトラフィックマネージャ機能との相互作用	337
ランタイムエラー	341
画像エラー	342
CSS エラー	343
JavaScript エラー	344
その他の設定可能なグローバル設定	345
 CHAPTER 21 イベント処理およびアラート	347
概要	347
イベントタイプ	348
新しいイベントタイプの作成	349
アクション	350
アクションのテスト	352
イベントハンドラの設定	352
重複イベント	353
カスタムアクション	353
プログラムやスクリプトの呼び出し	353

SOAP メッセージの送信	354
TrafficScript または Java Extensions からのイベント報告	356
例	357
CHAPTER 22 システムレベルの設定	359
ネットワーク設定	359
ホスト名と IP アドレスの設定	359
VLAN の設定	360
DNS の設定	361
ルーティングの設定	362
リターンパスルーティングの設定	362
IP フォワーディングおよびネットワークアドレス変換(NAT)の設定	366
時間と日付の設定	369
時刻の手動設定	370
NTP サーバの使用	370
トラフィックマネージャからの時刻の同期	370
トラフィックマネージャへのリモートログイン	370
カスタムカーネルパラメータの入力	371
パラメータの追加または変更	371
既存のエントリ	372
CHAPTER 23 システムセキュリティ	373
ファイアウォールと OS の設定	373
ファイアウォールの設定手法	373
トラフィックマネージャでのファイアウォールの設定	373
ネットワーク設計	375
Unix ユーザパーミッション	375
ファイルシステムのセキュリティ	376
OS の設定	376
CHAPTER 24 管理サーバのセキュリティ	378
管理サーバの基本設定	378
管理サーバの SSL 証明書の変更	378
管理サーバに対するアクセスの制限	379
管理サーバポートの変更	379
Traffic Manager SSH サーバのセキュリティ	380
クラスタ通信	381
管理サーバおよび内部接続の SSL 設定	383
REST API へのアクセス	384
ユーザ管理	384
ユーザ認証	384
ローカルユーザ	385
認証コード	386
認証コードのテスト	391
パーミッショングループ	392
ログインタイムアウト	393
サスペンドされているユーザ	393
ログインセキュリティとログイン行動	393
ログイン情報バナー	395

イベントおよび監査ログ	396
CHAPTER 25 トラフィックマネージャのコントロール API	398
トラフィックマネージャのコントロール API の概要	398
例: 実行中の仮想サーバのリスト表示	398
Perl と SOAP::Lite	398
C Sharp または Mono	399
その他の例	401
CHAPTER 26 コマンドラインインターフェース	402
CLI へのアクセス	402
許可	403
コマンド	404
コントロール API メソッド	405
内蔵コマンド	408
CLI のスクリプティング	412
スクリプトの出力	414
CHAPTER 27 zconf を使用したきめ細かい設定のインポート/エクスポート	415
はじめに	415
zconf の使用	415
完全バックアップのエクスポート	416
設定のリスト表示	417
部分インポート	417
CHAPTER 28 マルチサイトクラスタの管理	418
はじめに	418
アクティブ化と非アクティブ化	419
主なコンセプト	419
ロケーション設定	419
クラスタ	420
デプロイシナリオ	420
第 2 のトラフィックマネージャロケーションの作成と管理	420
マルチサイトクラスタへの新しいトラフィックマネージャの追加	421
複数の既存トラフィックマネージャクラスタのマージ	421
設定	423
ロケーションの設定	423
トラフィックマネージャのロケーションの設定	424
ロケーション固有の設定	424
ホームページの変更点	425
ワールドマップ	426
トラフィックの可視化	426
CHAPTER 29 トラフィックマネージャの DNS サーバ	428
DNS の概要	428
はじめに	428
DNS のレイアウト	428
権威の委任	428

名前解決.....	429
リソースレコード.....	429
ゾーンファイル.....	430
名前解決プロセス.....	432
サポートされる DNS 機能.....	433
RFC 1034 準拠の実装機能.....	434
RFC 1035 準拠の実装機能.....	434
RFC 1034 に関する除外事項.....	435
RFC 1035 に関する除外事項.....	435
その他の実装機能.....	436
その他の除外機能.....	437
DNS サーバの設定.....	437
設定概要.....	437
トラフィックマネージャへの DNS ゾーンファイルのアップロード.....	438
トラフィックマネージャのゾーンの設定.....	439
DNS 仮想サーバの設定.....	439
CHAPTER 30 グローバル負荷分散.....	441
概要と前提条件.....	441
グローバルサーバ負荷分散について.....	442
トラフィックマネージャ内での GSLB.....	443
デプロイ計画.....	443
トラフィックマネージャ の位置決め.....	443
デプロイ手法.....	444
存続時間(TTL)フィールド.....	449
トラフィックマネージャ GLB デプロイのコンポーネント.....	449
GLB ロケーション.....	450
GLB サービス.....	450
GLB を設定した仮想サーバとプール.....	451
DNS サーバ.....	451
サービス IP アドレス.....	451
サービスモニタ.....	451
GLB の設定.....	452
概要.....	452
GLB ロケーションの定義.....	453
サービスモニタの作成.....	454
GLB サービスの作成.....	454
DNS サーバプールの作成.....	460
DNS 仮想サーバの作成.....	460
トラフィックの可視化.....	462
現在のアクティビティグラフ.....	462
ヒストリカルアクティビティグラフ.....	462
Connections (コネクション)ページ.....	463
GLB リクエストログ.....	463
Dig による DNS のテスト.....	463
トラフィックマネージャの GeoIP データベースの拡張.....	464
未認識 IP アドレス.....	464
トラフィックマネージャの GeoIP データベースの拡張.....	465
IP アドレスマッピングのテスト.....	465
トラフィックマネージャのクラスタ設定の更新.....	466

CHAPTER 31	トラフィックマネージャの FIPS 認証.....	467
	FIPS について.....	467
	FIPS モード.....	467
	FIPS 140-2	467
	FIPS 140-2 とトラフィックマネージャ	468
	FIPS モードのデプロイ	471
	準備.....	471
	FIPS モードの有効化.....	476
	FIPS モードでの動作.....	476
CHAPTER 32	Kerberos 強制委任のサポート	478
	Kerberos プロトコル.....	478
	Kerberos プロトコルの遷移と制約付き委任	478
	プロトコル遷移.....	478
	制限付き委任.....	478
	Kerberos を使用する論理的根拠.....	479
	トラフィックマネージャでの Kerberos 設定	479
	トラフィックマネージャのサービスプリンシパル	479
	仮想サーバプロトコル遷移の設定	482
	プールプロトコル遷移の設定	483
CHAPTER 33	トラブルシューティング	484
	ツールおよび手法.....	484
	診断とイベントログ	484
	リクエストとレスポンスのモニタリング	485
	コネクション状況レポート	486
	リクエストログ	486
	高度なロギング	487
	イベントのモニタリング	487
	コネクションの詳細なデバッグ	487
	個々のノードのテスト.....	489
	設定の確認	489
	トラブルシューティングのヒント	490
	テストリクエストの生成.....	490
	自動バックエンドフェールオーバーの確認	491
	自動フロントエンドフェールオーバーの確認	492
	起こりやすい問題.....	492
	root として起動されない.....	492
	コネクションが拒否される	493
	トラフィック IP アドレスが正しく設定されていない.....	493
	トラフィックマネージャがプロトコル開始前にコネクションを破棄する	493
	Web サーバが Error 400 を返す.....	494
	間違ったポート番号が設定されている	494
	ファイル記述子を使い果たした	494
	ディスク空き容量を使い果たした	495
	サポートを利用するには.....	495

CHAPTER 34	用語集.....	497
CHAPTER 35	ソフトウェア使用許諾契約書	504
	Berkeley DB コード(バージョン 1.85)の使用許諾条件.....	504
	RSA PKCS11	505
	OpenLDAP コード、バージョン 2.4.23 の使用許諾条件	505
	PCRE2 の使用許諾条件	506
	Libnet の使用許諾条件	508
	Yahoo! UI Library の使用許諾条件	509
	ssleay 暗号化ライブラリの使用許諾条件	509
	libxml2 および libxslt の使用許諾条件	510
	Java Servlet API の使用許諾条件	511
	expat XML パーサの使用許諾条件	512
	MooTools の使用許諾条件	512
	OpenLayers の使用許諾条件	513
	rsync の使用許諾条件	515
	mod_imap.c の使用許諾条件	515
	Antlr および libantlr の使用許諾条件	516
	es3-grammar の使用許諾条件	517
	jsoncpp の使用許諾条件	518
	libjpeg の使用許諾条件	519
	libunwind の使用許諾条件	520
	Perl JSON ライブラリの使用許諾条件	521
	OpenSSL の使用許諾条件	524
	WebP の使用許諾条件	526
	Flex の使用許諾条件	527
	CryptoJS の使用許諾条件	528
	zlib の使用許諾条件	529
	zlib.js の使用許諾条件	530
	XML::Twig の使用許諾条件	530
	MIT Kerberos の使用許諾条件	531
	Libedit の使用許諾条件	552
	ZebOS の使用許諾条件	554
	Curl の使用許諾条件	554
	Jansson の使用許諾条件	555
	Digest::SHA の使用許諾条件	556
	Sys::SysLog の使用許諾条件	556
CHAPTER 36	索引	557

前書き

『*Brocade Virtual Traffic Manager: User's Guide*』へようこそ。この前書きでは、本マニュアルに記述されている情報、ドキュメント全体で使用されている表記上の慣例、補足資料、およびお問い合わせ先が概説されています。本章は以下のセクションで構成されています。

- [「このガイドについて」、次のセクション](#)
- [「ドキュメントとリリースノート」 \(18 ページ\)](#)
- [「テクニカルサポートの利用またはエラーのレポート」 \(20 ページ\)](#)

このガイドについて

『*Brocade Virtual Traffic Manager: User's Guide*』では、Brocade Virtual Traffic Manager (トラフィックマネージャ)を設定し、管理する方法について説明します。

はじめに、トラフィックマネージャのインストールおよび負荷分散サービスの基本的な設定方法について、お使いの製品に適用される『*Brocade Virtual Traffic Manager: Installation and Getting Started Guide*』を参照してください。

本マニュアルでは、トラフィックマネージャのリリース 10.4 の特徴および機能を説明します。

対象読者

本マニュアルは、Web サービスおよびインフラストラクチャの運営および管理に精通するシステム管理者のために書かれています。

本マニュアルは、読者がネットワーキング技術に精通していることを前提としています。

表記上の慣例

本マニュアルでは、新しい用語の紹介、画面表示の説明、およびコマンド構文の説明に以下の表記上の慣例を標準セットとして使用しています。

慣例	意味
斜体	斜体文字で記されている部分は、新しい用語や強調する言葉を示しています。
太字体	太字体で記されている部分は、CLI コマンドおよび UI コン

トロールを示しています。

Courier フォント	Courier フォントで記されている部分は、コード例を示しています。 <pre>amnesiac > enable amnesiac # configure terminal</pre>
◇	山括弧内に記されている部分は、ユーザが指定する値を示しています。 interface <ipaddress>
[]	角括弧内に記されている部分は、オプションのキーワードまたは変数を示しています。 ntp peer <addr> [version <number>]
{ }	中括弧内に記されている部分は、必須のキーワードまたは変数を示しています。 {delete <filename> upload <filename>}
	バーティカルバーは、バーの左右のいずれかのキーワードまたは変数を選択できることを示しています(このように表記されるキーワードまたは変数は、オプションまたは必須のいずれの場合もあります)。 {delete <filename> upload <filename>}

ドキュメントとリリースノート

全ての Brocade ドキュメントの最新バージョンを入手するには、Brocade の Web サイト(<http://www.brocade.com/en/products-services.html>)で対象製品のページを参照してください。

詳細情報が必要な場合は、既知の問題、使用方法に関するドキュメント、システム要件、および一般的なエラーメッセージについて、Brocade Knowledge Base を参照してください。タイトルを参照するか、キーワードや文字列で検索することができ

ます。Brocade Knowledge Base にアクセスするには、MyBrocade Web サイト (<https://login.brocade.com>) にログインします。

各ソフトウェアリリースには、リリースノートが付属しています。リリースノートには、ソフトウェアの新機能と、既知の修正済みの問題が明記されています。最新バージョンのリリースノートを手に入れるには、MyBrocade Web サイト (<https://login.brocade.com>) にログインします。

インストールと設定のプロセスを開始する前に、リリースノートをお読みください。

トラフィックマネージャのドキュメント

トラフィックマネージャには、トラフィックマネージャの一連の機能について詳述した包括的なユーザガイドが付属しています。製品別の入門ガイドと、TrafficScript ルール言語や製品 API などの追加機能を解説した一連のリファレンスガイドもあります。

以下の Brocade の Web サイトの Brocade Virtual Traffic Manager ページから、全てのサポート対象バージョンのトラフィックマネージャのドキュメントをダウンロードできます。

<http://www.brocade.com/en/products-services/application-delivery-controllers/virtual-traffic-manager.html>

トラフィックマネージャのオンラインヘルプ

管理 UI の全てのページにある **Help** (ヘルプ) ボタンをクリックすると、そのページに関する詳細なヘルプ情報が表示されます。目次を表示し、索引ページを使用して、オンラインヘルプの各ページに移動することもできます。

Rules (ルール) > **Edit** (編集) ページにも、全てのトラフィックマネージャ TrafficScript 関数のクイックリファレンスガイドである TrafficScript Help (TrafficScript ヘルプ) へのリンクがあります。

トラフィックマネージャのオンライン情報

トラフィックマネージャ製品の使用については、以下の Web サイトを参照してください。

<http://www.brocade.com/en/products-services/application-delivery-controllers/virtual-traffic-manager.html>

ホワイトペーパーやその他の資料など、さらに多くのドキュメントを参照するには、Brocade が提供する以下のコミュニティ Web サイトにアクセスしてください。

<http://community.brocade.com/>

テクニカルサポートの利用またはエラーのレポート

Brocade は、当社製品への投資の費用対効果を確実に維持するために全力を尽くしています。サポートが必要な場合やドキュメントに誤りが見つかった場合は、以下のいずれかの方法で Brocade にお問い合わせください。

Web アクセス

Brocade の Web サイトには、本ガイドおよびトラフィックマネージャに関する全ての他のユーザガイドの最新バージョンがあります。詳細については、<http://www.brocade.com/en/products-services/application-delivery-controllers/virtual-traffic-manager.html> を参照してください。

エラーをレポートするには、MyBrocade Web サイト(<https://login.brocade.com>)にログインし、**Support Cases** (サポートケース)をクリックして、新しいサポートケースを開きます。ケースの説明で、ドキュメントタイトルを必ず指定してください。

電子メールと電話によるアクセス

最新の電子メールと電話の連絡先情報については、<http://www.brocade.com/en/support.html> を参照してください。

CHAPTER 1 トラフィックマネージャの概要

トラフィックマネージャの紹介

トラフィックマネージャ製品ファミリは、高可用性と、アプリケーションベースのトラフィック管理および負荷分散を実現するソリューションを提供しています。これらのソリューションは、あらゆるアプリケーショントラフィックをコントロールするとともに、トラフィックにインテリジェンス、セキュリティ、そして高いフォールトトレランスをもたらします。

トラフィックマネージャは、HTTP (Web)やメディア送信などの TCP/UDP ベースのサービス、Web サービスなどの XML ベースのサービスなど、ビジネスクリティカルなサービスをホストする企業を対象とした製品です。

トラフィックマネージャ独自のプロセスアーキテクチャでは、大容量のネットワークトラフィックも効率よく処理することができます。また、このアーキテクチャが備える拡張性により、ニーズに応じてクラスタにフロントエンドトラフィックマネージャやバックエンドサーバを増設することが可能です。クラスタのサイズに制限はなく、トラフィックマネージャの性能は、使用プラットフォームの性能が高まるほど向上します。

トラフィックマネージャファミリは、新たなニーズにも適応して拡張することのできる非常に優れたソリューションです。独自の **TrafficScript** 言語と組み込み式 **Java Extensions** を使用すれば、ニーズに応じた高度なトラフィック管理ルールを作成し、リクエストとレスポンスをチェック、変換、管理、ルーティングすることができます。**TrafficScript** ルールでは、任意の TCP または UDP ベースのプロトコルでのコネクションを管理することが可能です。

トラフィックマネージャ製品は、導入直後から優れたセキュリティを発揮し、不正侵入や DoS 攻撃を阻止します。また、非常に高速で強力な **SSL(Secure Sockets Layer)**¹暗号化技術を採用しているため、多数のセキュアなコネクションを効率よく復号化および再暗号化することができます。エンドツーエンドのセキュリティを確保しながら、**TrafficScript** ルール、セキュリティポリシー、およびその他のコンテンツベースの計算を暗号化されたリクエストに適用することができます。

トラフィックマネージャは、クリティカルな高可用性ソリューションとして、クラスタ冗長機能を提供しています。この機能により、アクティブ/パッシブ(スタンバイ)構成のフロントエンドサーバを無制限にデプロイできるようになります。アクティブサーバに障害が発生した場合は、スタンバイサーバが自動的に起動します。さらにそのサーバにも障害が発生した場合は、別のスタンバイサーバが負荷を引き継

¹ トラフィックマネージャは、SSL とその後継アルゴリズムのトランスポート層セキュリティ(TLS)をサポートしています。個別に指定のない限り、このガイドを通じて SSL という用語は、一般に、両方のアルゴリズムを指します。

ることができます。このため、システム内に単一障害点が発生することはありません。

一般的なデプロイ

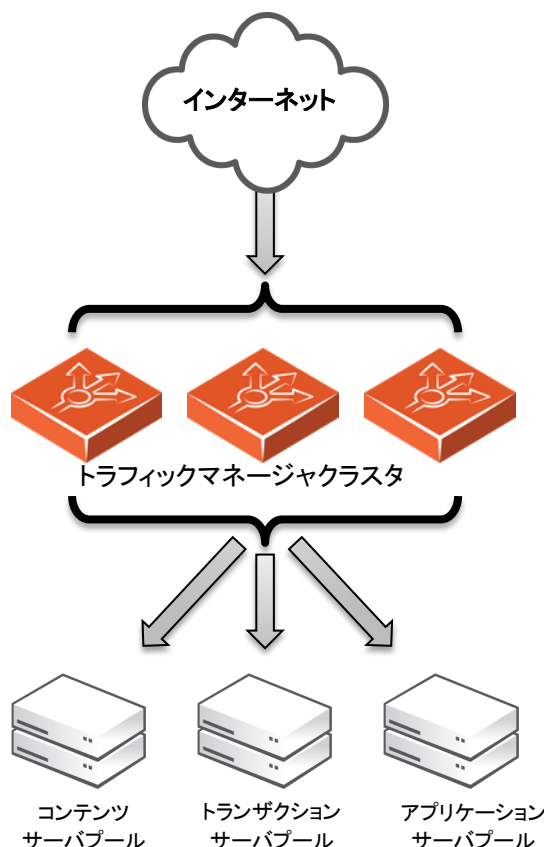


Fig 1. トラフィックマネージャのクラスタを使用した一般的なデプロイ

トラフィックマネージャ製品のバリエーション

トラフィックマネージャ製品ファミリは、ソフトウェア、ハードウェアアプライアンス、仮想アプライアンス、クラウドインスタンスなど様々な構成で提供されています。どの構成でもコアとなるトラフィックマネージャソフトウェアは同じですが、提供される機能レベルは、バージョンごとにライセンスキーに応じて異なる場合があります。ライセンスキーが提供されない所では、トラフィックマネージャは開発者モード(以下のセクション参照)で動きます。

本マニュアルでは、全てのオプションを有効化したトラフィックマネージャソフトウェアの全機能について説明します。ご使用の製品バージョンによっては、ここに記載されていても、存在しない、または表示されない機能がある場合があります。存在していても、ライセンスキーで有効にならない機能は、管理 UI に淡色表示され、選択できません。

例えば、グローバル負荷分散、サービスレベルモニタリング、レートシェイピング、オートスケーリング、および帯域制御は高度な製品機能であり、ご使用の構成によっては有効化できない場合があります。

その他にも、適切なライセンスキーのアップグレードによってのみ利用可能になる、トラフィックマネージャの2つのオプションコンポーネントが用意されています。

- **Brocade Virtual Web Application Firewall (Web Application Firewall):** Web アプリケーションに対して、高度な攻撃の検出と保護を実現します。トラフィックマネージャのインフラストラクチャにおけるこのコンポーネントの役割については、CHAPTER 7 の「Web Application Firewall」を参照してください。製品の詳細と詳しい手順については、『**Brocade Virtual Web Application Firewall User Guide**』を参照してください。このドキュメントは、以下の Brocade の Web サイトからダウンロードできます。

<https://www.brocade.com/vadc-docs>

- **Brocade Web Accelerator (Web Accelerator):** Web アプリケーションに対してコンテンツ最適化の機能を果たすコンポーネントです。トラフィックマネージャに完全統合されたコンポーネントとしても、スタンドアロンのプロキシモードでも使用でき、後者の場合はトラフィックマネージャの負荷分散機能が無効になります。どちらの形態が適しているかは、お近くの販売代理店でご相談ください。CHAPTER 20 の「Web コンテンツを最適化するための Brocade Web Accelerator の使用」では、インフラストラクチャで Web Accelerator を有効化および設定する詳しい方法について説明しています。

注: Web Accelerator の機能は、SPARC ベースの Solaris または SunOS プラットフォームで稼働する種類のソフトウェアでは利用できません。

トラフィックマネージャのアプライアンスおよびクラウドバージョンには、ソフトウェアのみのバージョンでは利用できない Networking (ネットワーキング) および Date/Time 設定オプションがあります。

利用可能な機能については、ご使用の製品バージョンの仕様に記載されています。該当する『**Installation and Getting Started Guide**』も参照してください。このドキュメントは、Brocade の Web サイトからダウンロードできます。

開発者モード

ライセンスがない場合、トラフィックマネージャは開発者モードと呼ばれるデフォルトの状態になります。これは、開発または評価を目的として、トラフィックマネージャの特徴と機能を全て体験できるように意図された設計です。製品の機能を全て使用できますが、環境の帯域に制約があり、最大でも 1Mb/s および 100 SSL TPS (1 秒あたりのトランザクション数) の帯域でしか動作しません。

重要:開発者モードは、完全に商用として利用することを想定して設計されたものではありません。希望する用途に合った、適切なライセンスキーを購入する方法については、サポート窓口にお問い合わせください。

対応プラットフォーム

トラフィックマネージャソフトウェアは、様々なプラットフォーム、物理または仮想サーバ、およびクラウドインフラストラクチャにデプロイできます。最新のプラットフォームおよびバージョン番号の必要条件については、

<http://www.brocade.com/en/products-services/application-delivery-controllers/virtual-traffic-manager.html> でリリースノートおよびドキュメントを参照してください。

ソフトウェア

トラフィックマネージャにはソフトウェアパッケージがあり、これは既存のサポートされる Linux サーバと UNIX サーバでのデプロイに適しています。対応ディストリビューションのリストは、前述したリリースノートに掲載されています。

アプライアンス

Brocade は、アプライアンスディスクディスクイメージとして、トラフィックマネージャを提供しています。これは、承認されたサーバハードウェアプラットフォームでのデプロイに適しています。

仮想アプライアンス

VMware vSphere、Citrix XenServer、OracleVM、Microsoft Hyper-V、および QEMU/KVM 向けに、トラフィックマネージャ仮想アプライアンスパッケージが用意されています。これらのアプライアンスイメージには、各 Linux OS レイヤとともに事前インストールされるトラフィックマネージャソフトウェアパッケージが含まれています。

クラウドコンピューティングサービス

トラフィックマネージャは、Amazon の Elastic Compute Cloud (EC2)、Rackspace、Microsoft Azure、および Google Compute Engine (GCE) でサポートされています。また、EC2 および GCE 上で実行されているサポート対象の Linux および UNIX 仮想マシンインスタンスでのトラフィックマネージャソフトウェア製品のインストールもサポートされています。

サポートされるクラスタの組み合わせ

前のセクションで述べたように、トラフィックマネージャは選択したプラットフォームに応じて各種の製品形態で利用できます。コアとなるトラフィックマネージャソフトウェアに加え、適切なライセンスをアップグレードすることによって様々な製品オプションと機能が追加されます。

複数のトラフィックマネージャをクラスタ化すると高可用性と冗長性を得ることができ、これによって同じクラスタ内の異なるプラットフォーム上でトラフィックマネージャの柔軟なアーキテクチャを利用してインスタンスをホストできるようになります。ただし、クラスタの各メンバが同じ製品設定で実行されているか確認が必要です。

クラスタを設計する際には、以下の点に留意してください。

- ソフトウェア製品に対して別々のサポートされているホスト OS が混在していてもサポートされます。クラスタの動作に悪影響を及ぼすことはありません。
- クラスタ内でライセンスキーのタイプを混在することは技術的には可能ですが、お奨めしません。クラスタのサブセットでのみ許諾されている機能を使用する場合、管理 UI で警告やエラーが表示される可能性があります。これは、設定の自動複製機能が、ライセンスのない機能をクラスタ全体に適用しようとするためです。
- クラウドインスタンスを、非クラウドインスタンス(ソフトウェア、仮想アプライアンス、ハードウェアアプライアンス)とともに直接クラスタ化することはできません。ただし、マルチサイトクラスタ管理機能を使用することにより、ロケーションに関係なくトラフィックマネージャの複数のクラスタを集中管理することができます。
- Web Application Firewall の機能は、プラットフォーム混在のクラスタ(ソフトウェアと仮想アプライアンスなど)では使用できません。
- クラスタで複数の製品バージョンを併用することはできませんが、クラスタ全体のアップグレード(またはダウングレード)を実行する場合は例外です。このような移行状態の間、クラスタで異なるバージョンが実行されているというメッセージが一部のトラフィックマネージャで表示されます。これは想定される現象であり、クラスタの全てのメンバがアップグレードされれば、このエラーは表示されなくなります。

重要: 推奨されるトラフィックマネージャインフラストラクチャの実行によって生じる影響について不明な点がある場合は、サポート窓口にお問い合わせで指示を受けるようにしてください。

² 詳細については、CHAPTER 28 を参照してください。

各章の概要

CHAPTER 2 ネットワークレイアウト

この章では、代表的なトラフィック管理されたサーバファームの運用に必要なネットワーク構成について説明します。ハードウェアの必要条件、ネットワークのレイアウト、IP アドレス、および DNS エントリについて取り上げます。

CHAPTER 3 初期設定

この章では、仮想サーバ、プール、およびルールの仕組みなど、トラフィックマネージャアーキテクチャのコンセプトを説明します。また、最初のサービスの管理を開始する方法について説明します。

CHAPTER 4 仮想サーバ

仮想サーバは、トラフィックマネージャソフトウェアを構成する 2 つの基本コンセプトのうちの 1 つです。この章では、仮想サーバについて詳しく説明します。

CHAPTER 5 プール

プールは、トラフィックマネージャソフトウェアを構成するもう 1 つの基本コンセプトです。この章では、プールについて詳しく説明します。

CHAPTER 6 トラフィック IP グループとフォールトトレランス

この章では、フォールトトレランス、すなわちバックエンドサーバまたは個々のトラフィックマネージャシステムの耐障害性について説明します。

CHAPTER 7 トラフィックマネージャ管理インターフェースの主要機能

これまでに、基本的なトラフィックマネージャシステムの動作について説明してきました。この章では、トラフィックマネージャの性能モニタリングや診断機能など、トラフィックマネージャ管理インターフェースで提供されるツールについて説明します。

CHAPTER 8 TrafficScript ルール

この章では、トラフィックを管理するためのルールの作成に使用する TrafficScript 言語を紹介します。ルールの作成方法を説明し、例をいくつか掲載します。トラフィックマネージャの TrafficScript 機能については、別冊の『TrafficScript Guide』で詳しく説明しています。

CHAPTER 9 TrafficScript での認証のサポート

この章では、TrafficScript によるリモート認証のサポートについて詳しく説明します。TrafficScript を使用すると、仮想サーバにリモートユーザの検証機能を付加することができます。

CHAPTER 10 Java Extensions

この章では、トラフィックマネージャの Java Extensions について説明します。Java Extensions を使用すると、TrafficScript から Java コードを呼び出すことができます。トラフィックマネージャの Java Extensions 機能については、別冊の『**Java Development Guide**』で詳しく説明しています。

CHAPTER 11 プロトコルのサポート

一部のプロトコルには、プロトコル固有のサポートによりメリットがもたらされます。この章では、トラフィックマネージャ製品の特殊なプロトコルハンドラについて説明します。

CHAPTER 12 セッション維持

この章では、トラフィックマネージャのセッション維持機能について説明します。役立つセッション維持ストラテジを紹介し、本製品での設定方法について説明します。

CHAPTER 13 SSL 暗号化

この章では、受信トラフィックの SSL 復号化、バックエンドサーバに送信されるリクエストの暗号化について取り上げます。また、本製品での SSL 証明書、認証局、および失効リストの設定方法と、サービスへの適用方法について説明します。

CHAPTER 14 ヘルスモニタリング

この章では、トラフィックマネージャのヘルスモニタリング機能について説明します。ヘルスモニタでは、バックエンドサーバノードが正常に動作しているかどうかテストされます。障害が発生すると、トラフィックマネージャからアラートが報告され、障害が発生したノードを経由しないようトラフィックがルーティングされます。

CHAPTER 15 サービスプロテクション

この章では、本製品のサービスプロテクション機能について説明します。カタログにサービスプロテクション設定のクラスを定義する方法と、1つ以上のサービスでそのクラスを使用する方法を説明します。

CHAPTER 16 帯域制御

この章では、本製品の帯域制御機能について説明します。カタログに帯域制御設定のクラスを定義する方法と、1つ以上のサービスでそのクラスを使用する方法を説明します。

CHAPTER 17 リクエストレートシェイピング

この章では、トラフィックマネージャを使用した、サーバおよびアプリケーションに対するリクエストのレートシェイピング方法について説明します。レートシェイピングを行うと、ユーザのアクティビティを制限し、リクエストによりアプリケーションが過負荷状態になるのを防ぐことができます。

CHAPTER 18 サービスレベルモニタリング

この章では、本製品のサービスレベルモニタリングについて説明します。カタログにサービスレベルモニタリング設定のクラスを定義する方法と、1つ以上のサービスでそのクラスを使用する方法を説明します。

CHAPTER 19 コンテンツキャッシング

この章では、トラフィックマネージャの HTTP コンテンツキャッシング機能について説明します。頻繁にリクエストされる Web リソースをキャッシングすることで、以降のリクエストに対して、バックエンドサーバノードに同じリクエストを何度も転送せずに、キャッシュから直接レスポンスすることができます。

CHAPTER 20 Web コンテンツを最適化するための Brocade Web Accelerator の使用

この章では、Brocade Web Accelerator と、Web サービスに Web コンテンツの最適化を適用するための方法について説明します。

CHAPTER 21 イベント処理およびアラート

この章では、特定のイベントが発生した場合のトラフィックマネージャの動作をコントロールする方法と、外部管理システムに対する通知または制御の方法を説明します。

CHAPTER 22 システムレベルの設定

この章では、ハードウェアアプライアンス、仮想アプライアンス、クラウドインスタンス、および一部のトラフィックマネージャソフトウェア製品に固有の補足的なシステムレベル設定について説明します。

CHAPTER 23 システムセキュリティ

この章では、トラフィックマネージャをセキュアに運用するためのヒントを紹介します。ファイアウォールの設定、ネットワーク設計、およびトラフィックマネージャソフトウェアに関連する Unix ソフトウェアのパーミッションについて取り上げます。

CHAPTER 24 管理サーバのセキュリティ

この章では、外部データベースに対するユーザ認証など、管理インターフェースへのアクセスをコントロールするためのセキュリティ対策について説明します。

CHAPTER 25 トラフィックマネージャのコントロール API

この章では、トラフィックマネージャのコントロール API について簡単に紹介します。詳細については、Brocade の Web サイトから入手できる別冊の『*Brocade Virtual Traffic Manager: Control API Guide*』を参照してください。

CHAPTER 26 コマンドラインインターフェース

この章では、『*Brocade Virtual Traffic Manager: Control API Guide*』で定義されているコントロール API メソッドを参照しながら、コマンドラインインターフェースについて詳しく説明します。

CHAPTER 27 *zconf* を使用したきめ細かい設定のインポート/エクスポート

この章では、*zconf* コマンドラインユーティリティについて説明します。このユーティリティを使用すると、個々の設定オブジェクトについてきめ細かいインポート/エクスポートを実行することができます。

CHAPTER 28 マルチサイトクラスタの管理

この章では、トラフィックマネージャのクラスタのコンセプトを応用し、複数のクラスタを、集中管理される 1 つのマルチサイトスーパークラスタにまとめるアイデアを紹介します。

CHAPTER 29 トラフィックマネージャの DNS サーバ

トラフィックマネージャには、外部にソースを持たせる DNS の代替手段として、内部 DNS サーバ機能が用意されています。この章では、トラフィックマネージャの DNS サーバの実装および構成について説明します。

CHAPTER 30 グローバル負荷分散

この章では、トラフィックマネージャへのグローバルサーバ負荷分散(GSLB)の実装について説明します。

CHAPTER 31 トラフィックマネージャの FIPS 認証

この章では、FIPS がトラフィックマネージャにどのように実装されているかを理解しておきたい将来的なユーザのために、その点について詳細に説明します。

CHAPTER 32 Kerberos 強制委任のサポート

この章では、トラフィックマネージャへの Kerberos プロトコルの実装について説明します。

CHAPTER 33 トラブルシューティング

この章では、負荷分散システムでの問題または異常動作を調査し、診断する方法について説明します。

CHAPTER 34 用語集

「用語集」では、本マニュアルで使用されている用語について説明します。用語によっては、コンピューティング関連の文献で様々な意味で使用されているものがありますので、必要に応じてこのセクションを参照してください。

CHAPTER 2 ネットワークレイアウト

本章では、ネットワークの構成について説明します。サーバファームのトラフィックを効果的に管理するために必要なハードウェアと、DNS および IP アドレスレイアウトについて取り上げます。

ネットワーク構成の基本コンポーネント

トラフィック管理される基本的なサーバファームは、以下のコンポーネントで構成されます。

- トラフィックマネージャソフトウェアが稼働する 1 台以上のフロントエンドマシン
- 多数のバックエンドサーバ(Web サーバ、メールサーバなど)

フロントエンドマシンは、インターネット(またはリモートクライアントが配備された場所)からトラフィックを受信できなければなりません。また、バックエンドマシンとも通信できなければなりません。

通常、バックエンドサーバは、内部ネットワークからのみ認識できます。インターネットとバックエンドマシンとの間でやり取りされるトラフィックは、フロントエンドマシンがルーティングする必要はありません。

一般に、トラフィックマネージャソフトウェアは、マルチホームマシンにデプロイします。インターネット側から認識できるのは 1 枚のネットワークインターフェースカードのみです。また、バックエンドサーバがデプロイされている内部プライベートネットワークからは、1 枚以上のネットワークインターフェースカードを認識できます。ネットワークカードが 1 枚のみ搭載されたマシンにトラフィックマネージャを設定することもできます(評価またはテスト環境では一般的な構成)。この場合、トラフィックマネージャはクライアントとサーバの両方と通信することができます。

完全なフォールトトレランスを備えた構成では、2 台以上のフロントエンドマシンと複数台のバックエンドサーバを使用します。この構成では、いずれかのマシンで障害が発生しても、トラフィックマネージャのフェールオーバー機能によってリクエストが他のマシンにルーティングされるため、システム内に単一障害点が発生することがありません。

注: 製品バージョンによっては、クラスタサイズが 2 台のトラフィックマネージャマシンに制限されます。このような製品バージョンでは、ソフトウェアキーをアップグレードすることにより、クラスタを拡張することができます。

利用できるハードウェアが限られている場合、少ない台数のサーバでネットワークを構成することもできます。最小構成として、トラフィック管理ソフトウェアとインターネットサービスを同一マシンにインストールすることも可能です。このような構成では、製品の有用性とハードウェア障害発生時のフォールトトレランス機能が損なわれるためお奨めしませんが、評価またはデモンストレーション用として役立つこともあります。

トラフィックマネージャは、スタンドアロン型のファイアウォールと組み合わせて使用することができます。トラフィックマネージャマシンは、インターネットと内部ネットワークの両方から認識できなければならないため、多くの場合は DMZ 内へのデプロイが適しています。

注: CHAPTER 23 では、ネットワーク環境におけるセキュリティについて詳しく説明しています。商用サービスをセットアップする前に、このセクションをよくお読みください。

専用管理ネットワーク

トラフィックマネージャを設定するときに、単一の管理 IP アドレスを指定することができます。これにより、トラフィックマネージャは、その IP アドレスでのみ管理トラフィックを受け入れるようになります。ご使用の製品に専用の管理 IP アドレスを設定する際に必要な設定手順については、『**Installation and Getting Started Guide**』を参照してください。

トラフィックマネージャに関するドキュメントは、Brocade の Web サイト、<http://www.brocade.com> を参照してください。

管理トラフィックには、Web ベースのトラフィックマネージャ管理 UI、コントロール API、REST API、およびトラフィックマネージャクラスタ内で共有されるあらゆる設定や状態に対する全てのアクセスが含まれます。

管理 IP アドレスを使用することで、ユーザは信頼性の高い専用の管理ネットワークを実現できます。一般に、トラフィックマネージャを運用する各ホストに、管理ネットワークに接続された専用のネットワークカードを搭載します。

注: ネットワーク環境におけるセキュリティの詳細については、373 ページの「システムセキュリティ」を参照してください。商用サービスをセットアップする前にこの章をよくお読みいただくことをお奨めします。

完全に構成されたトラフィックマネージャで管理 IP アドレスを変更するには、管理 UI の **System** (システム) > **Traffic Managers** (トラフィックマネージャ) ページを開きます。ソフトウェア製品の場合は、「**configure**」のスクリプトを再実行することもできます。この場合、ソフトウェアのリスタートが必要になることに注意してください。

重要: 各管理 IP アドレスは、トラフィックマネージャクラスタ内の単一障害点になり得ます。管理ネットワークで障害が発生すると、マシン間の通信が全て失われ、管理 UI、コントロール API、または REST API を使用したリモート設定が行えなくなります。

耐障害性を実現するために、各トラフィックマネージャが送信するフォールトトレランスメッセージは、全てのネットワークカードにブロードキャストされます。管理 UI では、このトラフィックを管理ネットワークに制限することができます。

クラスタのサイジング

バックエンドサーバ

トラフィックマネージャは、無制限の台数のバックエンドサーバをサポートできます。バックエンドサーバには、Web サーバ、メールサーバ、FTP サーバなどを含めることができます。これらのサーバでは、任意のプラットフォームで任意のソフトウェアを実行してサービスを提供でき、そのサービスはトラフィックマネージャによって管理されます。

バックエンドサーバは、それぞれが処理するコンテンツのタイプ別にプールと呼ばれるグループに整理できます。例えば、IISTMが稼働する Windows マシンや Zeus Web Server または ApacheTMが稼働する Unix マシンなどの Web サーバを複数のグループに分け、グループごとに異なるタイプの静的および動的 Web コンテンツを処理することができます。トラフィックマネージャのプールおよびルール of の仕組みを利用すれば、リクエストを分類して複数のサーバグループに振り分けることができます。プールとルールについては、54 ページの「初期設定」で説明します。

特定機能を実行するために専用使用するサーバの台数は、状況によって異なります。例えば、静的 Web ページを処理する場合、リクエストの負荷が小さければ、2 台の Linux マシンで十分なこともあります。一方、より複雑な SunSM JSPTM や Microsoft ASP ページを処理する場合は、より多くのマシンが必要になります。

トラフィックマネージャでは、任意の台数のマシンを使用できます。また、必要条件が変化したときは、サービスを中断することなく、バックエンドサーバを動的に追加または削除することができます。

バックエンドサーバでのコンテンツ管理

多数のバックエンドサーバで同一サイトの処理を行う場合、サイトのコンテンツが変更されたときに、それを各バックエンドサーバに伝搬する必要があります。これは、以下のいずれかの方法で行うことができます。

- 各バックエンドサーバにコンテンツのローカルコピーを保存し、データを定期的に同期する。

- NFS ファイルサーバなどの共有ファイルシステムを使用し、ここにコンテンツを保存して、各バックエンドサーバが必要に応じてデータを読み込む。

静的 Web コンテンツを処理する場合、どちらのコンテンツ管理方法も有効です。SMTP または POP3 サーバの場合は、サービスによってファイルが変更されるため、共有ファイルシステムを使用する必要があります。

フロントエンドサーバ

トラフィックマネージャの単純なデプロイでは、1 台のフロントエンドマシンを使用します。この場合、フロントエンドマシンのフォールトトレランス機能を除き、トラフィックマネージャソフトウェアの全機能を利用できます。すなわち、トラフィックマネージャマシンが単一障害点になる可能性があります。

完全なフォールトトレランスを備えた構成には、2 台以上のトラフィックマネージャが必要です。これらのトラフィックマネージャの IP アドレスを 1 つのトラフィック IP グループにまとめることで、いずれかのマシンで障害が発生した場合にフォールトトレランスを提供することができます。トラフィック IP グループについては、CHAPTER 6 で説明します。

一般的なネットワークの構成では、フロントエンドマシンを、インターネットとの通信用のネットワークとバックエンドとの通信用のネットワーク(通常はプライベートネットワーク)の 2 つのネットワークに 1 つずつデプロイします。ただし、フロントエンドマシンとバックエンドマシンを同一ネットワークにデプロイすることも可能です。インターネットからルーティング可能な全てのマシンには、適切にファイアウォールを設定してください。ファイアウォールの設定およびネットワークセキュリティについては、CHAPTER 23 を参照してください。

スループットを高めるために、トラフィックマネージャにネットワークカードを追加することもできます。必ずしも必要ではありませんが、ネットワークカードを 2 枚使用した場合、2 つの独立したネットワークにトラフィックマネージャを簡単にデプロイできるというメリットがあります。

トラフィックマネージャは、非常に優れた拡張性を備えています。サーバを増設したり、既存のハードウェアをアップグレードしたりすると、それに応じてトラフィック管理の性能も向上します。後からトラフィックマネージャマシンを増設するときも簡単で、既存の構成に手を加えたりサービスを中断したりする必要がありません。

注: ソフトウェアバージョンのトラフィックマネージャ製品ファミリを使用している場合、他のアプリケーションおよびサービスによってホストマシンが過負荷状態にならないよう注意してください。例えば、Web ブラウザやその他の GUI ツールを同一サーバで実行すると、サーバ上のトラフィックを管理するソフトウェアの処理能力が低下します。

下図は、トラフィック管理される一般的なサーバームを示しています。フロントエンドマシンには、前述のようにフロントエンド用とバックエンド用に個別のネットワークカードが使用されています。

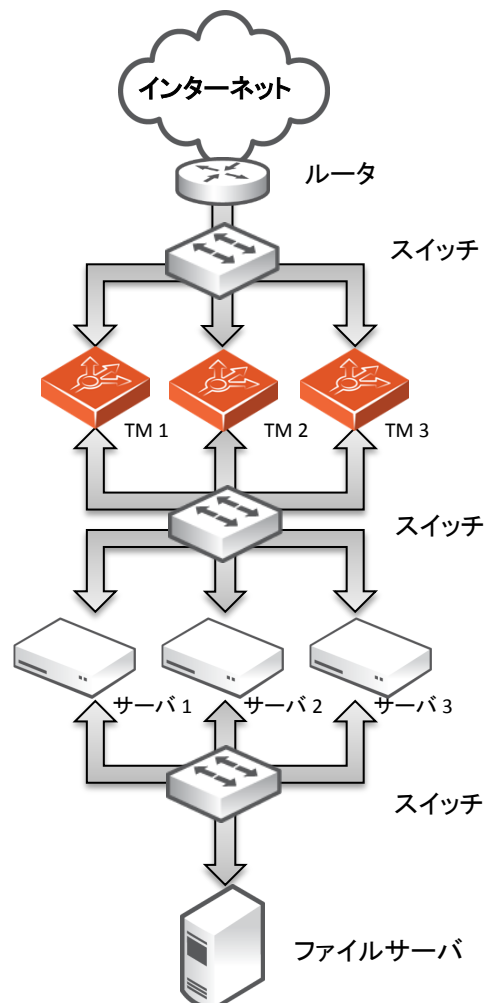


Fig 2. バックエンドノードを処理するトラフィックマネージャユニットのクラスタによってフォールトトレランスを実現するネットワーク

バックエンドサーバが処理するコンテンツは、NFS ファイルサーバに保持されます。各バックエンドサーバでは、スループットを最大限に引き上げるために、トラフィックマネージャマシンとの通信用と、このファイルサーバとの通信用に 2 枚のネットワークカードを使用します。

IP トランスペアレンシ

トラフィックマネージャは完全なアプリケーションプロキシとして機能し、リモートクライアントからのネットワークコネクションをいったん終端し、選択されたバックエンドサーバへのコネクションを新たに確立します。単純なレイヤ 4 ロードバ

ランサで採用されているパケット指向型の NAT ベースの負荷分散手法は使用しません。

このアーキテクチャでは、バックエンドサーバはクライアントのリクエストを、リモートクライアントではなくトラフィックマネージャから出されたものとして認識します。このことは、バックエンドサーバがクライアント IP アドレスに基づいてアクセスコントロールを行ったり、リモート IP アドレスを記憶しようとする場合に、問題になる可能性があります。この状況は、通常、アクセスコントロールまたはロギング機能をトラフィックマネージャ自体で実行するか、トラフィックマネージャがクライアントの IP アドレスを識別するために全ての HTTP コネクションに挿入する X-Cluster-Client-Ip ヘッダを利用することにより対処できます。

これらの対処方法が適切でない場合、トラフィックマネージャは、サーバ側コネクションの送信元 IP アドレスをスプーフィングし、リクエストがクライアントのリモート IP アドレスから出されたかのように見せかけることができます。この機能を IP トランスペアレンシと言います。

IP トランスペアレンシは選択的に使用することができます。例えば、トラフィックマネージャが Web ファームおよびメールファームへのトラフィックの負荷分散を行っている場合、SMTP トラフィックに「IP トランスペアレンシ」を適用し、Web トラフィックには適用しないようにすることが可能です。

トランスペアレンシは、プール単位で有効化します。すなわち、前述の例では、Web プールにはトランスペアレンシを適用せず、SMTP プールにはトランスペアレンシを適用するよう設定します。この場合、Web プールおよび SMTP プールでは、トラフィックを同じバックエンドノードに負荷分散することも、異なるノードに負荷分散することもできます。

デフォルトでは、トラフィックマネージャのアプライアンスイメージ、仮想アプライアンス、またはクラウドサービスの全てのバージョンで、IP トランスペアレンシを利用できます。トラフィックマネージャソフトウェア製品では、Linux ホストまたは UNIX ホストで、以下の条件でネイティブ IP トランスペアレンシ機能を使用できます。

- トラフィックマネージャソフトウェアがインストールされ、root ユーザとして実行されている。
- ホストオペレーティングシステムで、バージョン 2.6.24 以降のカーネルが使用されている。
- ホストオペレーティングシステムで、バージョン 1.4.11 以降の iptables が使用されている(--transparent オプションを使用可能な場合は、バージョン 1.4.11 よりも前のバージョンの iptables もサポートされます)。

カーネルバージョン 2.6.18～2.6.24 では、追加の専用カーネルモジュールを通じて IP トランスペアレンシのサポートが提供されます。詳細、および必要なソフトウェア

のダウンロードについては、Brocade Community の Web サイト (<http://community.brocade.com>)にある vADC ドキュメントを参照してください。

注: ダウンロード可能な Brocade IP トランスペアレンシモジュールは、バージョン 2.6.18 よりも前または 3.2 よりも後の Linux カーネルではサポートされていません。

Brocade IP トランスペアレンシモジュールの現行ユーザが、バージョン 2.6.24 以降にカーネルをアップグレードした場合、Brocade IP トランスペアレンシモジュールがネイティブ Linux IP トランスペアレンシのサポートよりも優先されて、アップグレードプロセス中の動作の変化が最小限に抑えられます。ただし、ネイティブの Linux 実装によるパフォーマンス向上が得られるよう、IP トランスペアレンシモジュールを計画的にアンインストールすることを強くお奨めします。

注: Solaris/SunOS ソフトウェア製品では IP トランスペアレンシを利用できません。

ルーティング設定

IP トランスペアレンシが適用されたトラフィックを受信した各サーバノードは、そのトラフィックの送信元であるトラフィックマネージャを経由してレスポンスをルートバックする必要があります。これを行うには、そのノードのデフォルトルートをトラフィックマネージャのいずれかのバックエンド IP アドレスに設定します。デフォルトルートの設定の詳細については、OS のドキュメントを参照してください。

サーバノードは、リクエストに対するレスポンスを返すときに、リモートクライアントの IP にアドレス指定します。ルーティングが正しく設定されていれば、送信元のトラフィックマネージャはこのトラフィックを傍受し、コネクションを終端して通常どおり処理します。トラフィックマネージャは、クライアントコネクションを通して(場合によっては変更された)レスポンスをクライアントに転送します。

通常は、トラフィックマネージャシステムにアドレス指定されていないその他全てのパケットについても、トラフィックマネージャによって単純に転送されるよう設定します。これにより、トラフィックマネージャシステムはローカルルータとして機能し、トラフィックマネージャをデフォルトルートとして使用するサーバが近隣またはリモートネットワーク上の他のシステムと通信することができます。トラフィックマネージャでは、プライベートネットワーク上のバックエンドノードから転送する全てのパケットを NAT 変換する必要があります。

注: ご使用の種類のトラフィックマネージャソフトウェアが稼働するホストシステムでこの「IP 転送」機能を設定する場合は、OS のドキュメントを参照してください。トラフィックマネージャの仮想アプライアンスまたはクラウドインスタンスでこの動作を設定するには、CHAPTER 22 を参照してください。

ローカルルーティングの制限事項

IP トランスペアレンシを使用する場合、バックエンドノードと同一ネットワーク上にあるクライアントは、トラフィックマネージャを通してこれらのノードにトラフィックを負荷分散することができません。

これは、バックエンドサーバノードが常にコネクションの送信元 IP アドレスにレスポンスを返そうとするためです。送信元 IP アドレス(クライアントの IP)がローカルネットワーク上に存在する場合、サーバノードは、接続元であるトラフィックマネージャシステムを経由してルーティングせずに、クライアントと直接通信しようとするため、コネクションがハングアップしたかのように見えます。

この場合、バックエンドネットワークをセグメント化し、サーバノード側から見てクライアントが別のサブネット上に存在し、デフォルトゲートウェイ(トラフィックマネージャシステム)を経由しなければアクセスできないかのように見せかけることで対処することがあります。または、TrafficScript ルールにより、クライアントがサーバノードと同一ネットワーク上に存在する場合に、サーバコネクションの送信元 IP アドレスをトラフィックマネージャシステム上の IP に選択的に設定することも可能です。

IP トランスペアレンシとトラフィックマネージャクラスタ

各サーバノードは、1つの IP アドレスを経由してレスポンスをルートバックしなければならないため、トラフィックマネージャのクラスタを使用する場合は、ルーティングが複雑になります。

このような状況での推奨事項については、下の「トラフィック IP アドレスとトラフィック IP グループ」を参照してください。

トラフィック IP アドレスとトラフィック IP グループ

一般的なネットワークでは、バックエンドサーバは、ローカル IP アドレスが割り当てられてローカルネットワーク内に配置され、インターネットからは直接アクセスできません。一方、フロントエンドのトラフィックマネージャマシンは、外部からアクセス可能な IP アドレスが割り当てられ、ローカルネットワークを介してバックエンドマシンに接続することができます。

フロントエンドマシンには、各ネットワークで固定 IP アドレスが割り当てられ、フロントエンドアドレスはインターネットから認識、ルーティングできます。これらの IP アドレスは OS によって設定され、マシンで障害が発生すると、その IP アドレスは失われます。

このような理由から、これらの固定 IP アドレスは、サービスを公開する場合には適していません。ハードウェアやシステムに障害が発生した場合に、サービスの一部または全部が利用できなくなるためです。

トラフィックマネージャのフォールトトレランス機能を利用すると、トラフィック IP アドレスを設定することができます。トラフィック IP アドレスは特定のマシンに紐付けられないため、トラフィックマネージャクラスタでは、クラスタ化された一部のトラフィックマネージャマシンで障害が発生しても、各 IP アドレスは完全に利用可能な状態に保たれます。

フォールトトレランスを備えた一般的な構成では、サービスを公開するために使用する DNS 名がトラフィックマネージャクラスタ内のトラフィック IP アドレスに名前解決されるよう設定します。これにより、サービスを完全に利用可能な状態に維持できます。

トラフィック IP はトラフィック IP グループに整理できます。このグループは、一部または全てのトラフィックマネージャマシンにまたがることができます。これらのマシン間でトラフィック IP アドレスを共有するための判別が行われ、各トラフィックマネージャでは、割り当てられた IP アドレス (1 つまたは複数) が有効になります。

トラフィック IP グループの設定については、CHAPTER 6 で説明します。

いずれかのトラフィックマネージャマシンで障害が発生すると、グループ内の他のトラフィックマネージャがこれを検出し、いずれかのトラフィックマネージャが、障害が発生したマシンのトラフィック IP アドレスを引き継いで、サービスが中断しないようにします。

注: デフォルトでは、フォールトトレランス機能ではユニキャストトラフィックによってヘルス状態が配信され、マルチホストトラフィック IP アドレスへのトラフィックが管理されます。マルチキャストトラフィックによる配信に設定を変更する場合、トラフィックマネージャを接続するスイッチは、IGMP スヌーピング、およびクラスタ内の全てのトラフィックマネージャに転送されなければならないトラフィックマネージャによって使用されるマルチキャストアドレスにブロードキャストされるメッセージをサポートする必要があります。

トラフィックマネージャを既存のクラスタに組み込むと、様々なテストが自動的に実行され、ブロードキャストメッセージが正常に転送されるかどうかを確認されます。

トラフィック IP アドレスのモード

トラフィックマネージャは、以下の複数のモードをサポートしています。

- **シングルホスト:** トラフィック IP アドレスは、フォールトトレランスクラスタ内の単一のトラフィックマネージャで有効になります。このトラフィックマネージャで障害が発生した場合、別のトラフィックマネージャがその IP アドレスを引き継ぎ、トラフィックを受け入れ始めます。
- **マルチホスト:** トラフィック IP アドレスは、クラスタ内の全てのトラフィックマネージャで有効になり、マルチキャスト MAC アドレスを使用することで、受信

トラフィックが全てのマシンに送信されます。トラフィックは、カスタム Linux カーネルモジュールにより、稼働しているトラフィックマネージャ間で均等に分散されます。

- **ルートヘルスインジェクション:** トラフィック IP アドレスは、加えられている全てのトラフィックマネージャによって「**private** に」(ループバックで)有効になり、ルーティングプロトコルとして **Open Shortest Path First** バージョン 2 (OSPFv2) または **Border Gateway Protocol (BGP)** のいずれかを使用して、隣接したルーティングドメインに動的にアドバタイズされます。レスポンスでは、ルータはアクティブなトラフィックマネージャにトラフィックを送信します。46 ページの「ルートヘルスインジェクション」を参照してください。

マルチホストトラフィック IP アドレスを有効化すると、クラスタ内のトラフィックマネージャに要求されるパケット処理量が増加するため、性能が低下します。テストによる実測では、中程度のトラフィックレベル(毎秒 10,000 リクエスト)で CPU 使用率が 25~30% 増加し、それに従って最大処理能力も制限されるという結果が示されています。

注: マルチホスト IP 機能は、トラフィックマネージャのハードウェアアプライアンスと仮想アプライアンスの全てのバージョンで利用できます。トラフィックマネージャソフトウェア製品にはデフォルトでは含まれていませんが、追加カーネルモジュールとしてダウンロードしてインストールすることができます。この機能は Linux カーネルのバージョン 2.6.18 以上でサポートされます。現在、Solaris/SunOS ソフトウェア製品ではマルチホスト IP 機能を利用できません。

サポートされているバージョンについての詳細は、Brocade が提供するコミュニティ Web サイト(<http://community.brocade.com>)を参照してください。

構成例

以下の構成は、クラスタに 2 台のトラフィックマネージャがあることを前提としていますが、これらの構成を、3 台以上のトラフィックマネージャから成るクラスタに拡張することもできます。

アクティブ-パッシブ構成 – シングルホストモードとルートヘルスインジェクションモード

Web サイトの外部 DNS 名が IP アドレス 162.34.64.29 にマッピングされ、2 台のトラフィックマネージャマシンによって多数のバックエンド Web サーバへのトラフィックが処理されているとします。

- シングルホストモードでは、上記の単一の IP アドレスを使用する、両方のトラフィックマネージャマシンにまたがるシングルホストトラフィック IP グループを設定できます。2 台のトラフィックマネージャ間でネゴシエーションが行われた後、いずれかのトラフィックマネージャでこの IP アドレスが有効になり、全ての受信リクエストが処理されます。もう一方のトラフィックマネージャマシン

はスタンバイ状態で待機します。最初のマシンで障害が発生した場合、2 台目のマシンが IP アドレスを引き継ぎ、トラフィックの管理を開始します。

- ルートヘルスインジェクション(RHI)モードでは、上記の単一の IP アドレスを使用する、両方のトラフィックマネージャマシンにまたがる RHI トラフィック IP グループを設定できます。一方のトラフィックマネージャをアクティブに、もう一方のトラフィックマネージャをパッシブに設定します。両方のトラフィックマネージャが IP アドレスをアドバタイズし、(選択したルーティングプロトコルに応じて)OSPFv2 や BGP メトリクスを使用して優先順を示します(アクティブなトラフィックマネージャが低い方のメトリクスを使用します)。アップストリームのルータ(一般的にデフォルトゲートウェイ)は、メトリクスが最も低いルータを選択して全てのトラフィックをアクティブなトラフィックマネージャに送信します。トラフィックマネージャは、障害を検出するとアドバタイズメントをキャンセルします。ルータは、障害を検出するとそのルートが無視します。いずれの場合も、ルータは 2 番目に最適なアドバタイズメントメトリクスに従った送信トラフィックに切り換わります(この場合はパッシブなトラフィックマネージャ)。

この構成のメリットは、いずれかのトラフィックマネージャで障害が発生しても、トラフィックを処理する十分なリソースを確保できることです。また、トラフィックを処理しているトラフィックマネージャが 1 台のみのため、デバッグや障害解決が容易です。

アクティブ-アクティブ構成 – シングルホストモードとマルチホストモード

アクティブ-アクティブ構成では、両方のトラフィックマネージャがトラフィックを管理します。2 台のトラフィックマネージャ間でトラフィックをどのように分配するかは、分散モード(シングルホスト IP またはマルチホスト IP)によってコントロールします。

シングルホストモードでは、トラフィック IP グループに 2 つのトラフィック IP アドレスを設定し、DNS 名を 162.34.64.29 と 162.34.64.30 など 2 つのアドレスにマッピングするよう設定できます。2 台のトラフィックマネージャ間でネゴシエーションが行われた後、各トラフィックマネージャで 1 つずつトラフィック IP アドレスが有効になります。DNS サーバは、各 IP アドレスに交互にリクエストを割り当てることができ(ラウンドロビン DNS)、トラフィックマネージャはそれぞれが受信したリクエストを処理します。

いずれかのマシンで障害が発生すると、もう一方のマシンがそれを検出します。このマシンは、元々使用していたトラフィック IP アドレスに加えて、障害が発生したマシンのトラフィック IP アドレスを引き継ぎ、両方のアドレスに送信された全てのトラフィックを処理します。

マルチホストモードでは、1 つのトラフィック IP アドレスで継続的に運用できるため、DNS を単純化し、外部に公開する必要のある IP アドレスの数を減らすことができます。トラフィック IP アドレスは、トラフィック IP グループに含まれる全て

のトラフィックマネージャで有効になり、この IP アドレス宛ての受信トラフィックはトラフィックマネージャ間で均等に分配されます。

ループバック付きアクティブ-アクティブ構成(単一の外部アドレス、シングルホストモード)

マルチホストモードが使用できない、またはインフラで禁止されている場合でも、単一の外部トラフィック IP アドレスを使用しながら、トラフィック負荷をクラスタ内の全てのトラフィックマネージャに分散することができます。

この構成では、追加の「ループバック」仮想サーバが外部トラフィック IP アドレスでトラフィックを監視し、トラフィックマネージャの固定 IP アドレスを使用してクラスタ内のトラフィックマネージャにリクエストを負荷分散します。

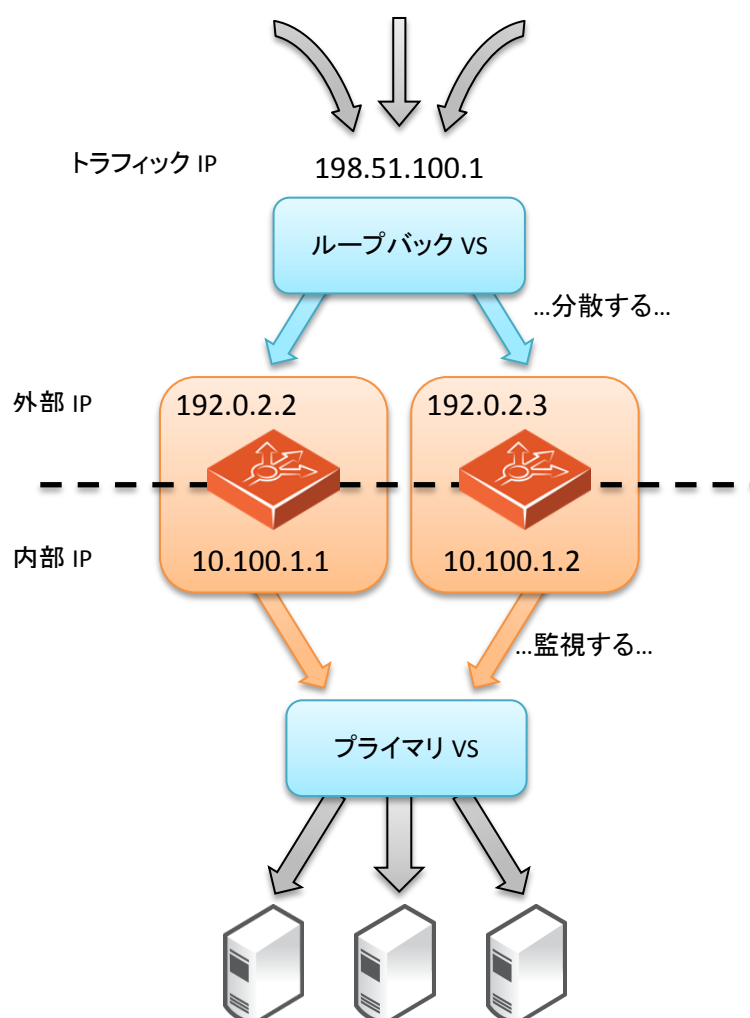


Fig 3. 「ループバック」仮想サーバは 1 つのトラフィックマネージャにホストされている単一のトラフィック IP を取得し、他の IP に分散します。

まず、トラフィックの処理とバックエンドサーバへの負荷分散を行うプライマリ仮想サーバを作成します。この仮想サーバは、外部からアクセス可能な IP アドレスで

はなく、各トラフィックマネージャの内部 IP アドレスとポートでトラフィックを監視するように設定します。例えば、仮想サーバが、内部からのみ認識可能な IP アドレスである 192.0.2.1:80 と 192.0.2.2:80 でトラフィックを監視するようにします。

次に、セカンダリとして「ループバック」仮想サーバを作成します。この仮想サーバでは、単一の外部 IP アドレスでトラフィックを監視し、クラスタ内の様々なトラフィックマネージャマシンにまたがるプライマリ仮想サーバにトラフィックをただちに送信します。

- アクティブ-パッシブ構成の例と同様に、いずれかのトラフィックマネージャでのみ有効になる単一のトラフィック IP アドレスを 1 つ設定します。このアドレスで受信する全てのトラフィックは、そのトラフィック IP アドレスで監視を行っている単純なループバック仮想サーバで処理する必要があります。ループバック仮想サーバは、クラスタ内のトラフィックマネージャマシンの内部 IP アドレスが含まれるループバックプールを即座に選択する必要があります。ループバックプールでは、ラウンドロビンまたは最小接続負荷分散によって、クラスタ内のトラフィックマネージャマシンにトラフィックを均等に分配する必要があります。レスポンス時間の影響を受ける負荷分散手法ではリクエストの分配が非常に不均等になるため、使用しないでください。

ループバック仮想サーバが使用する処理能力はわずかです。SSL 復号化、ルール、コンテンツ圧縮など CPU 負荷の大きい処理は全てプライマリ仮想サーバで実行するようにしてください。

この方法では、負荷の大きいタスクの負荷が 2 台のトラフィックマネージャに分割されます。いずれかのトラフィックマネージャで障害が発生しても、サービスは継続されます(一部トラフィックで瞬間的に中断する可能性もあります)。例えば、外部トラフィック IP アドレスをホストしているトラフィックマネージャで障害が発生した場合、そのトラフィック IP アドレスがもう一方のトラフィックマネージャに引き渡されます。また、ループバックプールでは、一方のノードが使用不可になっていることを検出し、残りのトラフィックマネージャで稼働しているプライマリ仮想サーバに全てのトラフィックを送信します。

ターゲット仮想サーバでは、リモートクライアントではなく、ループバック仮想サーバからの接続を監視します。一般に、X-Cluster-Client-Ip ヘッダから接続元を正しく特定できますが、通常、SSL リクエストはループバック仮想サーバによって転送されるため、CHAPTER 13 の「SSL 転送における IP アドレスの維持」セクションで説明する **ssl_enhance** および **ssl_trust_magic** 設定を使用する必要があります。

多重冗長(N+M)構成

前述の例では 2 台のトラフィックマネージャマシンを使用していますが、いずれかのマシンで障害が発生した場合、システムに単一障害点が発生する可能性があります。ミッションクリティカルなサービスを実行している場合は、複数のトラフィッ

クマネージャマシンを追加して多重冗長クラスタを構成することで、さらに強力な保護レベルを実装できます。

通常の運用で、 N 台のアクティブなトラフィックマネージャを使用する場合、 M 台のパッシブなトラフィックマネージャを組み込みます。 M は、障害が発生してもアクティブなトラフィックマネージャの台数を減らさずに済む台数です。

これを実現するには、トラフィックマネージャが稼働する $N+M$ 台のフロントエンドマシンが必要です。 N 個のトラフィック IP アドレスから成り、しかも $N+M$ 台のマシン全てで共有されるトラフィック IP グループを作成することができます。アクティブなグループのマシンで障害が発生した場合には、パッシブグループのバックアップマシンがアクティブになって負荷を引き継ぎます。さらに別のマシンで障害が発生すると、次のパッシブトラフィックマネージャがアクティブになり、というように続きます。これは、複数レイヤの冗長性を組み込んだ一般的なクラスタ編成です。

クラスタでの IP トランスペアレンシの使用

トラフィックマネージャマシンのクラスタでは、IP トランスペアレンシの使用が複雑になります。これは、各サーバノードが単一のトラフィックマネージャ IP アドレスにルーティングするよう設定される一方、サーバノードに対してクラスタ内のどのトラフィックマネージャマシンもトランスペアレントにコネクションを確立する可能性があり、ノードは、その接続元のトラフィックマネージャを経由して各レスポンスをルートバックしなければならないからです。

アクティブ-パッシブ構成

アクティブなトラフィックマネージャが 1 つのみの構成では、これを、シングルホスト IP アドレスを使用するトラフィック IP グループに **keeptogether** を設定することにより実現できます。

受信トラフィック用のフロントエンド IP アドレスとサーバ側ネットワークに存在するバックエンド IP アドレスの 2 つの IP アドレスで構成されるトラフィック IP グループを作成します。トラフィック IP グループで **keeptogether** オプションを選択します。

トラフィック IP グループに設定したバックエンド IP アドレスを経由して全てのトラフィックをルーティングするよう、各バックエンドサーバを設定します。

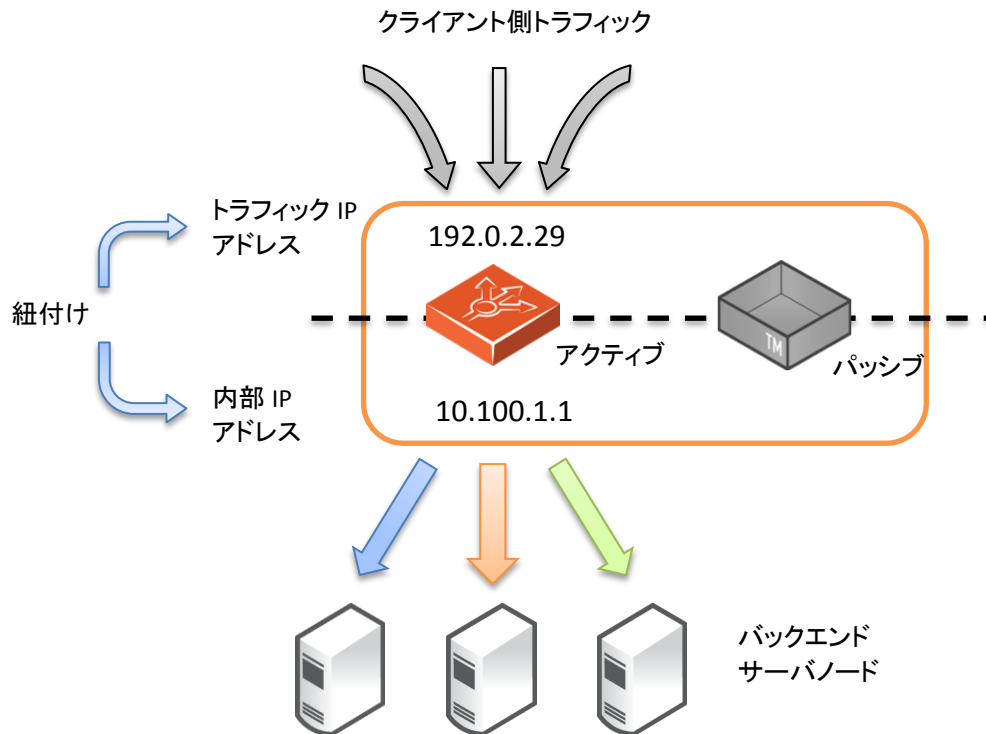


Fig 4. トラフィック IP と内部 IP は、トラフィック IP グループにまとめられます。

この設定により、両方の IP アドレスが同一のトラフィックマネージャマシンでホストされるようになります。そのマシンで障害が発生した場合、両方の IP アドレスが同じパッシブマシンに移行され、このマシンが新しいアクティブマシンになります。以降、バックエンドサーバは、新しいアクティブマシンを経由してトラフィックをルートバックします。

アクティブ-アクティブ構成

複数のアクティブなトラフィックマネージャを伴う構成では、バックエンドサーバをグループに分割し、各グループにアクティブなトラフィックマネージャマシンを1台ずつ割り当てる必要があります。次に、これらのグループをトラフィックマネージャ管理 UI 内のプールとして定義し、それぞれにバックエンドサーバをノードとして追加します。

同じプールのサーバは全て、そのデフォルトルートに対応するトラフィックマネージャのバックエンド IP アドレスに設定する必要があります。サーバのルート設定の操作方法については、ご使用の OS のドキュメントを参照してください。

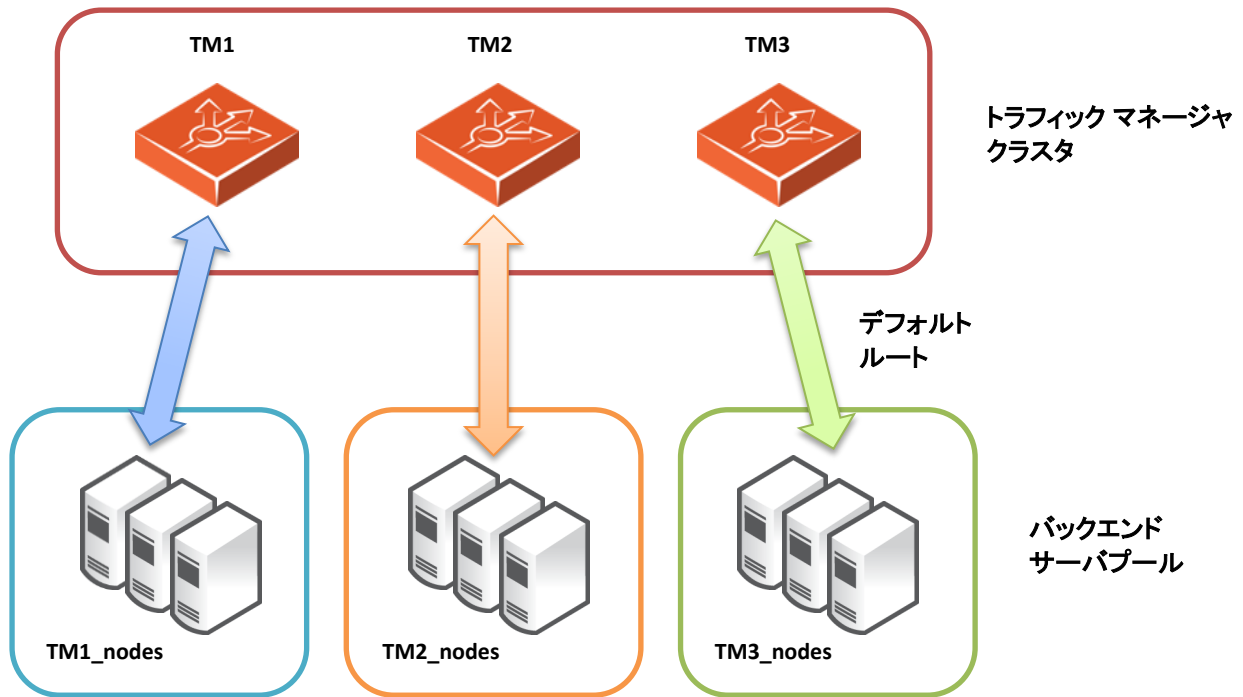


Fig 5. サーバをトラフィックマネージャ固有のプールに分割します。

次に、TrafficScript ルールを使用して、以下のいずれかの条件に基づいてトラフィックのルーティング先プールを正しく選択することができます。

- コネクションを管理しているトラフィックマネージャの名前
- ローカルクライアント側 IP アドレス(シングルホストモードで複数の「keep-together」トラフィック IP グループを使用する場合)

以下のコードの断片に、この動作の実例を示します(トラフィックマネージャ名を選択条件に使用します)。

```
$hostname = sys.hostname();

if( $hostname == "TM1" ) {
    pool.use( "TM1_nodes" );
}
if( $hostname == "TM2" ) {
    pool.use( "TM2_nodes" );
}
if( $hostname == "TM3" ) {
    pool.use( "TM3_nodes" );
}
```

重要: この設定には、トラフィックマネージャで障害が発生すると対応するプールが冗長になるという制限があります。また、セッション維持を使用するときの信頼性が下がります(特にマルチホスト IP アドレスを使用する場合)。

IP トランスペアレンシは選択的に使用することができます。例えば、トラフィックマネージャクラスタが **www.mysite.com** への大容量 Web トラフィックと、**mail.mysite.com** への小容量 SMTP トラフィックを管理しているとします。この場合、トランスペアレンシの適用が必要になるのは SMTP のみです。この場合、以下のようになります。

- IP トランスペアレンシを適用しないアクティブ-アクティブ TrafficCluster 構成では、**www.mysite.com** を複数の IP アドレスに名前解決できます。
- 前述のアクティブ-パッシブ **keepttogether** 構成では、**mail.mysite.com** を単一の IP アドレスに名前解決できます。

ルートヘルスインジェクションおよびネットワーク

ルートヘルスインジェクション(RHI)を使用する場合、トラフィックマネージャはルータとの通信を隣接したルーティングドメインで行います。通信が確立されると、トラフィックマネージャはルーティングドメインに加わり、そのドメインに RHI トラフィック IP アドレスをアドバタイズします。

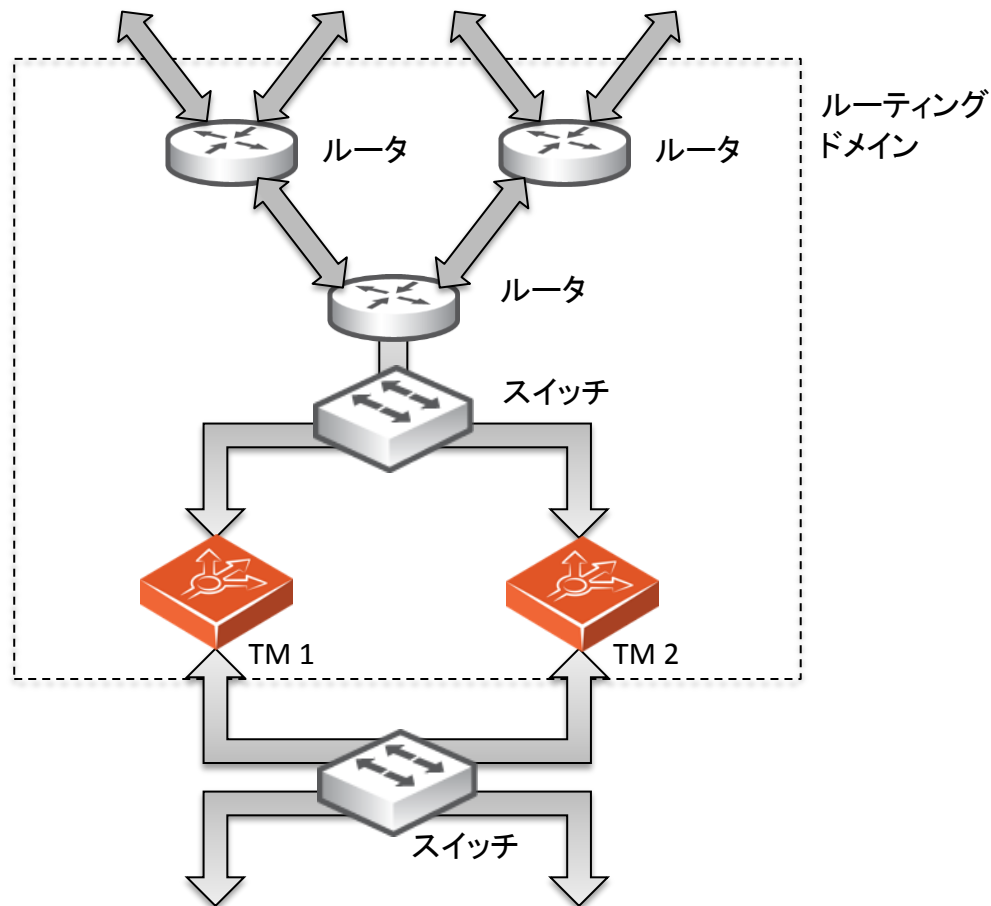


Fig 6. RHI を利用したトラフィック IP アドレスの通知

このようなアドバタイズメントは動的に行われ、トラフィックマネージャの構成に対して RHI トラフィック IP アドレスを作成、廃止または移動するような変更を加えた場合でも、通知は自動的にそれらの変更に応答を返します。さらに、トラフィックマネージャが検出した障害にもアドバタイズメントは自動的にレスポンスを返すほか、ルーティングドメインの動的なルーティングプロトコルを介して、トラフィックマネージャに固有ではないネットワーク障害にも自動的にレスポンスを返します。

これにより、幅広い種類のデプロイにおいて RHI を利用することが可能になります。例えば、ローカル接続のシンプルなフェールオーバーを管理するだけの小規模なデプロイ(単独で存在するデータセンタのラックの中など)はもちろん、企業内の複数のデータセンタ間やインターネット全体にわたってトラフィックの分配やフェールオーバーを管理する必要があるような、大規模なデプロイでも利用することができます。

RHI は、RHI 専用の IPv4 トラフィック IP グループを使用して機能します。単一のロケーションでは、単一のアクティブなトラフィックマネージャ、またはアクティブ-パッシブのペアになったトラフィックマネージャのいずれかによって提供される RHI トラフィック IP グループを使用することができます(39 ページの「アクティブ-パッシブ構成 – シングルホストモードとルートヘルスインジェクションモード」を参照してください)。

注: RHI は、IPv6 アドレスに基づくトラフィック IP グループをサポートしていません。

スケールを拡大するには、トラフィックマネージャのデータセンタのロケーションにおいて、このパターンを必要なだけ繰り返します。各グループのトラフィック IP アドレスが同一である場合、ロケーションごとに異なる RHI トラフィック IP グループが使用されていますが、通常、使用されている OSPFv2 または BGP メトリクスが異なります。

注: OSPFv2 の場合、このシナリオでは、全てのデータセンタが同じルーティングドメインにあることが要求されます。BGP の場合、データセンタは、ルーティングドメインの内部または外部に配置することができます。

各ロケーションには、異なる優先順位を設定することができます。例えば、プライマリ/スタンバイのロケーションが 2 点存在するデータセンタのデプロイでは、以下のように構成します。

1. プライマリデータセンタでは、アクティブ-パッシブのペアのトラフィックマネージャから提供されるプライマリ RHI トラフィック IP グループを構成します。
2. スタンバイデータセンタでは、アクティブ-パッシブのペアのトラフィックマネージャから提供されるスタンバイ RHI トラフィック IP グループを構成します。
3. プライマリデータセンタが利用可能である限りプライマリデータセンタが常に優先されるように、極端に高いメトリクスを用いてスタンバイ RHI トラフィック IP グループを構成します。

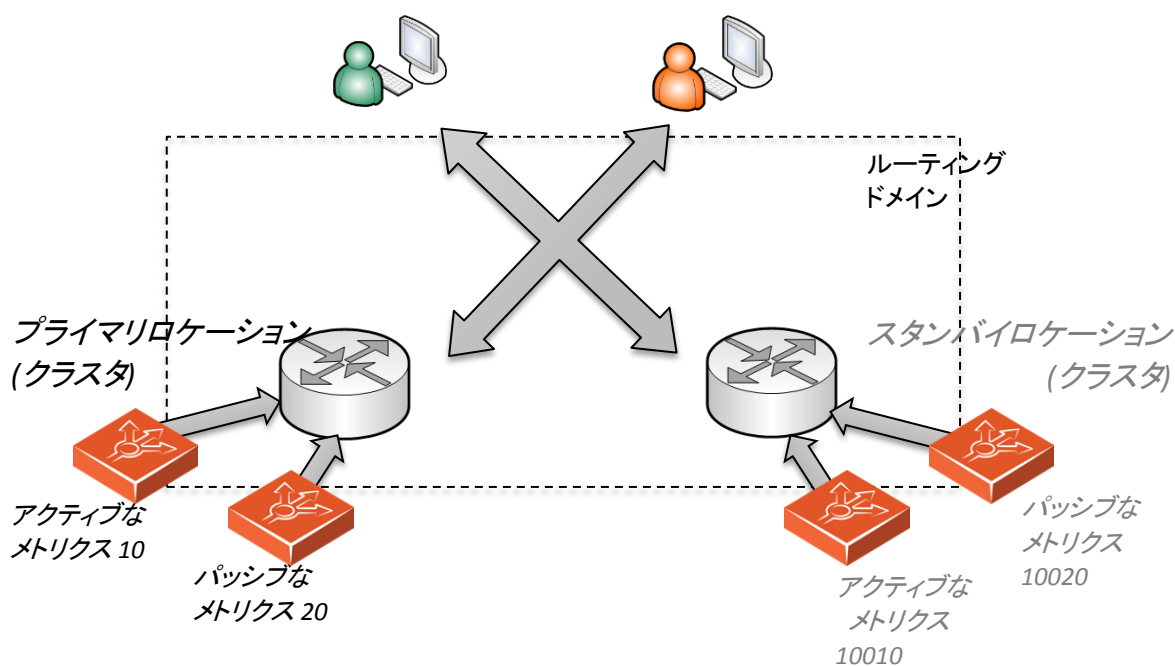


Fig 7. RHI を利用した 2 箇所のデータセンタでのデプロイ

別の方法として、データセンタのロケーションに同等の優先順位を設定することもできます。その場合は、アクティブなロケーションが複数存在することになり、ネットワークトポロジに応じた最適なルートに基づいてルーティング経路が決定されます。一般的に、この方法はエニーキャスト構成と呼ばれています。各ロケーションの RHI トラフィック IP グループを同じメトリクスを用いて設定することによって、この構成を実現することができます。

注: 複数のロケーションにわたる BGP を使用して実装された RHI では、RHI をサポートするためにインフラストラクチャの他の部分を設定する必要があります。言い換えれば、データセンタのロケーション間のルータが、提供されたメトリクスに従い、ロケーションの選択に影響を及ぼす可能性がある他のポリシーが設定されないようにする必要があります。

RHI 通信で使用されるクレデンシアル

各トラフィックマネージャは、OSPFv2 を使用するか、または BGP とのセッションを確立して、隣接ルータと通信することにより RHI を実行しています。トラフィックマネージャが、OSPFv2 または BGP に対応した隣接ルータとの通信を確立するためには、適切なクレデンシアルを用いてトラフィックマネージャを設定する必要があります。この手法は、ルータとのピアリングと呼ばれています。

注: OSPFv2 通信にはマルチキャストが必要です。

同じクレデンシアルを使用するトラフィックマネージャのクラスタを作成することができます。各クラスタ(またはマルチサイトマネージャクラスタ内の各ロケーション設定、418 ページの「マルチサイトクラスタの管理」を参照)は 1 セットのクレデンシアルを使用するため、クラスタに含まれる全てのトラフィックマネージャ(またはロケーション設定)が同一のエリア(OSPFv2 の場合)または自律システム(AS)(BGP の場合)に加わります。

別のクレデンシアルを必要とする複数のロケーションがある場合は、異なるロケーションのトラフィックマネージャをクラスタ化する必要はありません。ただし、それらのトラフィックマネージャをクラスタ化したい場合には、トラフィックマネージャのマルチサイトマネージャ機能を使用する必要があります。

OSPFv2 または BGP のクレデンシアルを入力するには、**System (システム) > Fault Tolerance (フォールトトレランス)** ページを使用します。

OSPFv2/BGP の設定および RHI トラフィック IP グループの設定の詳細については、111 ページの「トラフィック IP グループとフォールトトレランス」を参照してください。

RHI に関する問題のトラブルシューティング

トラフィックマネージャは、サードパーティのルーティングソフトウェアを使用して RHI を運用しています。このルーティングソフトウェアは、ご使用のファイルシ

システムの\$ZEUSHOME/log/routing_swにある特定ファイルに RHI イベントログを記録します。

このファイルの内容はテクニカルサポートレポートに含まれており(495 ページの「サポートを利用するには」を参照)、サポート窓口はこの情報を活用して、ご使用のトラフィックマネージャのデプロイで生じた RHI 通信問題のトラブルシューティングを行います。

このログファイルには、以下のローテーションポリシーが適用されます。

- 仮想アプライアンスおよびクラウドインスタンスの場合、トラフィックマネージャは固定のファイルサイズ(デフォルトは 50 MB)を使用して、ログローテーションを自動的に実施します。アーカイブされたログは「gzip」圧縮法で圧縮され、日付と連番を含むファイル名で保存されます。
- ソフトウェアインスタンスの場合は、独自のローテーションポリシーを制定する必要があります。ログファイルがローテーション(移動)されていることをトラフィックマネージャに伝えるために、
\$ZEUSHOME/zxtm/zebos/bin/reload_logfile ポストローテーションのスク립トを使用して、新規ファイルでロギングプロセスを再開させます。

OSPFv2 と BGP について

トラフィックマネージャは、RHI のルーティングプロトコルとして、Open Shortest Path First バージョン 2 (OSPFv2)と Border Gateway Protocol (BGP)をサポートしています。

OSPFv2 はインテリアゲートウェイプロトコルの 1 つで、通常、例えば ISP や大企業のネットワークなどにおいて、あるネットワーク上の単一の自律システム(AS)内でルートを分散するために使用されます。OSPFv2 の働きにより、ルータは同一 AS 内にある OSPFv2 が設定された他のルータを自動検出し、同期することができます。

OSPFv2 は、存続時間(TTL)値に 1 を使用し、TCP や UDP を介してではなく生の IP パケットを使用してレイヤ 3 で動作します。OSPFv2 は、マルチキャストアドレス指定を使用して大量のルーティング情報をチェーン内の次のルータに送信すると同時に、OSPFv2 自体のエラー検出およびエラー補正に対処することができます。通常、OSPFv2 は ISP などの組織内部に存在するため、速やかに収束させることができます(あるルートが変更された場合、ネットワークの全てのルータが新しい最短ルートに合意した時点で収束が実現します)。

OSPF の詳細については、http://en.wikipedia.org/wiki/Open_Shortest_Path_First を参照してください。

一方、BGP は一般に、個々の AS の外部でルーティングと到達可能性に関する情報を配信するために使用される外部ゲートウェイプロトコルです。

注: BGP は、内部ゲートウェイプロトコルとしても使用できます。そのためには、トラフィックマネージャで情報交換のために **Internal BGP (iBGP と呼ばれる)** が使用されます。異なる AS のルータ間の通信では、**External BGP (eBGP と呼ばれる)** が使用されます。

OSPFv2 とは異なり、BGP 対応ルータでは、ピアが自動検出されず、セッションが確立されるネイバーの明示的な設定を定義する必要があります。

BGP の詳細については、www.bgp4.as を参照してください。

上記のプロトコルのいずれかまたは両方を使用して、IP アドレスを通知し、ネットワークを通じてこの通知を伝搬させるようにトラフィックマネージャを設定できます。サービスでルートヘルスインジェクションを有効化し、使用するには、113 ページの「トラフィック IP グループの作成」を参照してください。

IPv6 について

IPv6 は、パケット交換ネットワークで使用されるネットワークレイヤプロトコルです。IPv6 の最大の特徴は利用可能なアドレスが莫大なことです。これは、IPv4 のアドレスのビット長が 32 ビットなのに対し、IPv6 では 128 ビットになっているためです。また、ネットワーク管理が単純化され、複雑なサブネットスキームを使用する必要がないというメリットもあります。

IPv6 のメリットを以下に挙げます。

- IP セキュリティ
- モバイル IP アドレス
- 単純なヘッダ構造
- アドレスの自動設定
- エニーキャスト(1 つのアドレスを多数のインターフェースで共有)と必須マルチキャストアドレス

トラフィックマネージャでの IPv6 の主な機能

- IPv6 のゲートウェイとして機能
- フロントエンドサーバおよびバックエンドサーバで複数のバージョンの IP アドレスを処理可能
- IPv4 と IPv6 の混在ネットワークでも、IPv4 のみのネットワークでも動作可能

IPv4 アドレスが使用可能であれば、トラフィックマネージャを設定するときに、トラフィック IP アドレス(RHI およびマルチホストトラフィック IP アドレスを除く)、仮想サーバが監視するノードまたはアドレスの指定、TrafficScript ルールなどで、IPv6 ユニキャストアドレスを使用できます。

トラフィックマネージャは、IPv4 から IPv6 または IPv6 から IPv4、あるいは同時にその両方のゲートウェイとして機能することもできます。

技術的制約

重要: トラフィックマネージャ環境で IPv6 を使用する場合、いくつかの制約があります。これらの制約がソフトウェアの通常の動作に影響を与えることはありませんが、常に留意しておいてください。

IPv6 の使用に関する主な制約は以下のとおりです。

- トラフィックマネージャ間の内部通信は IPv4 プロトコルで行われます。
- ハートビートメッセージは IPv4 でのみ機能します。管理サーバも同様です。
- バックエンドノードの設定でホスト名を使用する場合、トラフィックマネージャはまず IPv4 アドレスを検索します。DNS が IPv4 と IPv6 の両方のアドレスを持っている場合にマシンで IPv6 アドレスを使用するには、IPv6 アドレスを直接入力する必要があります。
- ホストの DNS に IPv6 アドレスのみが割り当てられている場合は、そのアドレスが使用されます。

重複アドレス検出の調整

注: このセクションは、トラフィックマネージャソフトウェア製品のみを対象としています。トラフィックマネージャのハードウェアアプライアンス、仮想アプライアンス、およびクラウド製品では、自動的に重複アドレス検出が調整され、設定されます。

多くの OS の重複アドレス検出(DAD)機能は、2 台のマシンで同時に同じアドレスが有効にならないようにすることを目的としています。この機能は、トラフィックマネージャのフォールトトレランス機能と競合する可能性があります。IP がトラフィックマネージャシステムから別のトラフィックマネージャシステムに引き渡されるときに、タイミングによっては、そのアドレスを引き継ぐトラフィックマネージャに対して DAD が実行されることがあります。DAD 機能は以下のように調整することができます。

- **Linux:** `sysctl` を呼び出して `net.ipv6.conf.default.dad_transmits` および `net.ipv6.conf.all.dad_transmits` の値を確認します。`/etc/sysctl.conf` ファイルで、これらの行を以下のように追加(または変更)します。

```
net.ipv6.conf.default.dad_transmits = 0
```

および:

```
net.ipv6.conf.all.dad_transmits = 0
```

この変更をすぐに有効にするには、関連する各インターフェース(この例では **eth1**)と、「**default**」および「**all**」に対して以下のコマンドを使用します。

```
# sysctl -w net.ipv6.conf.eth1.dad_transmits=0
# sysctl -w net.ipv6.conf.default.dad_transmits=0
# sysctl -w net.ipv6.conf.all.dad_transmits=0
```

- **Solaris 10:** システムでサポートされている場合は、以下の設定の値を変更します。

```
# ndd -set /dev/arp arp_probe_count 0
# ndd -set /dev/ip ip_dup_recovery 50
```

CHAPTER 3 初期設定

本章では、トラフィック管理される基本的なサイトをセットアップする方法について説明します。

アーキテクチャコンセプト

トラフィックマネージャは、Web サーバやアプリケーションサーバ(HTTP、HTTPS)のようなネットワークサービス、Web サービス(SOAP)、メール(SMTP、POP、IMAP)、および DNS やストリーミングメディアなどその他プロトコルのトラフィックを管理します。

サービスは、IP アドレスおよびポートとして公開され、適切なプロトコルを使用してトラフィックを受け入れます。

Web サイトは `www.mysite.com` でホストされます。この DNS 名は IP アドレス `123.123.12.1` に名前解決され、サーバはこのアドレスのポート 80 で HTTP トラフィックを監視します。

インターネットサービスプロバイダ(ISP)は、POP3 および SMTP メールサーバを `pop.mymail.com` および `smtp.mymail.com` として公開します。これらの各サーバの DNS エントリは以下の IP アドレスにリンクされています。

`pop.mymail.com` のアドレスは `202.3.45.67`

`smtp.mymail.com` のアドレスは `202.3.45.68`

`pop.mymail.com` はポート 110 で POP3 トラフィックを監視し、`smtp.mymail.com` はポート 25 で SMTP トラフィックを監視します。.

トラフィックマネージャ内では、特定のサービスに対する全てのトラフィックが仮想サーバで処理されます。仮想サーバは、トラフィックマネージャとインターネット間のインターフェースとして機能します。指定のポートとプロトコルが設定され、通常、そのプロトコルのトラフィックを全て管理します。

仮想サーバは、リクエストを受信すると、それをプールに割り当てます。プールはノードの集合であり、各ノードは `server1.mysite.com:80` のように特定のバックエンドサーバとポートに対応しています。トラフィックはプールによってノードに負荷分散されます。複数のプールを設定することができ、同じノードを複数のプールに割り当てることも可能です。

仮想サーバは、リクエストに対して使用するプールを決定するために、*ルール*のリストを適用できます。ルールでは、受信したリクエストを調べ、それに対して実行するアクションを決定します。ルールによって、リクエストを処理するプールを選択したり、リクエストを終了したり、またはリクエストをリスト内の次のルールに渡したりすることも可能です。どのルールでも明確なルーティングが決定されなかったリクエストは、仮想サーバのデフォルトプールに割り当てられます。

プール内のノードで障害が発生した場合、トラフィックマネージャのモニタがこれを自動的に検出し、そのノードへのリクエストの送信を停止します。トラフィックは、プール内の他のノードに分配されるため、サービスで目に見えるような中断は起こりません。

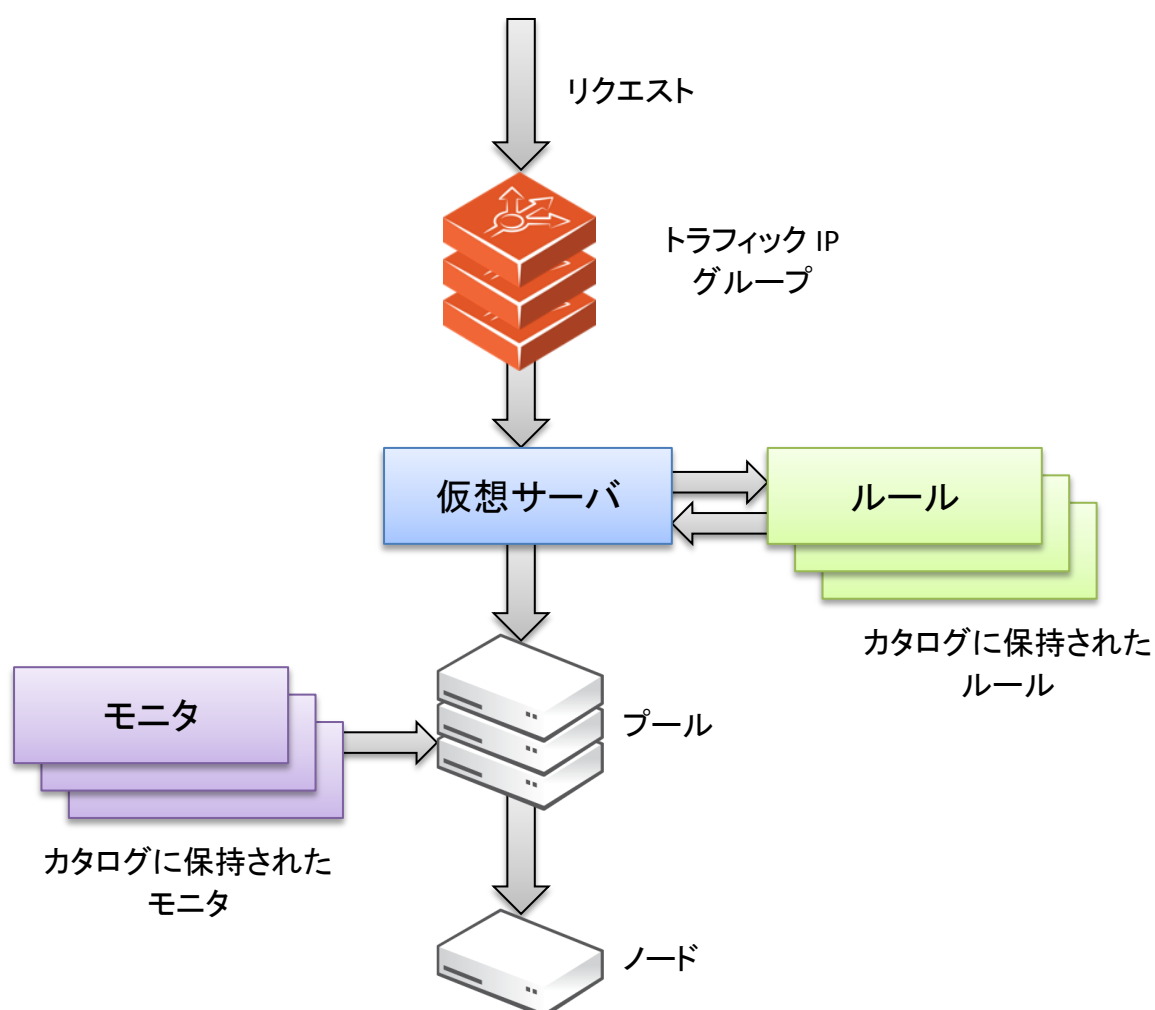


Fig 8. トラフィックマネージャの基本的な構成

トラフィックマネージャマシンでは、管理するサービスごとに1つずつ、多数の仮想サーバを実行することができます。各仮想サーバは複数のルールとプールを使用でき、様々なモニタによってプールを監視できます。

ルールとモニタは、*カタログ*として区分されます。例えば、ルールカタログには、そのシステム用に作成された全てのルールが保持されます。カタログ内のルールは、

任意の仮想サーバに簡単に適用することができます。SSL 証明書やサービスプロテクションクラスなどのその他の共有項目もカタログに保持されます。

複数のトラフィックマネージャマシンをクラスタとして扱うことができます。クラスタ内の全てのトラフィックマネージャマシンでは設定が共有され、各マシンで実行される仮想サーバ、使用されるルールなどは同じです。

サービスで使用するパブリックトラフィック IP アドレスは、トラフィック IP グループの中に割り当てることができます。1つのトラフィック IP グループは、クラスタ内の一部または全てのトラフィックマネージャにまたがり、これらのトラフィックマネージャ間でそのグループの IP アドレスがホストされます。特定のトラフィックマネージャマシンで障害が発生した場合、同じトラフィック IP グループに含まれる他のマシンのいずれかがその IP アドレスを引き継ぎます。この動作とプールのフェールオーバーシステムにより、フロントエンドとバックエンドの両方に完全なフォールトトレランスが実現されます。

トラフィックマネージャの実行中、ホスト名は、通常 5 分以下の間隔で定期的に名前解決されます。これにより、DNS が変更された場合も、ノードを効率よく名前解決することができます。

最初のサービスの管理

最初のサービスの管理を開始するには、仮想サーバとデフォルトプールを作成する必要があります。これは、**Manage a New Service** (新規サービスを管理)ウィザードまたは **Configure** (設定)ページから行えます。

管理サーバのホームページのアドレスにアクセスし、ユーザ名とパスワードを使用してログインします。

注: configure スクリプト/**Initial Configuration** (初期設定)ウィザードの実行時にライセンスキーを指定しなかった場合、**System** (システム) > **Licenses** (ライセンス)ページでライセンスキーをインストールすることができます。ライセンスキーが指定されていない場合、トラフィックマネージャは開発者モードで動作します。

ウィザードを使用した仮想サーバとプールの作成

「Wizards (ウィザード)」ドロップダウンメニューをクリックして、**Manage a New Service** (新規サービスを管理)ウィザードを選択します。表示される指示に従います。

1. 仮想サーバを識別するための名前を指定します。仮想サーバのポートとポート(例: HTTP、ポート 80)を選択します。

Manage a new Service, step 2 of 4

2. Specify the service

Please enter a brief name to identify the service you would like to balance.

Name:

Please select the protocol that the service uses.

Protocol:

Please specify the port that the protocol listens on.

Port:

Cancel < Back Next >

Fig 9. サーバを定義するための3つの基本パラメータ: 名前、プロトコル、ポート番号

2. バックエンドノードのリストを作成します。これらのノードによって、仮想サーバのデフォルトプールが形成されます。ノードは、ホスト名とポートで識別されます。これらのパラメータは、後で **Pools (プール) > Edit (編集)** ページから変更できます(CHAPTER 5を参照)。指定するホスト名とポート番号の組み合わせによってコンテンツを直接提供できるようにしてください。

Manage a new Service, step 3 of 4

3. Specify the back-end nodes

Please enter the hostname and port of each node:

Hostname: Port:

Nodes:

- server1.mysite.com:80
- server2.mysite.com:80
- server3.mysite.com:80

To remove a node from the list, select it and press 'Remove node':

Cancel < Back Next >

Fig 10. サーバのノード作成

注: 評価のために、トラフィックマネージャを Web サーバ(またはその他のテストサーバ)と同じマシンで実行している場合、2つのサービスが異なるポートでトラフィックを監視するようにしてください。

例えば、Web サーバへのトラフィックを管理しているとします。デフォルト HTTP ポートは 80 です。Web サーバがポート 8080 で、またトラフィックマネージャがポート 80 でトラフィックを監視するよう設定します。この場合、ブラウ

ザにマシン名(例: `server1.mysite.com`)を入力すると、トラフィックはトラフィックマネージャ経由で送信されます。

- 最後に、選択した設定を見直してから、**Finish** (完了)をクリックします。
- ここで、新しいサービスに設定したマシンとポートをブラウザして、トラフィックマネージャの設定をテストすることができます。

注: 仮想サーバ、プール、エラーファイルなどの名前の先頭文字にはピリオド(.)、アンダースコア(_)、またはシャープ(#)を使用できません。また、最後の文字にはチルダ(~)を使用できません。

手動でのプールと仮想サーバの作成

管理インターフェースのトップバーにある **Services** (サービス) ボタンをクリックすると、**Services** (サービス) セクションに移動します。ここでは、仮想サーバ、プール、およびトラフィックマネージャを管理することができます。

プールの作成

- まず、バックエンドサーバのプールを作成する必要があります。**Pools** (プール) タブをクリックし、**Create a New Pool** (新規プールの作成) セクションに詳細を入力します。

プールの名前を選択し、ノードのリストを入力する必要があります(オートスケールリングを有効にする場合、ノードは不要)。各ノードは、`server1.mysite.com:80` の形式で指定する必要があります。ここで、80 は `server1` がトラフィックを監視しているポートです。ノードはスペースで区切ってください。

```
s1.mysite.com:80 s2.mysite.com:8080 s3.mysite.com:80
```

- Create Pool** (プールの作成) ボタンをクリックすると、新しいプールの **Pools** (プール) > **Edit** (編集) ページに移動します。このページでは、基本設定や詳細設定を編集できます。

仮想サーバの作成

- ここで、**Virtual Servers** (仮想サーバ) タブをクリックして、新しい仮想サーバを作成します。名前、プロトコル、および仮想サーバがトラフィックを監視するポートを指定する必要があります。トラフィックマネージャとサーバソフトウェアを同じマシンにインストールする場合は、前のセクションのポート番号に関する注記に留意してください。

仮想サーバのデフォルトプールを選択する必要があります。ルールによって別のプールまたはアクションが指定されない場合、リクエストはこのプールに割り当てられます。

2. **Create Virtual Server** (仮想サーバを作成) ボタンをクリックすると、新しい仮想サーバの **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページに移動します。このページでは、基本設定や詳細設定を編集することができます。これには、仮想サーバがトラフィックを監視する IP アドレス(ドメイン名またはアドレスで指定)も含まれます。

クラスタの作成

一般に、トラフィックマネージャは、フォールトトレランス(CHAPTER 6 を参照)および管理上の理由から、複数のインスタンスから成るクラスタとしてデプロイします。同じクラスタ内の全てのトラフィックマネージャは、同じ設定を共有し、単一のエンティティとして管理されます。

注: ただし、ネットワーク設定などのシステム固有の設定はクラスタメンバごとに固有のまま保たれ、個別に管理する必要があります。

複数のトラフィックマネージャシステムをまとめてクラスタを形成するには、以下のいずれかの方法を用います。

- 新たにトラフィックマネージャクラスタを作成する場合、いずれかのトラフィックマネージャを最初のクラスタメンバとして選択します。次に、他の各トラフィックマネージャの管理 UI に順々にログインし、**Join a cluster** (クラスタに組み込む)ウィザードを使用して、最初のトラフィックマネージャに組み込みます。
- 既存のトラフィックマネージャクラスタに組み込む場合は、新しい各インスタンスの管理 UI にログインし、**Join a cluster** (クラスタに組み込む)ウィザードを使用して、そのインスタンスを既存のクラスタに組み込みます。

注: クラスタ内では、全てのトラフィックマネージャが同等に扱われます。任意のトラフィックマネージャの管理 UI にアクセスでき、設定を変更すると、変更内容が自動的にクラスタ全体に反映されます。全てのトラフィックマネージャが連携することで、フォールトトレランスと簡潔な管理が実現されます。

クラスタへの組み込み

いずれかのトラフィックマネージャの管理 UI にログインし、ツールバーの「Wizards:(ウィザード:)」ドロップダウンボックスから **Join a cluster** (クラスタに組み込む)ウィザードを選択します。

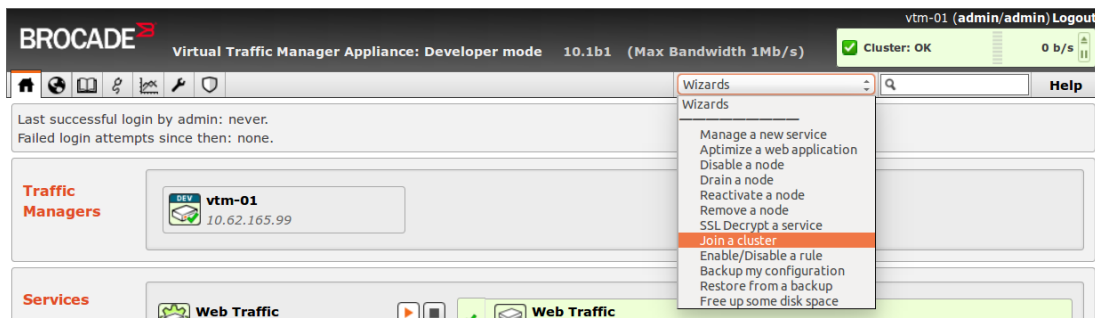


Fig 11. ウィザードによるクラスタの作成

Cluster Joining (クラスタに組み込む)ウィザードは、別のポップアップウィンドウで開かれます。

Cluster Joining wizard, step 1 of 5

1. Getting Started

This wizard joins your current traffic manager to an existing cluster so that it can share the cluster's configuration and traffic.

Would you like to select an existing cluster from a list of available clusters on your network, or enter the Administration Server address and port of a specific traffic manager to join?

☒ Select existing cluster
☐ Manually specify host/port

Cancel ◀ Back Next ▶

Fig 12. Cluster Joining (クラスタに組み込む)ウィザードのステップ1

ネットワークで通信可能なトラフィックマネージャを自動的にスキャンするようにトラフィックマネージャに指示するには、「select existing cluster (既存のクラスタの選択)」をクリックします。または、組み込む特定のホスト名とポートを入力するには、「Manually specify host/port (ホスト/ポートの任意指定)」をクリックします。**Next** (次へ)をクリックして操作を続けます。

次のステップには、ステップ1の選択が反映されます。「select existing cluster (既存のクラスタの選択)」をクリックした場合は、検出されたトラフィックマネージャインスタンスおよびクラスタのリストが表示されます。

Cluster Joining wizard, step 2 of 5

2. Cluster selection

Please select the cluster you wish to join:

- ☐ Cluster 1 athos.cam.zeus.com:9090
- ☐ Cluster 2 atropos.cam.zeus.com:9090
- ☐ Cluster 3 charon.cam.zeus.com:9090
- ☐ Cluster 4 hera.cam.zeus.com:9090
- ☐ Cluster 5 pyramid.cam.zeus.com:9090
- ☐ Cluster 6 sabus.cam.zeus.com:9090
- ☐ Cluster 7 stimp. cam.zeus.com:9090

Cancel < Back Next >

Fig 13. 組み込む既存のトラフィックマネージャクラスタの選択

「Manually specify host/port (ホスト/ポートの任意指定)」をクリックした場合は、表示されるボックスにホスト名とポート番号を入力します。

Cluster Joining wizard, step 2 of 5

2. Cluster selection

Please provide the admin server host and port of one of the machines in the cluster you wish to join:

Hostname:

Port:

Cancel < Back Next >

Fig 14. 組み込む特定のホスト名とポート番号の入力

Next (次へ)をクリックして操作を続けます。

指定したインスタンスまたはクラスタに接続するには、クラスタ内のトラフィックマネージャの ID を検証し、クラスタによって使用される管理クレデンシャルを提供する必要があります。

Cluster Joining wizard, step 3 of 5

3. Authentication

The admin server your are clustering with is using an SSL certificate with the following SHA-1 fingerprint:

tstace-04:9090 ☐ **C1:C3:95:52:60:BD:05:20:7A:CF
4A:1A:9E:46:DE:27:2E:3F:22:60**

▶ **Unfold to view full certificate details ...**

Please check the box beside the fingerprint above to indicate that you have verified it or that you trust the network between it and this system.

If you do not already have this fingerprint on record you can get it by logging into the target admin server and visiting the **System > Security** page. (Refer to the product documentation for further information on cluster security.)

Enter the username and password of a user in the target cluster with permission to add and remove traffic managers.

Username:

Password:

Fig 15. 選択したクラスタの認証

組み込むクラスタで各トラフィックマネージャの ID を検証することをお奨めします。トラフィックマネージャの ID を検証するには、表示された SHA-1 フィンガープリントを、対象トラフィックマネージャの管理 UI で **System (システム) > Security (セキュリティ)** に表示されるフィンガープリントと照合してチェックします。

各トラフィックマネージャの横にあるチェックボックスを選択して、その ID を信頼することを確認し、クラスタの管理ユーザ名とパスワードを入力します。**Next (次へ)**をクリックして操作を続けます。

設定を確認し、**Finish (完了)**をクリックしてクラスタに組み込みます。トラフィックマネージャソフトウェアが自動的に再設定され、新しいホームページが表示されて、接続された全てのトラフィックマネージャインスタンスがトラフィックマネージャリストに示されます。

さらにトラフィックマネージャをクラスタに追加するには、追加する各トラフィックマネージャの管理 UI で **Join a cluster (クラスタに組み込む)** ウィザードを実行します。

トラフィックマネージャを組み込もうとしている既存のクラスタがウィザードのリストに表示されない場合は、ネットワーク設定とケーブル接続を確認し、ネットワークでマルチキャストおよびブロードキャストパケットが許可されていることを確認してください。

トラフィック IP グループが設定されたクラスタへの組み込み

クラスタに既に 1 つ以上のトラフィック IP グループが設定されている場合、ウィザードでは、新しいトラフィックマネージャをこれらのトラフィック IP グループに追加し、ただちにトラフィックの処理を開始できるようにすることができます。

ただし、この場合、トラフィックの割り当てを新しいトラフィックマネージャに引き渡す必要があるため、新しいトラフィックマネージャをトラフィック IP グループに追加する瞬間に、多数のコネクションが失われる可能性があります。このような状況に備えて、新しいトラフィックマネージャを、トラフィック IP グループの「パッシブ」メンバとして追加することができます。こうすることで、グループ内の別のメンバで障害が発生するまで、そのトラフィックマネージャがトラフィックを受け入れることはありません。

CHAPTER 4 仮想サーバ

仮想サーバは、指定されたポートとプロトコルの全てのトラフィックを管理します。リクエストを処理するプールを決定するためにルールを適用したり、必要に応じて SSL トラフィックを復号化したりすることもできます。

トラフィックマネージャは、幅広いプロトコルをサポートしています。HTTP、RTSP、SIP など複雑なプロトコル用に専用のハンドラも用意されており、ソフトウェアは、「一般的な」バージョンのプロトコルを使用して TCP コネクションや UDP セッションのトラフィックを負荷分散することができます。プロトコルのサポートの詳細については、CHAPTER 11 で説明します。

仮想サーバの設定を変更するには、**Services** (サービス) ボタンをクリックし、次に **Virtual Servers** (仮想サーバ) タブをクリックします。仮想サーバのリストが表示されます。ここでは、新しい仮想サーバを作成できます。また、既存の仮想サーバの名前をクリックすると、その仮想サーバの **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページにアクセスできます。

仮想サーバの数について

一般に、実行するサービスごとに 1 つの仮想サーバを実行するよう計画してください。すなわち、トラフィックを受け入れる TCP ポートまたは UDP ポートごとに実行するよう計画します。

例えば、ポート 80 で実行している Web サイトの数には関係なく、HTTP に対して 1 つの仮想サーバを実行します。これは、Web サイトごとに仮想サーバを 1 つずつ設定する Web サーバの設定方法とは異なります。Web サイトごとに必要なトラフィック管理設定がわずかに異なる場合は、TrafficScript を使用してこれらの設定を実装し、トラフィックの管理方法をコントロールすることができます。

一般的なトラフィックマネージャ環境では、5 または 10 以上の仮想サーバを設定することはほとんどありません。仮想サーバ数が 100 を超えることが見込まれる場合は、Web ベースの管理サーバではなく、コントロール API (CHAPTER 25 の「トラフィックマネージャのコントロール API」を参照) または REST API (『*Brocade Virtual Traffic Manager: REST API Guide*』を参照) を使用してプログラムによってトラフィックマネージャシステムを管理するよう計画してください。

プロトコル

仮想サーバを設定するときに、その仮想サーバが処理するトラフィックのプロトコルを指定する必要があります。指定したプロトコルの値は内部プロトコルとして、トラフィックマネージャ内でトラフィックを解析および解釈するために使用されます。管理するトラフィックに最適なプロトコル設定を選択してください。

「Generic Server First」、 「Generic Client First」、および「Generic Streaming」プロトコルは、TCP コネクションを処理する単純な TCP プロトコルハンドラで、クラ

クライアントとサーバ間でデータを転送します。この3つのプロトコルの違いは以下のとおりです。

- トラフィックマネージャは「Generic Server First」仮想サーバでコネクションを受け取ると、ただちに TrafficScript™ ルールを実行し、負荷分散に関する決定を行ってバックエンドサーバに接続します。その後、クライアントとサーバとの間でデータを双方向に渡します。

このプロトコルは、クライアントアプリケーションの接続時に、クライアントがデータをサーバに書き込む前にサーバアプリケーションが「handshake」または「banner」メッセージを書き込むプロトコルに適しています。

- トラフィックマネージャは「Generic Client First」仮想サーバでコネクションを受け取ると、クライアントから何らかのデータを受信するまでコネクションを処理しません。その後、トラフィックマネージャは TrafficScript™ ルールを実行して、負荷分散に関する決定を行い、選択されたバックエンドサーバに接続し、クライアントから受信したデータを書き込みます。以降、トラフィックマネージャはクライアントとサーバとの間でデータを双方向に渡します。

このプロトコルは、サーバにおいて、クライアントがまずデータを書き込んでダイアログを開始することが想定されているプロトコルに適しています。

- 「Generic Streaming」プロトコルは「Generic Server First」に似ていますが、クライアントまたはバックエンドサーバから送信されたリクエスト/レスポンス以外のデータを仮想サーバが処理することが想定されている場合に適しています。トラフィックマネージャは、関連付けられた TrafficScript™ ルールを毎回実行して、負荷分散に関する決定を行い、その後クライアントとサーバとの間でデータを双方向に渡します。

このプロトコルは、サーバまたはクライアントがダイアログを開始でき、受信したデータに対するレスポンスを自動的に想定しないプロトコルに適しています。

これらの一般プロトコルは、他の TCP プロトコルハンドラのベースとなります。一般プロトコルでは、リクエストおよびレスポンスデータをチェックして書き換え、クライアントとサーバ間の通信を同期することができますが、これらの処理は非常に低いレベルに留まります。

ソフトウェアで提供されるその他の TCP プロトコルハンドラは、FTP、HTTP など特定のプロトコル用に特化されています。HTTP などの特化されたプロトコルを選択すると、以下のようなメリットが得られます。

- 仮想サーバおよびプールの設定において、HTTP の keepalive 設定や FTP のセキュリティ設定など、より多くのプロトコル固有のオプションを利用できるようになります。
- トラフィックマネージャによってそのプロトコルのメッセージが解析されるため、例えば HTTP トランザクションでクッキーを読み取って設定するヘルパー関数な

ど、追加の **TrafficScript** 機能を使用してリクエストおよびレスポンスデータを容易に管理できるようになります。

HTTP など高レベルプロトコルを処理する場合は、適切な基本の「**Generic**」プロトコルハンドラを使用できます。この場合、そのプロトコル用に特化された処理を利用することはできません。

UDP によるトラフィックもサポートされます。基本的な UDP (DNS などの単純な UDP リクエスト-レスポンスプロトコル用)と UDP ストリーミング(サーバがリクエストに対するレスポンスとして UDP パケットを無制限に送信可能)のいずれかを選択できます。

特定のプロトコルのサポートに関する詳細については **CHAPTER 11** を参照してください。

注: SSL で暗号化されたトラフィックをトラフィックマネージャが復号化する場合、復号化されたトラフィックのプロトコルがプロトコル値として使用されます。例えば、トラフィックマネージャがポート **443** で **HTTPS** トラフィックを受信し、SSL 復号化によってそのトラフィックを復号化する場合、トラフィックマネージャ内で使用されるプロトコルは **HTTP** になります。

ルールの適用

仮想サーバは、実行するアクションを選択するために、**TrafficScript** ルールを使用してリクエストをチェックします。各ルールによってリクエストがチェックされ、場合によっては変更され、最終アクションが実行されます。

- リクエストを処理するプールを選択する。
- リクエストを終了する。
- トランザクションを記録する。
- 何も行わず、リクエストをリスト内の次のルールに渡す。

各仮想サーバには、デフォルトプールが関連付けられます。どのルールでも明確なルーティングが決定されなかったリクエストは、このプールに割り当てられます。

ルールでは、特定の接続について、トラフィックマネージャの機能を選択的に有効化または無効化することもできます。例えば、ルールによって、リクエストが特定のセッション維持クラスを使用するようにしたり、レスポンスをキャッシュしないように指定することが可能です。

ルールは、**Brocade** の **TrafficScript** 言語を使用して作成します。この言語では、受信するリクエストについて、送信元および宛先ポートや IP からトラフィックのタイプや実際のコンテンツまで、あらゆる側面をチェックすることができます。

TrafficScript では、XPath (XML ドキュメントのクエリに使用される言語)、XSLT、およびその他の XML 固有の機能がサポートされます。これらの XML 機能は、Web サービスで採用されている SOAP ベースの protocols で広く使用されており、ユーザが介入しなくても複雑なデータを自動的に交換したり解釈したりすることができます。

ルールは、指定した順序で適用されます。ルールは、トランザクションの完了時に、受信する各リクエストおよび外部に送信されるレスポンスに適用することができます。

ルールの作成はルールカタログで行い(管理 UI の **Catalogs (カタログ)** > **Rules (ルール)** ページを参照)、仮想サーバを介して適用します。全ての仮想サーバは、カタログの任意のルールを使用することができます。

仮想サーバにルールを追加するには、その仮想サーバの **Virtual Servers (仮想サーバ)** > **Edit (編集)** ページに移動し、**Rules (ルール)** リンクをクリックします。リクエストルールの場合は、「**Request Rules (リクエストルール)**」セクションを使用します。同様に、レスポンスルールおよびトランザクション完了ルールの場合は、それぞれ「**Response Rules (レスポンスルール)**」および「**Transaction Completion Rules (トランザクション完了ルール)**」セクションを使用します。目的のセクションで、ドロップダウンボックスから追加する新しいルールを選択し、**Add Rule (ルールを追加)** をクリックします。

仮想サーバのルールはリストに表示され、表示されている順序で適用されます。リスト内のルールは、各ルールバーの左にあるグリップハンドルを使用して上下に移動できます。マウスカーソルを使用して、新たな位置にルールをドラッグアンドドロップすると、トラフィックマネージャは自動的にリストの並び替えを行います。ルールが 1 つしかない場合、グリップハンドルは表示されず、セクション間でルールをドラッグすることはできません。

リスト内のルールは、一時的に実行されないよう無効化したり、再び有効化して再度アクティブにしたりすることができます。

多数のリクエストとレスポンスによる長いダイアログを 1 つのコネクションで実行する非 HTTP プロトコルの場合、ルールを一度のみ(最初のリクエストに対してのみ)実行するか、全てのリクエストに対して実行するかを選択できます。CHAPTER 8 では、ルールの作成と適用について詳しく説明します。TrafficScript 言語については、Brocade の Web サイト(<http://www.brocade.com>)からダウンロードできる別冊のガイドに記載されています。

SSL 復号化

仮想サーバは、SSL トラフィックを復号化することができます。この機能は、以下の2つの理由から有益です。

1. 復号化の後に、ルールによってリクエストのヘッダとコンテンツを解析し、これらの情報に基づいてルーティングを決定することができます。復号化しない場合、パケットから得られる情報はほんのわずかです。
2. リクエストを復号化するには、処理能力が必要です。トラフィックマネージャがノードに渡す前にリクエストを復号化すれば、バックエンドサーバの負荷が軽減されるため、効率の向上を図ることができます。

ルールを適用するためにトラフィックを復号化した場合、バックエンドサーバへの送信前にトラフィックを再暗号化することもできます。再暗号化は、プールで処理されます(CHAPTER 4 の「SSL 暗号化」セクションを参照)。

仮想サーバで SSL トラフィックを復号化するよう設定するには、その仮想サーバの **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページに移動し、**SSL Decryption** (SSL 復号) をクリックします。トラフィックを復号化するかどうか、SSL 証明書カタログのどの証明書を使用するかを選択できます。各仮想サーバに対して許可する暗号スイートと SSL または TLS (トランスポート層セキュリティ) のバージョンを設定することもできます。

SSL クライアント証明書をリクエストするかどうかを選択することもできます。クライアント証明書はクライアントの識別に役立ち、選択した個人にのみアクセスを限定するために使用することができます。

トラフィックマネージャでは、OCSP (オンライン証明書ステータスプロトコル) を使用したクライアント証明書の確認も可能です。OCSP は証明書失効リスト (CRL) に代わるもので、トラフィックマネージャは OCSP を使用してクライアント証明書の失効ステータスを取得することができます。TLS コネクションを行っているクライアントは、TLS ハンドシェイクの一部としてサーバの証明書のためのステータス情報を提供するように、仮想サーバにリクエストすることができます。OCSP Stapling を有効にすると、トラフィックマネージャに必要な OCSP レスポンスを読み込ませ、そのレスポンスをハンドシェイクメッセージに含めるように指示することができます。

トラフィックマネージャの SSL 機能については、CHAPTER 13 で詳しく説明します。

SSL パススルートラフィックの復号化

これまでに説明したとおり、仮想サーバはプロトコル値として、トラフィックマネージャが管理する SSL 復号化などの変換実行後の内部プロトコルを使用します。

仮想サーバのプロトコル値が「SSL」に設定されている場合、仮想サーバが SSL パススルーモードで SSL トラフィックの転送のみを行うことを示します。SSL 復号化

を設定するには、まずプロトコル値を、内部プロトコルを表す正しい値(例: HTTP)に変更する必要があります。この場合、プールから、SSL 暗号化トラフィックを想定しているノードにトラフィックが送信される可能性が高いため、プールで SSL 暗号化を設定することも必要です。

SSL Decrypt a Service (サービスの SSL 復号)ウィザードを使用すると、SSL パススルーサービスを設定し、トラフィックを仮想サーバで復号化してプールで再暗号化することができます。このウィザードについては、CHAPTER 13 の「SSL 復号化ウィザード」セクションで説明します。

SSL 復号化がサポートされるのは、一部のプロトコルのみです。UDP ベースのプロトコル、すなわち SIP など SSL で自動的にラップできないプロトコルに SSL 復号化を使用することはできません。

サービスプロテクションクラス

サービスプロテクションクラスは、サービスを DoS (サービス妨害)や DDoS (分散型サービス妨害)など悪意のある攻撃から保護するために指定する一連の設定です。サービスプロテクションクラスはサービスプロテクションカタログ内に作成でき、以下のような設定を行います。

- 禁止 IP と承認 IP のリスト。禁止 IP アドレスからの接続が許可されることなく、また承認 IP アドレスからの接続は常に許可されます。
- 1 台のマシンまたはマシングループからの接続数に関する制限。
- 任意の 1 つの IP アドレスからの接続レートに関する制限。
- RFC2396 を厳密に遵守しなければならないかなど、HTTP リクエストに関する制限。

サービスプロテクションクラスを仮想サーバに適用するには、該当する **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの **Classes** (クラス) リンクをクリックします。**Service Protection** (サービスプロテクション) セクションで、リストからクラスを選択し、**Update** (更新) をクリックします。

サービスプロテクションについては、CHAPTER 15 で詳しく説明します。

帯域制御クラス

帯域制御クラスでは、仮想サーバがクライアントへの送信データに適用できる帯域制限を定義します。例えば、複数の大容量ダウンロード接続に 250Kbits の共有制限帯域クラスを割り当てた場合、これらの接続は、この制限を超えて帯域を使用することはできません。

帯域制御クラスを仮想サーバに適用するには、該当する **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの **Classes** (クラス) リンクをクリックします。 **Bandwidth Management** (帯域制御) セクションで、リストからクラスを選択し、**Update** (更新) をクリックします。これにより、指定された制限共有タイプ(コネクション単位、マシン単位、またはクラス全体)に従って、仮想サーバが管理する全てのトラフィックを、そのクラスに定義された値に制限する設定が有効になります。

帯域制御については、CHAPTER 16 で詳しく説明します。

注: トラフィックマネージャの設定によっては、帯域制御を利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

サービスレベルモニタリングクラス

サービスレベルモニタリング(SLM)クラスは、サービスのエンドユーザが受けるサービスレベル(レスポンス時間)をモニタリングするために使用します。SLM クラスでは、目標とするレスポンス時間を定義します。また、許容限度のパーセンテージを定義することもできます。目標レスポンス時間を満たしたリクエストのパーセンテージがこの限度を下回ると、アラートまたはログメッセージが報告されます。

サービスレベルモニタリングクラスを仮想サーバに適用するには、該当する **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの **Classes** (クラス) リンクをクリックします。 **Service Level Monitoring** (サービスレベルモニタリング) セクションで、リストからクラスを選択し、**Update** (更新) をクリックします。これにより、仮想サーバは、そのサーバが管理する全てのコネクションのレスポンス時間をモニタリングして記録するようになります。また、サービスレベルに問題が生じた場合は、ログメッセージまたはアラートを報告します。

サービスレベルモニタリングについては、CHAPTER 18 で詳しく説明します。

注: トラフィックマネージャの設定によっては、サービスレベルモニタリングを利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

HTTP コンテンツキャッシング

トラフィックマネージャの HTTP 仮想サーバは、頻繁にリクエストされる HTTP リソースを検出し、そのリソースがリクエストのたびに変わらない場合はそのコンテンツを記憶することができます。この機能を「コンテンツキャッシング」と言います。

仮想サーバは、以降、同じリソースに対するリクエストを検出すると、そのリクエストを(過負荷状態にある可能性のある)バックエンドサーバに転送せずに、リソース

のコンテンツをローカルキャッシュから直接返すことができます。この機能により、バックエンドサーバの負荷が軽減され、HTTP サービスの性能と処理能力が向上します。

Virtual Servers (仮想サーバ) > **Edit** (編集) ページの **Content Caching** (コンテンツキャッシング) リンクをクリックすると、コンテンツキャッシングの設定が表示されます。ここでは、コンテンツキャッシング機能を有効化したり、様々なタイプのコンテンツをキャッシュする時間を指定することができます。コンテンツキャッシングは、HTTP 仮想サーバと、HTTPS トラフィックを受け入れて復号化する仮想サーバでのみ利用できます。

コンテンツキャッシングについては、CHAPTER 19 で詳しく説明します。

注: トラフィックマネージャの設定によっては、HTTP コンテンツキャッシングを利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

Web Accelerator

Brocade Web Accelerator (Web Accelerator) は、Web コンテンツの最適化技術を提供します。ロードを高速化するために、HTML コード、画像、スクリプト、カスタムスタイルシートなど Web ページのマークアップと要素を自動的に最適化します。Web ページのロード時間を短縮することは、ユーザエクスペリエンスの改善に不可欠です。

可能な場合には、このように最適化された要素の各バージョンが内部的にキャッシュされるので、以降のリクエストへのレスポンスはさらに高速になります。高速化された Web ページは短時間でロードされるため、帯域利用が改善され、データトラフィックとサーバインフラストラクチャについても節約されます。

サービスに対して Web Accelerator を有効化するには、**Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの **Web Accelerator** リンクをクリックします。これは、HTTP 仮想サーバでのみ使用できる機能です。

Web Accelerator の設定の詳細については、CHAPTER 20 の「Web コンテンツを最適化するための Brocade Web Accelerator の使用」を参照してください。

注: トラフィックマネージャの設定によっては、Web Accelerator 機能を標準として利用できない場合があります。Web Accelerator が必要な場合は、ソフトウェアまたはライセンスキーのアップグレードについて Brocade サポートにお問い合わせください。

HTTP コンテンツ圧縮

トラフィックマネージャは、HTTP レスポンスをリモートクライアントに送信するときに圧縮することができます。これにより、帯域使用率を減らし、低速なコネクションを使用したクライアントへの大容量 Web ページの配信を高速化することができます。

ブラウザによっては、圧縮されたコンテンツを受信できない場合があります。このようなブラウザでは、このことが HTTP リクエストヘッダに指定されます。トラフィックマネージャは、ブラウザがコンテンツを解凍できる場合のみコンテンツを圧縮します。

また、一部の Web サーバでもコンテンツを圧縮できるため、トラフィックマネージャでコンテンツを圧縮せずに、この処理を Web サーバに分担させる方が効率的な場合もあります。ただし、トラフィックマネージャと Web サーバの両方で圧縮を有効化しても問題は生じません。

Virtual Servers (仮想サーバ) > Edit (編集) ページの **Content Compression (コンテンツ圧縮)** リンクをクリックすると、コンテンツ圧縮の設定が表示されます。ここでは、圧縮を有効化するかどうか、圧縮する MIME ファイルタイプ、および圧縮するドキュメントのサイズを選択することができます。

サイトでダウンロード用の大容量ファイルを提供する場合は、リクエストのたびにサーバで圧縮するのではなく、ファイルをあらかじめ圧縮しておくことをお勧めします。

コンテンツ圧縮のコントロール

トラフィックマネージャは、リクエストに「Accept-Encoding: gzip」ヘッダが含まれている場合に、HTTP レスポンスを圧縮できます。トラフィックマネージャは、TrafficScript ルールの実行の前後にリクエストを確認し、いずれかのリクエストにこのヘッダが存在している場合にコンテンツを圧縮します。

例えば、リクエストから Accept-Encoding ヘッダを削除するようルールを設定できます。この場合、バックエンドサーバがレスポンスを圧縮して送信することはありませんが(このヘッダを検出することがないため)、リモートクライアントが圧縮をサポートしている場合、トラフィックマネージャはバックエンドサーバからのレスポンスを圧縮します。この手法を使用すると、全ての圧縮タスクをバックエンドサーバからトラフィックマネージャに任せることができます。

また、TrafficScript 関数の `http.compress.enable()` および `http.compress.disable()` を使用して、トラフィックマネージャがコンテンツを圧縮するかどうかをコントロールすることもできます。

コネクション分析

トラフィックマネージャには、特定のコネクションに関する大量の情報を表示する機能があります。この機能は、仮想サーバが処理したトラフィックをトレースおよびデバッグするときに役立ち、**Services** (サービス) > **Virtual Servers** (仮想サーバ) > **Edit** (編集) > **Connection Analytics** (コネクション分析) で有効化できます。

このページには、以下の 2 つのオプションがあります。どちらのオプションにも、有効化されているときにのみ利用できる追加オプションがあります。

設定オプション	説明
recent_conns!enabled	有効化すると、この仮想サーバで処理されているコネクションが Activity (アクティビティ) > Connections (コネクション) ページに表示されます。

設定オプション	説明
recent_conns!save_all	この仮想サーバで処理されている全てのコネクションをデフォルトで Activity (アクティビティ) > Connections (コネクション) ページに表示するか、非表示にするかを選択します。さらに、TrafficScript を使用してこの動作をオーバーライドするかどうかを、個々のコネクションに対して指定します。 デフォルトで全てのコネクションを非表示にするには、No (いいえ) に設定します。この場合、TrafficScript 関数の <code>recentconns.include()</code> を使用して、選択的にコネクションをルールに含めます。 デフォルトで全てのコネクションを表示するには、Yes (はい) に設定します。この場合、TrafficScript 関数の <code>recentconns.exclude()</code> を使用して、選択的にコネクションをルールから除外します。 これらの関数の詳細については、オンラインヘルプの『TrafficScript Reference』、または Brocade の Web サイト (http://www.brocade.com) からダウンロードできる『TrafficScript Guide』を参照してください。
request_tracing!enabled	有効化すると、トラフィックマネージャは内部イベントに関するデータを収集します。このデータには、各リクエストを受信した時間、および各ルールを実行した時間が含まれます。
request_tracing!trace_io	このコネクションに対して個々の読み取りおよび書き込みイベントに関するデータ収集を有効にするには、Yes (はい) に設定します。

コネクション分析の使用

前述のオプションを有効化すると、**Activity** (アクティビティ) > **Connections** (コネクション) ページで個々のリクエストの詳細情報を得ることができます。各コネクションの横にある虫眼鏡アイコンをクリックすると、具体的な詳細が表示されます。詳細については、CHAPTER 7 の「**Activity** (アクティビティ) > **Connections** (コネクション)」セクションを参照してください。

Request Logging (リクエストロギング)

このページを使用すると、仮想サーバが処理するリクエストの詳細ログを保存することができます。ログは、RFC 3164 で規定されているプロトコルに従って、トラフィックマネージャのローカルファイル、またはリモートマシンで実行されている **syslog** デーモンに書き込むことができます。仮想サーバのリクエストロギングを有効化するには、該当する **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの **Request Logging** (リクエストロギング) リンクをクリックします。

デフォルトのロギング動作は、`log!save_all` 設定を使用して設定することができます。デフォルトでトラフィックマネージャに全てのコネクションを記録するように指示するには、**Yes** (はい) に設定します。さらに、**TrafficScript** 関数の `requestlog.exclude()` をルール内で使用することにより、特定のコネクションをログから除外することができます。デフォルトでトラフィックマネージャにコネクションを記録しないように指示するには、**No** (いいえ) に設定します。さらに、**TrafficScript** 関数の `requestlog.include()` をルール内で使用することにより、特定のコネクションを選択してログに含まれるようにすることができます。**Update** (更新) をクリックすると、設定が適用されます。

ファイルへのリクエストロギング

ファイル名と、関係する IP アドレス、関係するプールとノード、送受信されたバイト数などの詳細を記述するログエントリフォーマットを指定できます。リクエストされた URL、指定されたヘッダの値、HTTP メソッドなど、HTTP 固有のオプションも設定できます。

注: 他のログとは異なり、リクエストログはクラスタ全体で同期されず、各トラフィックマネージャがそれぞれ独自のログファイルを保持します。

ファイルにリクエストを記録するには、`log!enabled` を **Yes** (はい) に設定し、ログ用のファイル名を指定します。**Update** (更新) をクリックすると、設定が適用されます。

各ログが大きくなり過ぎないように、ログを自動的にローテーションさせることができます。そのためには、ファイル名で `%{Time-String}t` マクロを使用します。このマクロは、現在の時刻または日付に展開されます。

例えば、`%Y%m%d` マクロは 20131020 (2013 年 10 月 20 日の場合) のような値に展開されます。これにより、ログファイルは、マクロの値が変化する午前 0 時に自動的にローテーションするようになります。

トラフィックマネージャの仮想アプライアンスおよびクラウドインスタンスでは、ログファイルの命名とローテーションは、以下のように自動的に処理されます。

- ログファイルは 1 時間ごとにチェックされます。
- 1 GB に達したファイルはアーカイブされます。
- 現在の全ログファイルの累積サイズが 6 GB に達すると、サイズが最も大きいログから順に、累積サイズが 6 GB を下回るまでアーカイブされます。
- 残りのログは毎日アーカイブされます。

ログディレクトリが容量の 85% に達すると、トラフィックマネージャは、最も大きいディレクトリにある最も古いファイルから順に削除を開始します。

リクエストログの表示

[View Request Logs](#) (リクエストログ表示) リンクをクリックすると、リクエストログをリアルタイムに表示することができます。この操作により、**Activity** (アクティビティ) > **View Logs** (ログ表示) ページにリダイレクトされます。このページについては、CHAPTER 7 の「*Activity* (アクティビティ) > **View Logs** (ログ表示)」セクションで詳しく説明します。

リモートリクエストロギング

トラフィックマネージャは、ローカルディスク上のファイルではなく、リモート **syslog** サーバにリクエストを記録することができます。同一クラスタ内の全てのトラフィックマネージャは、同じ **syslog** サーバにトラフィックを記録します。

リモートロギングを設定するには、`syslog!enabled` を **Yes** (はい) に設定し、エンドポイント (IP アドレスまたはホストと、オプションでポート) を指定し、使用するログファイルフォーマットを選択します。**Update** (更新) をクリックすると、設定が適用されます。

重要: **syslog** プロトコルでは **UDP** が使用されますが、その信頼性は保証されていません。通信量の多いネットワークやデータ損失を伴うネットワークでトラフィックレベルが特に高いときは、ログレコードが破棄される可能性があります。

リクエストロギングのコントロール

リクエストは、選択的に記録させることができます。**TrafficScript** 関数の `request.setLogEnabled()` を使用すると、個々のリクエストのロギングを有効化または無効化することができます。ロギングを有効化すると、**Request Logging**

(リクエストロギング)ページの設定に基づいて、リクエストをどのように記録するか(ローカルログファイルまたはリモート **syslog** サーバ)が決定されます。

コネクション管理

Virtual Servers (仮想サーバ) > **Edit** (編集) ページでは、リモートクライアントと仮想サーバ間のコネクションを管理するための詳細設定を行うことができます。これには以下のような設定があります。

タイプ	説明
HTTP 設定	keepalive 設定、およびトラフィックマネージャが X-Cluster-Client-IP ヘッダをリクエストに追加するかどうかを設定します。
クッキー設定	トラフィックマネージャがどのようにクッキーを処理するかを指定します。
RTSP 固有の設定	RTSP で使用する特定のポート範囲を定義できます(RTSP 仮想サーバのみ)。詳細については、RTSP に関する章を参照してください。
SIP 設定	トラフィックマネージャがどのように SIP トラフィックを処理するかを指定します(SIP 仮想サーバのみ)。詳細については、SIP に関する章を参照してください。
ロケーションヘッダ設定	不正なプロトコル、ホスト名、またはポートを使用してノードが 301 (永続的移動) または 302 (一時的リダイレクト) ステータスコードを返した場合に、URL をリダイレクトすることができます(HTTP 仮想サーバのみ)。
FTP 設定	FTP データコネクションのセキュリティ設定とポート範囲を設定します(FTP 仮想サーバのみ)。
UDP 設定	タイムアウトと、1 つのコネクションに対して想定される最大レスポンスデータグラム数を設定します(UDP 仮想サーバのみ)。
タイムアウト設定	新しいコネクションと確立済みのコネクションのタイムアウトを設定します。

タイプ	説明
コネクションエラー設定	コネクションエラーの詳細ロギングを有効化します。
メモリ制限	クライアントが送信するデータとサーバが返すデータの最大バッファサイズを指定します。

タイプ	説明
低レベル設定	「close_with_rst」を有効化すると、FIN (終了)パケットではなく RST (リセット)パケットによってクライアントからのコネクションを切断するようにトラフィックマネージャに指示することができます。これにより、TIME_WAIT 状態を回避して、まれな状況で、行き場を失った重複パケットを安全に無視することができます。また、ピアによるデータ受信も完全に確保することができます。一般に、TCP コネクションはピアへの FIN パケットおよび返答としての確認応答 FIN パケットによって終了するため、TCP の状態遷移の意味を十分に理解している場合に限り、このオプションを有効化してください。このメカニズムによってデータ送信者は、キューイングされた全てのデータがピアによって読み込まれたことを確認することができます。これは、RST パケットによって切断するコネクションでは起こりえません。信頼性のないネットワークでは、送信されたデータが送信中に失われることがあります。 TIME_WAIT 状態の目的は、<A>:<port P>から:<port Q>へのアドレスを使用する 1 つの TCP コネクションからの行き場を失った遅延パケットが、同じアドレスを使用するその後の TCP コネクションによって受け入れられないようにすることです。すなわち、1 つ目のコネクションからの重複パケットがネットワーク内で遅延し、そのシーケンス番号が 2 つ目のコネクションのウィンドウに存在するときに、そのパケットが 2 つ目のアドレスにたどり着いた場合、カーネル内のネットワークスタックは、遅延パケットに 1 つ目のコネクションからのデータが含まれていることを特定することができず、データの破損を引き起こす可能性があります。カーネルは使用中のポート番号を増加させるため、このようなことはほとんど起こりませんが、その可能性はゼロではありません。 proxy_close を有効化すると、クライアント FIN をバックエンドサーバに送信し、ただちにコネクションを切断せずにサーバのレスポンスを待機させることができます。 so_nagle を有効化すると、ネットワーク経路で送信する必要のあるパケット数を減らすことにより、TCP コネクションの効率を向上させることができます。

タイプ	説明
透過プロキシ	この仮想サーバでバインドされたソケットに対して透過プロキシを有効化するには、「transparent (透過)」を Yes (はい)に設定します。この設定では、ファイアウォールルールを使用して、宛先アドレスを変更せずに、選択したトラフィックを仮想サーバに送信できます。 TrafficScript リクエストルールでは、 「request.getDestIp」または「request.getDestPort」を使用してリクエストの宛先 IP アドレスまたはポートを確認することにより、実行される処理を判断できます。次に、「pool.select」を使用して元の宛先に対して、フォワードプロキシを実行できます。

特定のプロトコルに固有の設定は、仮想サーバがそのプロトコルを管理している場合のみ表示されます。

通常、これらの設定を変更する必要はありません。デフォルト値は、幅広いシステムで良好な性能が得られるよう選択されています。

MIME タイプ自動検出

MIME タイプは HTTP レスポンスのコンポーネントで、画像、HTML ページなど、レスポンス内のデータのタイプを記述します。これにより、クライアントはそのレスポンスをどのようにレンダリングまたは処理するかを判断できます。

Web サーバや Web アプリケーションのなかには、MIME タイプレスポンスヘッダを正しく設定しないものがありますが、多くのクライアントソフトウェアは、この間違いを検出し、MIME タイプ自体を自動検出することができます。

状況によっては、トラフィックマネージャでもレスポンスの MIME タイプを認識する必要があります。例えば、トラフィックマネージャは、特定タイプの HTTP レスポンスデータのみ圧縮しようとしませんが、MIME タイプが欠落しているか正しく指定されていないと、トラフィックマネージャのコンテンツ圧縮は正しく機能しません。

このような場合、**Connection Management** (コネクション管理)設定の **HTTP Specific Settings** (HTTP 固有設定)セクションで「MIME Type auto-detection (MIME タイプ自動検出)」を有効化することで対処できます。トラフィックマネージャは、一連の経験則を適用してレスポンスデータの MIME タイプを推測します。自動検出は、以下の全ての条件に当てはまる場合に実行されます。

- レスポンスボディが存在する。

- Content-Type ヘッダが欠落している、またはヘッダと **mime!default** の値が一致する。
- レスポンスが圧縮されていない。

チャンクトランスファエンコーディングが使用されている場合、トラフィックマネージャは、必要に応じてデータをデチャンクします。

MIME タイプ自動検出を使用すると、トラフィックマネージャシステムの性能が低下するため、絶対に必要な場合のみ設定してください。

ロケーションヘッダ設定

この設定では、301 (オブジェクトが永続的に移動)および 302 (オブジェクトが一時的に移動)という、Web サーバから送信される最も一般的な URL リダイレクション レスポンスを書き換えることができます。

- **location!regex** - 想定されるノードからのレスポンスに一致する正規表現を入力します(301/302 リダイレクション)。(.)などの汎用文字を使用して、表現の一部に一致させることができます。
- **location!replace** - \$1～\$9 を使用して正規表現を入力し、レスポンスを別のオブジェクトにリダイレクトします。
- **location!rewrite** - ノードからのレスポンスが **location!regex** に指定した表現と一致しなかった場合の希望レスポンスを選択します。

例えば、以下の値を使用するとします。

- **location!regex**=(.*)www.example.com/products/(.*)
- **location!replace**=\$1products.example.com/\$2

ここで、Web サーバが以下のようなレスポンスを返したとします。

```
HTTP/1.1 302 Moved Temporarily
Location: http://www.example.com/products/hacksaw.htm
```

トラフィックマネージャは、これを以下のように書き換えます。

```
HTTP/1.1 302 Moved Temporarily
Location: http://products.example.com/hacksaw.htm
```

エラーの処理

Connection Management (コネクション管理) ページの **Connection Error Settings** (コネクションエラー設定) セクションでは、仮想サーバで発生したコネクションエラーのロギングを有効化するための設定を行えます。

コネクションエラーは、以下のような様々な理由で発生します。

- 選択されたプール内のどのノードも使用できない。
- バックエンドサーバとのコネクションがタイムアウトになり、再試行できなかった(CHAPTER 14 の「失敗したリクエストの再試行」セクションを参照)。
- サーバ側またはクライアント側のコネクションで SSL プロトコルエラーが発生した。
- バックエンドサーバから有効なレスポンスが返されなかった(例: 不正形式の HTTP レスポンス)。

クライアントコネクションエラーは、クライアントのアクションや動作が原因である可能性が高く、その多くはトラフィックマネージャがコントロールすることができません。サーバコネクションエラーは、トラフィックマネージャからバックエンドサーバへのコネクションに関連しているため、トラフィックマネージャとバックエンドサーバの設定の影響を受けている可能性が高く、これらの設定によって対処することができます。

一般に、コネクションエラーは、パッシブモニタリングによって検出されるバックエンドサーバでのエラーに関連していない限り、記録されません。

以下のエラーを記録させると、デバッグに非常に役立つことがあります。

- **log!client_connection_failures** - この設定により、エラーログ、またはトラフィックマネージャのイベント処理設定で指定されている場合は別の場所へのクライアントコネクションエラーの詳細ロギングが有効化されます。
- **log!server_connection_errors** - この設定により、サーバコネクションエラーと、トラフィックマネージャのあらゆる内部障害の詳細ロギングが有効化されます(例: **max_instr** 制限を超えたことによる TrafficScript ルールの中断)。

詳細については、本マニュアルの CHAPTER 31 のトラブルシューティングに関する記述を参照してください。

カスタムエラーメッセージの返信

トラフィックマネージャがリクエストに対する有効なレスポンスを得られず、クライアントコネクションがまだ存続している場合、トラフィックマネージャはそのクライアントコネクションを切断するか、(HTTP のみ)以下のエラーをクライアントに返します。

```
Service Unavailable
```

```
The service is temporarily unavailable. Please try again later.
```

このエラーメッセージは、conf/extra リソースフォルダにアップロードされているファイルのコンテンツを使用して置き換えることができます。**error_file** 設定を使用してエラーメッセージを選択してください。

ファイルを conf/extra リソースフォルダにアップロードするには、**Catalog** (カタログ) > **Extra Files** (追加ファイル) > **Miscellaneous Files** (その他のファイル) 設定ページを使用します。

このファイルの処理方法について明確な指示を設定したい場合、トラフィックマネージャは、実際のファイルタイプに関わらず、エラーファイル内の HTTP ヘッダを処理できます。例えば、GIF または JPG 画像ファイルなどの特定タイプのファイルを表示する場合は、Content-Type ヘッダを設定できます。この場合、バイナリファイルの先頭に以下の行を挿入します。

```
Content-Type: image/gif
```

この機能を拡張して、以下の例のように HTTP レスポンスコードをオーバーライドすることもできます。

```
HTTP/1.1 200 OK
```

この場合、トラフィックマネージャにより、目的のコードがレスポンスで強制的にクライアントへ返されます。

注: ヘッダとファイルの実際の内容との間には、必ず空白行を挿入してください。

コネクション用メモリ制限

トラフィックマネージャには、仮想サーバが各コネクションに対して使用できるメモリ量を制限できる設定があります。

- **max_client_buffer** – トラフィックマネージャが(サーバに送信するために)クライアントから受信するデータ量のうち、バッファ処理される量を制限します。設定された制限値に達すると、コネクションの状態に応じて、トラフィックマネージャは様々な方法でレスポンスを返します。
 - トラフィックマネージャが HTTP ヘッダで読み込みを行っている場合、413 HTTP エラーコードがクライアントに送信され、警告が記録されます。

- トラフィックマネージャがリクエストルールを実行している場合、バッファ処理が継続され、警告が記録されます。
- トラフィックマネージャがサーバへのデータストリーミング段階に達している(リクエストルールの処理が終了している)場合、フロー制御が適用されます。この場合、クライアント側のコネクションが一時停止されるため、トラフィックマネージャは、バッファ処理されたデータの一部をサーバにプッシュできるようになるまでは、クライアントからのデータをそれ以上読み込みません。
- **max_server_buffer** – トラフィックマネージャが(クライアントに送信するために)サーバから受信するデータ量のうち、バッファ処理される量を制限します。設定された制限値に達すると、コネクションの状態に応じて、トラフィックマネージャは様々な方法でレスポンスを返します。
 - トラフィックマネージャがレスポンスヘッダで読み込みを行っている場合、500 HTTP エラーコードがクライアントに送信され、警告が記録されます。
 - トラフィックマネージャがレスポンスルールを実行している場合、バッファ処理が継続され、警告が記録されます。
 - トラフィックマネージャがクライアントへのレスポンスのストリーミングを開始している(レスポンスルールの処理が完了している)場合、フロー制御が適用されます。この場合、サーバ側のコネクションが一時停止されるため、トラフィックマネージャは、バッファ処理されたデータの一部をクライアントにプッシュできるようになるまでは、サーバからのデータをそれ以上読み込みません。

注: 両方の制限が同時にアクティブになることがあります。トラフィックマネージャがサーバへのリクエストボディのストリーミング段階に達しており、かつサーバが、リクエストボディ全体を読み込む前にレスポンスボディのストリーミングを開始した場合や、クライアントとサーバ両方の送信が速いにもかかわらず読み込みが遅い場合、トラフィックマネージャは両方のフロー制御メカニズムを同時にアクティブにすることがあります。

CHAPTER 5 プール

プールはバックエンドノードの論理グループです。各ノードは、サーバ名または IP アドレス、およびポートの組み合わせで表されます。

有効なノードエントリの例は以下のとおりです。

- server1.mysite.com:80
- 192.0.2.1:1234
- [2001:DB8:1b6c:6402:250:56ff:fea6:3f43]:4444

IPv6 アドレスとしてノードを指定するには、URI の角括弧表記法の[<IPv6 address>]:<port>を使用します。

注: トラフィックマネージャは、ノードに対する IPv6 リンクローカルアドレスの使用をサポートしていません。

仮想サーバがリクエストをプールに割り当てると、プールはそのプール内のノードにリクエストを負荷分散します。プール内の各ノードは、指定されたポートで、仮想サーバのプロトコルを使用してリクエストを受信できなければなりません。

各プールは負荷分散を行うほか、セッション維持、トラフィックの SSL 暗号化、およびオートスケーリングに関する設定をそれぞれ保持しています。これらの設定は、そのプールの設定ページから編集できます。

プールの設定ページにアクセスするには、トップメニューバーにある **Services** (サービス) ボタンをクリックし、次に **Pools** (プール) タブをクリックします。全てのプールのリストが表示されます。このページでは、新しいプールを作成したり、既存のプールの名前をクリックしてそのプールの **Pools** (プール) > **Edit** (編集) ページにアクセスしたりできます。

負荷分散

トラフィックマネージャでは、リクエストをプール内のノードに分配するための負荷分散アルゴリズムを選択することができます。以下のアルゴリズムがあります。

アルゴリズム	説明
ラウンドロビン	各バックエンドサーバに順々に接続をルーティングします。
加重ラウンドロビン	ラウンドロビンと同様ですが、各ノードには割合の異なるトラフィックが送信されます。各ノードの加重は、入力ボックスで指定する必要があります。

アルゴリズム	説明
パーセプティブ	各ノードの負荷とレスポンス時間をモニタリングし、トラフィックをどのように分配するのが最善かを予測します。これにより、レスポンス時間を最適化し、どのサーバも過負荷状態にならないようにします。
最小接続	現在の接続数が最小のバックエンドサーバを選択します。
加重最小接続	各サーバの重みを考慮して、現在の接続数が最小のバックエンドサーバを選択します。重みは、入力ボックスで指定できます。
最速レスポンス時間	現在のレスポンス時間が最速のバックエンドサーバにトラフィックを送信します。
ランダムノード	バックエンドサーバをランダムに選択します。

注: (トラフィックマネージャマルチサイトモードのみ) トラフィックマネージャ UI の他のエリアと同様に、ノードの加重を単一ロケーションベースの入力と複数ロケーションベースの入力で切り替えることはできません。ノードをロケーションごとに設定するよう選択した場合は、関連付けられた加重が使用ロケーションごとに自動的に表示されます。

最適な負荷分散手法について

一般に、トラフィックが均一の場合は、「**最小接続**」負荷分散アルゴリズムが最適です。この場合、全てのバックエンドサーバの性能が同じであれば、リクエストによってバックエンドサーバにかかる負荷は同じになります。多くの HTTP サービスはこの状況に該当します。「最小接続」手法では、一部のリクエストによって生成される負荷が他のリクエストより大きい場合でも(画像検索に対してデータベースの検索など)、リクエストがマシンに均等に分配されますが、各タイプのリクエストが十分にあれば、負荷は非常に効率的に分配されます。加重最小接続はこれを改善したもので、サーバによって処理能力が異なる場合に有効です。重みの大きいサーバは、その重みに比例してより多くのコネクションを受け取ります。

個々の高負荷のリクエストによって速度が著しく低下し、このようなリクエストが頻繁でない場合、最小接続は適していません。また、サーバごとに処理能力が異なる場合も同様です。「**最速レスポンス時間**」アルゴリズムでは、現在の性能が最高(レスポンスが最速)のサーバにリクエストが送信されますが、リアクティブアルゴリズム(イベント後に初めて速度低下を認識)であるため、高速なサーバが過負荷状態に陥りやすく、これによってサーバの性能が大きく上下します。

「**パーセプティブ**」は、「最小接続」と「最速レスポンス」の特長を取り入れた設計となっており、トラフィックの性質とサーバの性能に応じて負荷分散方法を変化させます。トラフィックが均一なときは「最小接続」の特性が強くなり、負荷が大

大きく増減するときは「最速レスポンス時間」の特性が強くなります。現在の接続数と最近のレスポンス時間から各サーバの性能の傾向を捉えて予測します。

このアルゴリズムでは、新しいサーバ(または障害状態から復旧したサーバ)に割り当てられるトラフィックは、最初は少なく、その後完全な運用レベルまで徐々に増加します。新しいサーバがプールに追加されると、そのサーバに単一のリクエストを送信して試します。レスポンスを受信すると、新しいサーバの負荷がプール内の他の同等ノードと同じ割合になるまで、そのサーバに送信するリクエスト数を徐々に増やします。このリクエスト数の増加は、サーバのレスポンス速度を考慮しながら行われます。そのため、例えば、少量の静的コンテンツを処理する新しい Web サーバの場合は、最大速度まで急速に増加し、初めて使用する JSP をコンパイルする Java アプリケーションサーバ(これによって開始時のレスポンスが遅くなる)の場合は、ゆっくり増加します。

「最小接続」は「パーセプティブ」より単純で決定的であるため、状況が適しているときはこのアルゴリズムを優先的に使用してください。

負荷分散アルゴリズムに関する注意事項

最小接続、最速レスポンス時間、およびパーセプティブは、トラフィックレベルが非常に低いときに予測外の動作を示すことがあります。処理するコネクションが一度に 1 つのみの場合、最小接続ではリクエストが分配されません。また、パーセプティブおよび最速レスポンス時間では、レスポンス時間が良好であることが分かっているノードに優先的にリクエストが送信される傾向にあり、テストされていないノードは無視されます。

負荷分散のメトリクスは、クラスタ内のトラフィックマネージャ間で共有されません。例えば、2 台のトラフィックマネージャがラウンドロビンアルゴリズムでリクエストを分配する場合、それぞれのサーバが独立して、ノードに順々にリクエストを送信します。

トラフィック管理ソフトウェアをマルチコアサーバで運用する場合、多くの負荷分散メトリクスがプロセス(CPU コア)間で共有されます。ただし、レスポンス時間情報は例外で、コア間で共有されません。

バックエンドサーバをテストし、トラフィックが全てのバックエンドサーバに送信されていることを確認する(負荷分散よりもリクエストの分配を重視する)場合は、テストトラフィックに対して「ラウンドロビン」または「ランダム」を使用してください。

局所認識リクエスト分散(LARD)

パーセプティブ、最小接続、および最速レスポンス時間の各アルゴリズムでは、LARD (局所認識リクエスト分散)と呼ばれる手法によって、同じリクエストを同じバックエンドサーバに送信しようとします。この手法ではサーバのキャッシングが利用されます。すなわち、サーバが特定項目のコンテンツを返す場合、そのコンテンツは内部キャッシュ(メモリまたはディスク)に保存されるため、同じコンテンツで

あれば、以降のリクエストに対してコンテンツを迅速に提供できる可能性が高くなります。

各アルゴリズムでは、負荷分散に関する決定を行うときに、以前に同じリクエストを処理したノードに関する情報が考慮されます。

- 「最小接続」では、接続数が最小のノードが複数ある場合、これらの候補ノードのうち、以前同じリクエストを処理したことのあるノードが選択されます。
- パーセプティブおよび最速レスポンス時間アルゴリズムでは、内部選択時に、以前同じリクエストを処理したことのあるノードにさらにわずかな重みが付加されます。

局所認識リクエスト分散は、バックエンドノードへのリクエストのルーティング方法についてトラフィックマネージャにアドバイスする軽量な方法です。同じ URL に対するリクエストが同じバックエンドサーバに送信されるようにするには、ユニバーサルセッション維持(URL でキー設定) (CHAPTER 12 を参照)を使用して、さらに強制的な方法でリクエスト送信を行ってください。または、TrafficScript™とネームドノードセッション維持、フォワードプロキシ、またはプール選択手法を使用すれば、リクエストのルーティングを完全にコントロールできます。

セッション維持

セッション維持は、同じクライアントセッションによる全てのリクエストを同じバックエンドサーバに送信するプロセスです。任意の TCP または UDP プロトコルで使用できます。

静的 Web コンテンツを処理するプールは、通常、セッション維持を必要としません。これは、特定のクライアントに各ページまたは画像を異なるマシンから提供しても、悪影響はないからです。一方、オンラインショッピングサイトを処理するプールでは、セッション維持を使用すると、ユーザのリクエストを、そのユーザの買い物かごの詳細が保持されているノードに必ず送信するようにできます。

トラフィックマネージャでは、いくつかの方法で同じセッションに属するリクエストを識別することができます。様々なクッキーを使用したり、一定のルールに基づいてセッションを維持したり、またはクライアントの IP アドレスによってセッションを識別したりすることもできます。受信トラフィックが SSL で暗号化されている場合は、SSL セッション ID を利用できます。

セッション維持が失われたときの対応を選択することができます。セッション維持が失われる原因には、無効なセッションデータや、処理を行っているノードでの障害発生などがあります。この場合、コネクションを切断するか、リクエストを新しいノードに送信するか、ユーザをエラーページなどの指定 URL にリダイレクトするかを選択できます。

セッション維持をプールに適用するには、そのプールの **Pools (プール) > Edit (編集)** ページの **Session Persistence (セッション維持)** リンクをクリックします。セッション維持クラスを選択し、**Update (更新)** ボタンをクリックします。

重要: 他のプールで既に使用されているセッション維持クラスを使用するときは注意が必要です。セッション維持クラスは、このプール内のノードが、そのクラスを既に使用しているプールのノードと一致している場合のみ再利用してください。それぞれノードが異なる複数のプールで同じセッション維持クラスを使用した場合、セッションアフィニティを保証できません。

セッション維持については、CHAPTER 12 で詳しく説明します。

帯域制御

トラフィックマネージャは、バックエンドノードへのコネクションに使用する帯域に制限を適用することができます。これらの制限は、バックエンドノードがリモートデータセンタに存在し、そのリモートサイトへの帯域を管理する必要がある場合に役立ちます。

帯域制限をプールに適用するには、そのプールの **Pools (プール) > Edit (編集)** ページの **Bandwidth Management (帯域制御)** リンクをクリックします。帯域制御クラスを選択し、**Update (更新)** ボタンをクリックします。

注: 帯域制限は、トラフィックマネージャシステムからリモートサーバノードに送信されるデータにのみ適用されます。リモートノードからトラフィックマネージャに戻る帯域には適用されません。また、帯域制御は、UDP ベースのプロトコルでは動作しないことに注意してください。帯域制御については、CHAPTER 16 で詳しく説明します。

注: トラフィックマネージャの設定によっては、帯域制御を利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

ヘルスマニタリング

トラフィックマネージャは、ノードに送信したリクエストに対するレスポンスを受信しない場合、そのノードで障害が発生したものと見なします。この場合、そのノードへのリクエスト送信を停止し、プール内の残りのノードにトラフィックを負荷分散します。

トラフィックマネージャのヘルスマニタでは、バックエンドサーバノードに対して高度なヘルスチェックを実行し、ノードが正常に動作しているかどうかを判断することができます。

ヘルスマニタがプールに割り当てられている場合、ヘルスマニタによってプール内の各ノードがチェックされます。特定数の障害が検出された場合、トラフィックマネージャは、そのノードまたはプールが使用不可であると見なします。これらのヘルステストは、トラフィックマネージャがノードにリクエストを送信し、ノードからのレスポンスを読み取るに行われる接続テストに加えて実行されます。

ヘルスマニタでは、各ノードに対する **Ping** の実行などの単純なテストから、該当するポートが開いていてノードがホームページなど指定されたデータを処理できるかどうかをチェックする高度なテストまで、幅広いテストを実行できます。特定のタスク(**POP3** ログインの検証など)を実行するための設定済みのモニタが多数用意されているほか、必要な任意のテストを実行するためのカスタムモニタを作成することもできます。

モニタは、**ノード単位**と**プール単位**の2つのカテゴリに分類されます。ノード単位のモニタは、プール内の各ノードのヘルスをテストします。プール全体のモニタは、プール全体のヘルスに影響を与える特定のマシンに対してテストを実行します。例えば、メールサーバプールがデータを **NFS** サーバに保存し、各バックエンドサーバがその **NFS** サーバにアクセスしている場合、プール全体のモニタではこの **NFS** サーバをテストできます。このサーバで障害が発生すると、どのバックエンドサーバもデータを読み込むことができないため、プール全体で障害が発生したものと見なされます。

ヘルスマニタをプールに割り当てるには、**Pools (プール) > Edit (編集)** ページの **Health Monitors (ヘルスマニタ)** リンクをクリックします。設定を選択したら、**Update (更新)** をクリックして設定を適用します。

ヘルスマニタについては、CHAPTER 14 で詳しく説明します。

SSL 暗号化

データをトラフィックマネージャからバックエンドサーバに送信する前に暗号化したい場合があります。

SSL 暗号化設定を指定するには、**Pools (プール) > Edit (編集)** ページの **SSL Settings (SSL 設定)** リンクをクリックします。設定を選択したら、**Update (更新)** をクリックして設定を適用します。

SSL の設定については、CHAPTER 13 で詳しく説明します。

コネクション管理

Pools (プール) > Edit (編集) ページの **Connection Management (コネクション管理)** リンクをクリックすると、トラフィックマネージャとノード間のコネクションを管理するための設定を行えます。これには以下のような設定があります。

- トラフィックマネージャからノードへの接続時に、接続元として強制的に特定の IP アドレスを使用するかどうか
- ノードへの HTTP keepalive コネクションを維持するかどうか
- コネクションを確立するまでの最大回数、またはノードからのレスポンスを待機する最大回数
- 断念してクライアントにエラーを返す前に試行する最大ノード数
- トラフィックマネージャからノードへのコネクションをデフォルトの FIN パケットではなく RST パケットによって切断するかどうかこの設定の効果についての詳細は、77 ページの「コネクション管理」を参照してください。

通常、これらの設定を変更する必要はありません。デフォルト値は、幅広いシステムで良好な性能が得られるよう選択されています。

プールコネクション制限

はじめに

全てのサーバまたはアプリケーションは、特定量のシステムリソースを使用してコネクションを処理します。これらのリソースには、スレッド、ファイル記述子、一定量のメモリなどが含まれます。各バックエンドサーバは、特定数のコネクションまでは、性能を最適な状態に保ちながら同時に処理することができますが、それ以上のコネクションを処理しようとするとうと性能が低下します。極端な場合、過負荷状態のシステムが引き続きリクエストに応えようとして、サーバが新しいコネクションをキューイングしたか、破棄したか、または拒否したように見えることがあります。

このような場合、サーバが受け取る同時接続数に制限を設けることで、サーバを最適なレベルで継続的に動作させるとともに、過負荷状態を確実に防ぐことができます。ノードのプールに対して行える同時接続数を制限する 1 つの方法がプールコネクション制限です。コネクション制限によってトラフィックマネージャの他の多くの機能が影響を受ける可能性があるため、コネクション制限機能を使用する前に、このセクションを全てお読みください。

プールコネクション制限

コネクション制限はプールに対して設定しますが、実際にはそのプール内の各ノードに適用されます。例えば、コネクション制限を 50 に設定すると、そのプール内の各ノードが同時に受け取る最大接続数は 50 になります。コネクション制限機能では、プール内のノードが全て同じ仕様で、同じアプリケーションを提供しているものと見なされます。特定のノードが複数のプールに存在する場合は、そのノードが同時に受け取る可能性のある接続数は、各プールに設定されているレート制限の合計になります。

クラスタ環境でのコネクション制限

バックエンドマシンへのコネクションは、クラスタ内のトラフィックマネージャ間で共有されません。アクティブ-パッシブ構成では、全てのコネクションがアクティブマシンで使用され、パッシブマシンでは全く使用されません。アクティブ-アクティブ構成では、コネクション制限は事実上 2 倍になり、アクティブ-アクティブ-アクティブ構成では 3 倍になります。

コネクションキューイング

ノードが負荷分散アルゴリズムによって選択された場合、そのノードに対する現在の接続数が指定された制限に達すると、コネクションはキューに保存されます。接続リクエストが許容接続数を越えた状態が長時間続くと、(例えば HTTP の場合)ブラウザへのレスポンスがタイムアウトになる可能性があります。このリスクを軽減するために、TrafficScript 関数の `connection.checkLimits()` を使用して、プールでコネクションのキューイングが開始されたときにそれを検出することができます。この情報を使用して、(例えば)レスポンスがタイムアウトになるまで待たずに、ただちにクライアントに対して別のページをレスポンスとして返すことができます。詳細については、『*Brocade Virtual Traffic Manager: TrafficScript Guide*』を参照してください。

考慮事項

コネクション制限は複雑な機能で、トラフィックマネージャの他の機能と相互に影響し合います。このセクションでは、これらの影響について説明し、これらの機能を同時に使用する場合に何を想定すべきかを示します。

キュー管理と SIP

RFC 3261 - SIP: Session Initiation Protocol Section 17.2.1 では、SIP サーバのレスポンス時間が 100ms を超える場合に必要な処理について、以下のように具体的な動作を定義しています。トラフィックマネージャは、SIP 規格のこの部分に従って開発されています。

「サーバトランザクションでは、TU が 200ms 以内に暫定的または最終的なレスポンスを生成することが分かっている場合を除き、100 (試行中)レスポンスを生成する

必要があります。TU が 200ms 以内に暫定的または最終的なレスポンスを生成することが分かっている場合も、100 (試行中)レスポンスを生成することができます。」
[<http://www.faqs.org/rfcs/rfc3261.html>]

コネクション制限とリクエストレートシェイピング

トラフィックマネージャのリクエストレートシェイピング機能では、コネクション制限機能とは異なるキューメカニズムが使用されます。リクエストは、レートシェイピングクラスのキューから抜けた後もコネクション制限が適用され、そのコネクション制限によってキューイングされる可能性があるため、ノードに送信されるリクエストのレートが、レートシェイピングクラスで指定されたレートと異なる可能性があります。

コネクション制限とサービスレベルモニタリング

コネクション制限により、サービスレベルモニタリングが影響を受ける可能性があります。これは、リクエストがコネクション制限によってキューイングされる可能性があるためです。この場合、SLM クラスで測定されるレスポンス時間には、リクエストがコネクションキューイングされていた時間が加算されます。

プールコネクション制限の有効化

プールにコネクション制限を設定するには、以下の手順に従います。

1. **Services** (サービス) > **Pools** (プール) > **Edit** (編集) > **Connection Management** (コネクション管理)をクリックします。
2. `max_connections_pernode` パラメータを必要な数値に設定します。
3. ページの下部までスクロールし、**Update** (更新)をクリックします。

コネクション制限: オプション

コネクション制限に対して、以下の追加オプションを設定できます。

設定オプション	説明
<code>queue_timeout</code>	<code>queue_timeout</code> に指定した秒数を超えてキューイングされているコネクションは破棄されます。
<code>max_queue_size</code>	関連付けられた仮想サーバのプロトコルが HTTP に設定されている場合、クライアントに内部サーバエラー (HTTP レスポンスコード 500) が送信されます。この内部サーバエラーの HTML ページを設定するには、関連付けられた仮想サーバの「 Connection Error Settings (コネクションエラー設定)」設定ページで <code>error_file</code> パラメータを使用します。

コネクション制限のトラッキング

コネクション制限ではキューイングが行われるため、コネクションがタイムアウトになる可能性があります。現在のアクティビティグラフを使用すると、キューイングされている接続数と、キューイング中にタイムアウトになった接続数をトラッキングできます。

1. **Activity** (アクティビティ) > **Current Activity** (現在のアクティビティ) ページに移動します。
2. **Change data** (データ変更) ボタンをクリックします。
3. 選択されている全てのボックスの選択を解除します。
4. **Values** (値) ツリーの **Pools** (プール) ブランチを展開します。
5. **ConnsQueued** および **QueueTimeouts** に対応するボックスを選択します。
6. コネクション制限が機能しているかどうかを確認する場合は、**Values** (値) ツリーの **Nodes** (ノード) ブランチを展開します。
7. **CurrentConn** ボックスを選択します。
8. ページの下部までスクロールします。
9. **New Settings** (新規設定) フィールドに、新しいチャートの名前 (Connection Limits など) を入力します。
10. **New Settings** (新規設定) の横のラジオボタンをクリックします。
11. **Apply** (適用) ボタンをクリックします。

コネクション制限のテスト

zeusbench ユーティリティを使用すると、設定したコネクション制限が正しく機能するかどうかを確認することができます。**zeusbench** ユーティリティは `$ZEUSHOME/admin/bin` にあります。同時実行モードで **zeusbench** を実行すると、バックエンドノードへの空きコネクションを使い果たすのに十分なトラフィックを生成できます。(例えば) 2つのノードから成るプールがあり、各ノードのコネクション制限を 10 に設定した場合、以下の **zeusbench** コマンドを実行すると、コネクションがキューイングされます。

```
./zeusbench -c 100 -t 60 http://<your site>/index.html
```

この場合、**zeusbench** は、(URL で) 指定された仮想サーバに対して 100 のコネクションを確立して、できる限り早く各ノードをダウンさせるために十分なリクエストを送信しようとします。これにより、テスト対象となっているプールに対してこれとほぼ同数のアクティブなコネクションが確立されます。ノード数が多いほど、

zeusbench ユーティリティを呼び出すときに指定する必要がある同時接続数が多くなることに注意してください。

バックエンドのフォールトトレランス

トラフィックマネージャでは、フォールトトレランスを実現するための強力なアプローチが採用されています。トラフィックをできる限り効率よく管理できるよう、様々な方法でプールを設定することができます。

プール内のノードで発生した障害は、トラフィックマネージャのヘルスモニタによって検出されます。また、負荷分散中には、ノードがリクエストに対するレスポンスを返さないことから、障害を検出することもできます。それ以降、リクエストはそのノードに送信されず、プール内の他のノードに分配されます。障害から復旧したノードは、トラフィックマネージャが信頼性に問題がないと判断するまで、徐々に元の運用レベルに戻されます。

プールには障害プールを関連付けることができます。元のプールの全ノードで障害が発生した場合、リクエストはそのプールを迂回して障害プールに送信されます。

障害プールを選択するには、**Services** (サービス) ボタンをクリックして **Pools** (プール) ページに移動します。編集するプールの名前をクリックします。**Basic Settings** (基本設定) セクションで、ドロップダウンリストから障害プールを選択できます。

HTTP サービスの一般的な障害プールは、単一の「sorry server」で構成することができます。このサーバは、受信した全てのリクエストを、「使用不可」メッセージを表示する単純な Web ページのリクエストに書き換えることができます。

注: トラフィックマネージャでは、プールで障害が発生した場合に、単純なエラーページを直接提供することもできます。この方法をプールの **Connection Management** (コネクション管理) ページから設定する手順は、前述の「コネクション管理」セクションで説明しています。

プライオリティリスト

プール内にプライオリティリストを設定することができます。プライオリティリストでは、プライオリティに従ってノードをグループ化することができます。トラフィックを常時受信できるようにするマシンの最小数を指定できます。

通常のトラフィックに使用する 3 台のサーバ(primary1、primary2、および primary3)と、2 台のオフサイトバックアップサーバ(backup1 および backup2)があるとします。ここで、常時少なくとも 2 台のサーバを利用可能にするとします。

2 台のバックアップサーバを 3 台のプライマリサーバの障害プールとして設定することができます。ただしこの場合、障害プールが使用されるのは、プライマリプールの全てのノードで障害が発生した場合のみです。primary1 および primary3 で障

害が発生した場合は、primary2 のみで全てのトラフィックを受信することになります。

この方法の代わりに、グループ化されたプライオリティリストを設定できます。これには、{primary1, primary2, primary3}とその下の{backup1, backup2}の、合わせて2つのプライオリティグループがあります。また、常時少なくとも2台以上のノードを使用可能にするよう指定します。

全てのサーバが稼働している状態では、プライマリサーバ3台が全て使用され、バックアップマシンにトラフィックは送信されません。

primary1 で障害が発生した場合、上位グループの2つのノードでリクエストを受信できるため、この2つのノードで全てのトラフィックが処理されます。

ここで、さらに primary3 で障害が発生すると、上位グループで残されたノードは1つのみになります。2台のサーバを使用可能にするよう指定したため、トラフィックマネージャは、バックアップグループの両方のマシンにリクエストを送信し始め、リクエストは primary2、backup1、および backup2 で処理されるようになります。

プライオリティレベルはこれ以上ありません。さらにノードで障害が発生した場合、トラフィックは残されたノードに負荷分散されます。全てのノードで障害が発生した場合は、プールに障害プールが設定されていれば、障害プールにトラフィックが渡されます。

▼ Priority Lists

Priority Lists control the order of preference in which the nodes in a pool are used.

To alter the priorities of the nodes use the up and down buttons.

Enable priority lists
priority!enabled: ☐ Yes ☒ No

Minimum number of highest-priority active nodes
priority!nodes:

Highest Priority Group

Node	Up	Down
primary1:80	▲	▼
primary2:80	▲	▼
primary3:80	▲	▼

Lowest Priority Group

Node	Up	Down
backup1:80	▲	▼
backup2:80	▲	▼

Fig 16. プライオリティグループの設定例

プールのプライオリティグループを設定するには、**Pools** (プール) > **Edit** (編集) ページからそのプールを編集します。**Load Balancing** (負荷分散) をクリックし、**Priority Lists** (プライオリティリスト) セクションを展開します。

プール内のノードが **Highest Priority Group** (最高プライオリティグループ) リストに表示されます。このプールでプライオリティリストを有効化するには、**Yes** (はい) ラジオボタンをクリックし、常時使用可能にする最小ノード数を選択します。次に、各ノードの横にある上下矢印ボタンを使用して、最初のグループの上位または下位に別のグループを作成します。

注: 最上位のグループ内で使用可能なノード数が指定の最小ノード数を下回るまで、下位のプライオリティグループ内のノードは使用されません。使用可能なノード数が最小ノード数を下回ると、その下位にあるグループの全てのノードで処理が開始されます。これらの上位 2 グループの健全なサーバ数が指定の最小ノード数を下回るまで、これらの 2 つのグループのサーバが使用され、最小ノード数を下回った時点で、さらにその下位にあるグループ全体が使用されるようになります。これは、全てのプライオリティグループが使用されるまで続きます。

プール内の全てのノードで障害が発生した場合、障害プールが設定されていれば、このプールにリクエストが送信されます。

ノードのドレインと無効化

負荷分散が行われているクラスタからいずれかのバックエンドサーバを除外したい場合があります。これは、恒久的な措置の場合もあれば、メンテナンスやアップグレードのための一時的な措置である場合もあります。

トラフィックマネージャから新しいリクエストが送信されないようにするために、そのノードを**無効化**または**ドレイン**することができます。

ノードの無効化: トラフィックマネージャからそのノードに対してこれ以上コネクションが送信されなくなります。既存のコネクションは通常どおり完了します。トラフィックマネージャでは、そのノードのモニタリングが停止されます。

ノードのドレイン: トラフィックマネージャからそのノードに対してこれ以上新たなコネクションが送信されなくなりますが、既に確立されているセッションは維持されます。トラフィックマネージャがリクエストを受信し、セッションを維持するためにそのノードを使用する必要がある場合、トラフィックマネージャはそのノードを使用します。詳細については、CHAPTER 12 の「コネクションのドレイン」セクションを参照してください。トラフィックマネージャでは、そのノードがアクティブにモニタリングされます。

ノードの無効化またはドレインは、**Pools (プール) > Edit (編集)** ページから実行できます。目的のノードが含まれるプールを編集し、ステータスを **Disabled (無効)** または **Draining (ドレイン)** に変更して、**Update (更新)** ボタンをクリックします。

注: この操作では、そのプールでのみノードがドレインまたは無効化されます。そのノードを使用する他のプールでは、引き続きトラフィックが送信されます。

ノードの無効化

ノードでの処理を一時的に停止し、そのノードとの間で確立されているセッションを考慮する必要がない場合は、ノードを「無効化」します。この場合、(例えば、サーバを停止したために) ノードの動作が停止しても、トラフィックマネージャはエラーを報告しません。

ノードを構成から除去せずに無効化すると、そのノードが使用可能な状態であれば、無効化したノードを簡単に処理に復帰させることができるというメリットがあります。

ノードのドレイン

既に確立されているセッションを考慮する必要がある場合は、ノードを「ドレイン」します。ただし、ノードの動作が停止した場合、トラフィックマネージャからエラーが報告されることに留意してください。ノードを停止する場合は、まずそのノードをドレインし、全てのセッションが完了するまで待ってから無効化してください。

ノードがまだコネクションを処理しているかどうかを判断するには、**Activity (アクティビティ)** ボタンをクリックし、**Draining Nodes (ノードのドレイン)** タブに移動し

ます。このページには、ノードがまだ処理している接続数と、最後の接続からの経過時間が表示されます。ノードの既存の接続が終了した時点で、そのノードを無効化するか、**Remove a Node** (ノード削除)ウィザードを使用してクラスタから完全に削除することができます。

Drain a Node (ノードのドレイン)および Disable a Node (ノードの無効化)ウィザードの使用

様々なサービス全体でノードをドレインまたは無効化する処理を簡単に実行できるように、トラフィックマネージャには **Drain a Node** (ノードのドレイン)ウィザードと **Disable a Node** (ノードの無効化)ウィザードが用意されています。これらのウィザードには、トップメニューの「ウィザード」バーからアクセスできます。

ドレイン/無効化するノードを指定し(選択したウィザードによって異なります)、全てのサービスに対して実行するか、指定したサービスに対してのみ実行するかを選択します。ドレイン/無効化を実施したノードは、後から **Reactivate a Node** (ノードの再アクティブ化)ウィザードを使用してアクティブな状態に戻すことができます。

オートスケーリング

注: オートスケーリングはライセンスキーによって管理される機能であり、トラフィックマネージャ製品によっては利用できない場合があります。サポートされている設定の詳細については、Brocade の Web サイトを参照してください。

はじめに

アプリケーションオートスケーリングオプションは、サポートされている仮想またはクラウドプラットフォームで実行中のサービスの性能をモニタリングします。性能が目標サービスレベルを下回った場合、トラフィックマネージャはオートスケーリングアクションを開始し、プラットフォームに追加のサービスインスタンスをデプロイするよう要求することができます。また、新しいインスタンスが使用可能になった直後から、自動的にトラフィックをそのインスタンスに負荷分散します。オートスケーリング機能は、モニタリング/決定エンジンと、関連するプラットフォームと連携するドライバスクリプトの集合で構成されます。

オートスケーリングはプールのプロパティの 1 つです。有効化されている場合、プールの設定時に具体的なノードリストを指定する必要がありません。代わりに、性能が低下し始めると、必要とされる追加の処理能力を提供するために、自動的に追加ノードをプールに組み込むことができます。逆に、追加ノードが不要になったときには、プールを元の規模にスケールダウンしてノードを解放できます。このように、この機能を利用すると、トラフィックの一時的な急増や長期的な負荷の増加に動的に対応することができます。

内蔵サービスモニタは、プールの自動スケールアップまたはスケールダウンがいつ必要か判断するために使用されます。プールによって提供されるサービスレベル(レスポンス時間)は綿密にモニタリングされます。レスポンス時間が目標レベルを下回ると、最小のコストでサービスレベルを満たすために、オートスケーリングによってプールのノードが追加または削除され、リソースが増加または減少します。

オートスケーリングの動作

上の説明のとおり、オートスケーリングメカニズムは決定エンジンと、プラットフォーム依存のドライバスクリプトの集合で構成されます。

決定エンジン

決定エンジンは、プールからのレスポンス時間をモニタリングし、スケールアップ/スケールダウンしきい値を提供します。その他のパラメータは、プール内の最大ノード数および最小ノード数と、スケールアップまたはスケールダウン完了後にレスポンス時間が安定するまでトラフィックマネージャが待機する時間をコントロールします。

例えば、250ms という SLA を確保する場合について考えます。この SLA 内に完了するトランザクションが 50%を下回った場合に、最大 10 台のバックエンドノードまでスケールアップ(ノードを追加)するようトラフィックマネージャに指示することができます。または、この SLA 内に完了したトランザクションが 95%を超えた場合は、最小 1 台のノードまで段階的にスケールダウン(ノードを削除)するようにします。

注: プール内の最大ノード数と最小ノード数を編集して、手動でノードをプロビジョニングすることができます。この場合、トラフィックマネージャは、最大/最小ノード数と実際のアクティブなノード数の不一致を検出すると、一連のスケールアップまたはスケールダウンアクションを開始します。

クラウド API ドライバ

トラフィックマネージャには、Amazon EC2、Rackspace、および VMware vSphere クラウド環境用の API ドライバスクリプトが用意されています。オートスケーリングプールを作成する前に、使用するクラウド API に関連する一連のクラウドクレデンシャルを作成する必要があります。これらのクレデンシャルには、トラフィックマネージャが前述のクラウドプロバイダと通信するために必要な情報が保持されます。使用される具体的なクレデンシャルは、指定するクラウドプロバイダによって異なります。詳細については、CHAPTER 7 の「クラウドクレデンシャル」セクションを参照してください。

決定エンジンは、設定されたクレデンシャルとパラメータを使用してドライバを呼び出し、スケールアップまたはスケールダウンアクションを開始します。ドライバは、仮想マシンをデプロイまたは削除するよう仮想化レイヤに指示します。このアクションが完了すると、ドライバはプールの新しいノードリストを返し、決定エンジンがプールの設定を更新します。

設定

重要: プールにオートスケーリングを実装するためのメカニズムでは、トラフィックマネージャがプール設定ファイルを自動的に変更(必要に応じてノードを追加または削除するなど)することができます。このようなプールがアクティブに使用されている状況では、この自動更新プロセスと同時にユーザが管理 UI または SOAP インターフェースから設定を変更した場合、自動更新プロセスによってユーザによる変更が妨げられる可能性がわずかにあります。そのため、変更を行った場合は、プールページを更新して、行った更新が有効になったことを再確認することをお奨めします。

選択したクラウド API 用に一連のクレデンシャルを作成したら、選択したプールのオートスケーリングプロパティの設定に進むことができます。**Pools (プール) > Edit (編集)** ページの **Autoscaling (オートスケーリング)** リンクをクリックすると、利用可能な設定オプションが表示されます。

基本設定

設定	説明
<code>autoscale!enabled</code>	オートスケーリングを有効化または無効化します。この機能が有効化されている場合、ノードは、「 <code>autoscale!external</code> 」に指定したメカニズムに従ってプールに追加またはプールから削除されます。
<code>autoscale!external</code>	一部のクラウドプロバイダは、オートスケーリングを実行するための独自のメカニズムを持っています。この場合、クラウドプロバイダはマシングループ(プール内のマシン)の性能をモニタリングし、必要に応じてマシンの追加または削除を行います。その後、トラフィックマネージャに対して、そのクラウドプロバイダの API を使用してマシンを追加または削除したことを通知します。トラフィックマネージャは、この変更に関する通知を受け取ると、それに従ってオートスケーリングプールのノードを追加または削除します。 クラウドプロバイダのオートスケーリングメカニズムを使用する場合は、このパラメータを yes に設定してください。トラフィックマネージャのオートスケーリングメカニズムを使用する場合は、no に設定してください。

設定	説明
クラウドクレデンシヤル	クラウドの API が必要とする一連のクレデンシヤルです。一連のクラウドクレデンシヤルは、 Catalogs (カタログ) > Cloud Credentials (クラウドクレデンシヤル) ページであらかじめ設定することができます。

トラフィックマネージャは、クラウドプロバイダの必要条件に適用できる一連の設定キーを提示するために、ここで指定される API タイプのクレデンシヤルを使用します。

Amazon EC2 クラウドの設定

設定	説明
EC2 AMI	新しいノードインスタンスの作成元となる仮想マシンイメージの固有の識別子です。トラフィックマネージャは、処理能力を上げる必要があると判断すると、新たに作成された各ノードをプールに追加します。トラフィックマネージャは、クラウド内でこの AMI を使用するインスタンスを検出すると、それらのインスタンスが対応するプールに属するものと見なします。 ここで使用される固有の識別子によって、ノードで実行される OS とサービスが決まります。EC2 クラウドでは、このパラメータは ImageId と呼ばれています。
EC2 Machine Type	プールに追加されるノードのインスタンスに必要とされる、仮想マシンのリソースに関するサイズを示す識別子です。EC2 では、このパラメータは InstanceType と呼ばれています。 注: 通常、強力なマシンほど高いコストが伴います。
Name Prefix	オートスケーリングイベント中に作成される新規ノードの名前に追加される、オプションのプレフィックスです。 トラフィックマネージャは、クラウド内でこの名前で始まるインスタンスを検出すると、それらのインスタンスが対応するプールに属するものと見なします。

設定	説明
EC2 Security Group IDs	1つのセキュリティグループで、1つ以上のノードインスタンスに対するトラフィックをコントロールするファイアウォールとして機能します。あるインスタンスが起動されるときに、EC2はそのインスタンスを1つ以上のセキュリティグループに関連付けることができます。この関連付けを行うためには、セキュリティグループIDのリストを <i>Extra Arguments</i> (余剰引数)テキストボックスで指定します(グループ名は使用しません)。EC2-Classicのデプロイでは、EC2-Classic 専用で作成したセキュリティグループを使用する必要があります。同様に、EC2-VPC では、VPC 専用で作成したセキュリティグループを使用する必要があります。ここでセキュリティグループが指定されていない場合、EC2 はデフォルトのセキュリティグループを適用します。
EC2-VPC Subnet IDs	新たな EC2-VPC インスタンスが起動された場合の、VPC サブネットの ID のリストです。インスタンスは、<i>Extra Arguments</i> (余剰引数)テキストボックスで指定したサブネットに均等に分配されます。EC2-Classic 内部でインスタンスを起動するには、このリストを空の状態のままにします。 オートスケーリングされた EC2-VPC ノードを可用性の異なるゾーン全体で均等に分配する場合は、可用性の異なるゾーンに存在するサブネットのリストを作成することをお奨めします。
Extra RunInstances Arguments	AWS API 機能の <code>RunInstances</code> に対するキー/値の引数をカンマで区切ったリストで、ノード作成時に使用されます。 <code>RunInstances</code> については AWS に関するドキュメントを参照し、利用可能な引数のリストを確認してください。

Rackspace クラウドの設定

設定	説明
Image ID	新しいノードインスタンスの作成元となる仮想マシンイメージの固有の内部識別子です。トラフィックマネージャは、処理能力を上げる必要があると判断すると、新たに作成された各ノードをプールに追加します。ここで使用される固有の識別子によって、ノードで実行される OS とサービスが決まります。Rackspace クラウドでは、このパラメータは imageId と呼ばれています。 重要: ここでは、内部イメージ識別子を使用する必要があります。 特定のクラウドクレデンシャルセットで使用可能なイメージ識別子のリストを取得するには、以下のように組み込みの <code>rackspace.pl</code> スクリプトを手動でコマンドラインから実行します。 <pre>\$ZEUSHOME/zxtm/bin/rackspace.pl \ listimageids \ --cloudcreds=<credentials></pre> <credentials>の部分を Rackspace クラウドのクレデンシャルに置き換えます。

設定	説明
Flavor ID	プールに追加されるノードのインスタンスに必要とされる、仮想マシンのリソースに関するサイズを示す内部識別子です。Rackspace では、このパラメータは FlavorId と呼ばれています。 注: 通常、強力なマシンほど高いコストが伴います。 特定のクラウドクレデンシャルセットで使用可能なサイズ識別子のリストを取得するには、以下のよう組み込みの rackspace.pl スクリプトを手動でコマンドラインから実行します。 <pre>\$ZEUSHOME/zxtm/bin/rackspace.pl \ listsizeids \ --cloudcreds=<credentials></pre> <credentials>の部分を Rackspace クラウドのクレデンシャルに置き換えます。
Name Prefix	オプションのネームプレフィックスで、オートスケーリングイベントで作成されたクラウドでホストされる各仮想マシンに適用できます。トラフィックマネージャは、クラウド内でこの名前で始まるインスタンスを検出すると、それらのインスタンスが対応するプールに属するものと見なします。

VMware vSphere の設定

設定	説明
VMware Template	組み込み式 vSphere ドライバスクリプトは、既存の仮想マシン (VM) テンプレートを基に VM をデプロイして、新しい VM を作成します。これらのテンプレートはあらかじめ設定されている VM で、新しい VM のひな形として使用できるテンプレートであることを示すマークが付けられています。一般に、vCenter Server 上のテンプレートのパスは以下の形式です。 <DatacenterName>/vm/<TemplateName>
Name Prefix	vCenter Server では、新たに作成されるノードにはこの Name Prefix (ネームプレフィックス) が付けられます。同一の vSphere インフラストラクチャ上で複数のプールがオートスケーリングされる場合、各プールの Name Prefix (ネームプレフィックス) がユニークになるよう注意してください。
Data Center	vCenter Server で管理される全ての ESX ホストは、論理的な Data Center (データセンタ) に整理されます。オートスケーリングは、この Data Center (データセンタ) 内のホストまたはクラスタに対して実行されます。

設定	説明
Data Store	新たに作成された仮想マシンで使用される Data Store (データストア)です。空白のままにすると、(vCenter の設定に従って)その Data Center (データセンタ)のデフォルトデータストアが使用されます。 Data Store (データストア)をここで指定する場合、VM テンプレートで同じ Data Store (データストア)が使用されていることを確認してください。
VMware ESX Cluster	通常、vCenter Server 上のインフラストラクチャの最上位に Data Center (データセンタ)があり、ここに ESX ホストまたは ESX ホストのクラスタが保持されます。クラスタに Dynamic Resource Scheduling (DRS)が設定されている場合、新しい仮想マシンの配置先ホストは vCenter によって決められます。クラスタが構成されていない場合、この設定で、新しい VM を作成する ESX ホスト名または IP を指定する必要があります。

ノードの設定

設定	説明
autoscale!ipstouse	(使用 IP アドレス)一般に、クラウド環境内のインスタンスでは、同じクラウド内の他のノードと通信するためのプライベートネットワークの IP アドレスと、インターネットと通信するためのパブリックネットワークの IP アドレスの、少なくとも 2 つの IP アドレスが使用されます。「private」または「public」を選択し、トラフィックを新たなノードのプライベートアドレスに送信するか、パブリックアドレスに送信するかを定義します。 デフォルトでは、Amazon EC2-VPC 内のインスタンスにパブリック IP アドレスはありません。作成時に IP アドレスを割り当てるには、「public」を選択してください。

設定	説明
<code>autoscale!port</code>	ノードをプールに追加するときに使用するポートを定義します。このポートは、クラウド環境内に作成されたバックエンドマシンでサービスを監視するポートに対応します。
<code>autoscale!min_nodes</code>	オートスケーリングプールに存在できる最小ノード数を定義します。このノード数には、現在障害発生としてマークされているノードも含まれます。このような状況におけるノードの障害は、レスポンス時間の低下につながる可能性があり、その結果、オートスケーリングによって新しいノードが組み込まれます。ただし、この動作は保証されません。障害が発生したノードが含まれるオートスケーリングプールは、 <code>autoscale!max_nodes</code> で指定した数を超えてオートスケーリングされません。
<code>autoscale!max_nodes</code>	オートスケーリングプールに存在できる最大ノード数を定義します。これにより、プールは莫大な数のノードにオートスケーリングできなくなります。

スケーリングパラメータ

設定	説明
<code>autoscale!response_time</code>	トラフィックマネージャは、オートスケーリングプール内のノードのレスポンス時間(ミリ秒単位)をモニタリングします。この時間は、トラフィックマネージャがそのノードへのコネクションを開始してからレスポンスの最初のバイトを受信するまでの時間として測定されます。 <code>autoscale!response_time</code> 未満のレスポンス時間は適合していると見なされ、これより長い時間は不適合と見なされます。トラフィックマネージャは、適合しているコネクションの割合を継続的に測定します。

設定	説明
<code>autoscale!scaledown_level</code>	適合しているレスポンスの割合が <code>autoscale!hysteresis</code> 秒以上にわたってこの値を上回ると、プールはスケールダウン(すなわち、プールからノードが削除)されます。
<code>autoscale!scaleup_level</code>	適合しているレスポンスの割合が <code>autoscale!hysteresis</code> 秒以上にわたってこの値を下回ると、プールはスケールアップ(すなわち、プールにノードが追加)されます。
<code>autoscale!refractory</code>	プールサイズの変更(増加または減少)後、トラフィックマネージャは、 <code>autoscale!refractory</code> 秒待機してから、次の変更を行います。これは、新しいノードがプール全体のレスポンス時間に反映されるまで時間がかかるためです。一度に多数のノードが作成(または廃止)されないようにするために、この時間を増加させることが必要な場合があります。
<code>autoscale!hysteresis</code>	トラフィックマネージャが変更を開始する前に、オートスケーリング条件が満たされなければならない時間(秒単位)です。これは、(例えば)スケールアップイベントを発生させるために、プールの適合している接続の割合が <code>autoscale!scaleup_level</code> を下回った状態が継続しなければならない時間に相当します。
<code>autoscale!lastnode_idletime</code>	オートスケーリングされたプール内の最後のノードが、廃止される前にアイドル状態でなければならない時間(秒単位)です。 注: これは、 <code>autoscale!min_nodes</code> が 0 に設定されている場合にのみ該当します。

DNS を用いたオートスケーリング

トラフィックマネージャは、DNS レコードを使用してノード数を自動的に増減することにより、トラフィックマネージャによって処理されるトラフィックのリクエス

トに応えることができます。このメカニズムを DNS を用いたオートスケーリングと言います。

ホスト名に対する DNS レコードが複数の IP アドレスに名前解決され、アドレス数が有効な名前解決に応じて変化する場合、その変化をプール内で使用されるノードに反映させることができます。例えば、**backend.example.com** は最初に 4 つの IPv4 レコードに名前解決され、続いて 6 つの IPv4 レコードに名前解決されます。トラフィックマネージャは、新たな IP アドレスが含まれるように、プール内のノードリストを更新します。

DNS を用いたオートスケーリングの設定は、以下に従って行います。

設定	説明
<code>dns_autoscale!enabled</code>	プールに対する DNS を用いたオートスケーリングを有効化するには、このキーを Yes に設定します。 デフォルトでは、プールはオートスケーリングされません。
<code>dns_autoscale!hostnames</code>	トラフィックマネージャが定期的にクエリする DNS ホスト名のリストです。IPv4 A または IPv6 AAAA のレコードは全て、このプールのノードリストに対して追加または削除されます。
<code>dns_autoscale!port</code>	DNS を用いたオートスケーリングを使用して追加された全てのノードにトラフィックマネージャが追加したポート番号です。

CHAPTER 6 トラフィック IP グループとフォールトトレランス

トラフィックマネージャのクラスタでは、受信したネットワークトラフィックをトラフィックマネージャに分配し、いずれかのトラフィックマネージャで障害が発生した場合は、そのトラフィックマネージャに割り当てられたトラフィックを別のトラフィックマネージャに転送することができます。トラフィックの分配は、トラフィック IP グループを使用して設定します。

本章で説明する機能は、ご使用の製品プラットフォームによってわずかに異なる場合があります。詳細および製品の仕様については、Brocade の Web サイト (<http://www.brocade.com>) を参照してください。

フォールトトレランス

トラフィック IP アドレスとトラフィック IP グループ

トラフィック IP アドレスは、高い可用性を維持しなければならない IP アドレスです。これらのトラフィック IP アドレスは、トラフィック IP グループと呼ばれるグループに割り当てられます。

各トラフィック IP グループは、クラスタ内の一部(または全て)のトラフィックマネージャで管理されます。トラフィックマネージャは連携してトラフィックをトラフィックマネージャ間で分配し、トラフィック IP アドレスに送信された全てのトラフィックがクラスタ内のいずれかのトラフィックマネージャに送信され、管理されるようにします。

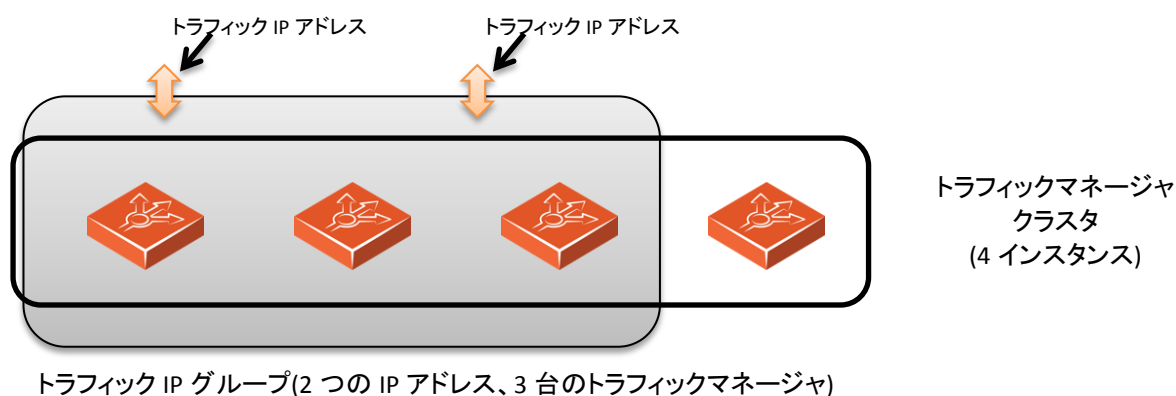


Fig 17. トラフィック IP アドレスの設定例

上図では、クラスタ内の 4 台のトラフィックマネージャのうち 3 台にまたがるようトラフィック IP グループが設定されています。このトラフィック IP グループはト

ラフィック IP アドレスを管理し、3 台のトラフィックマネージャによって、これらのトラフィック IP アドレスが利用可能な状態に保たれます。

例えば、Web サービスを IP アドレス 34.56.78.90 で公開しているとします。この場合、その IP アドレスをトラフィック IP グループに追加することにより、サービスを常に利用可能な状態に保つことができます。IP アドレスを追加すると、トラフィックマネージャクラスタでその IP アドレスが有効になり、この IP アドレスに送信される全てのトラフィックが管理されます。通常、1 つ以上のトラフィック IP アドレスに名前解決されるよう、サービスに DNS エントリを設定します。

トラフィック IP グループ内でのトラフィックの分配

トラフィックは、以下のいずれかの分散ポリシーに従ってクラスタ内のトラフィックマネージャ間で分配されます。

- **シングルホストモード:** 各トラフィック IP アドレスは、グループ内のいずれかのトラフィックマネージャで有効になります。グループが複数の IP アドレスで構成されている場合、トラフィックができる限り均等に分配されるように各 IP アドレスが別々のトラフィックマネージャで有効になります。
- **マルチホストモード:** 各トラフィック IP アドレスは、グループ内の全てのトラフィックマネージャで有効になります。各 IP アドレス宛てのトラフィックは、全てのトラフィックマネージャに均等に分配されます。
- **ルートヘルスインジェクション:** (一部の種類の製品は該当せず) 各 RHI トラフィック IP グループにアクティブなトラフィックマネージャが 1 つずつ、オプションでパッシブなトラフィックマネージャが 1 つずつ含まれます。グループ内の全てのトラフィック IP アドレスは、加えられている両方のトラフィックマネージャによって **private** に(ループバックで)有効になり、ルーティングプロトコルとして **OSPFv2** または **BGP** を使用して、隣接したルーティングドメインに動的にアドバタイズされます。レスポンスでは、ルータはアクティブなトラフィックマネージャに全てのトラフィックを送信します。

各トラフィックマネージャは定期的に他のトラフィックマネージャにレポートを送信し、いずれかのトラフィックマネージャで障害が発生した場合は、そのトラフィックマネージャに割り当てられたトラフィックを残りのトラフィックマネージャが引き継ぎます。これにより、トラフィック IP アドレスを使用するサービスは常に利用可能な状態に保たれます。

トラフィック IP グループの設定例については、39 ページの「構成例」を参照してください。トラフィックマネージャクラスタの作成方法については、59 ページの「クラスタの作成」を参照してください。

トラフィック IP アドレスの選択

通常、トラフィックマネージャには、受信トラフィック用にはフロントエンドネットワークが、バックエンドサーバ用にはバックエンドネットワークが追加されます。IP レベルでは、これらのネットワークは IP サブネットとして定義されます。例えば、192.0.2.0/24、198.51.100.0/16 などになります。

シングルホストおよびマルチホストトラフィック IP グループでは、これらの直接追加されたサブネット内にあるトラフィック IP アドレスを選択する必要があります。適切な物理インターフェースにあるこれらの IP アドレスがトラフィックマネージャで有効になることによって、前述のネットワークにあるルータやその他のデバイスがそれらの IP アドレスに直接アクセスできるようになります。

RHI トラフィック IP グループでは、これらの直接追加されたサブネット内にないトラフィック IP アドレスを選択する必要があります。RHI トラフィック IP アドレスがトラフィックマネージャで有効になるときは、必ずトラフィックマネージャの内部ループバック(lo)インターフェースにおいて、RHI トラフィック IP アドレス自体の/32 サブネットで *private* に有効になります。ルータやその他のマシンは、常に OSPFv2 ルーティングプロトコルを使用して確立されたルーティング情報に従って、トラフィックマネージャのフロントエンド IP アドレスを経由してそれらの IP アドレスに間接的にトラフィックを送信します。

直接追加されたサブネットとは、以下の条件に該当する場合を指します。

- 特定のインターフェースで静的に有効になる IP アドレスを設定するときに指定する。
- 「トラフィック IP ネットワーク」テーブルにエントリを追加することによって指定する。詳細については、116 ページの「インターフェースとサブネットのマッピング(トラフィック IP ネットワーク)」を参照してください。

トラフィック IP グループの作成

トラフィック IP グループを作成するには、**Services (サービス) > Traffic IP Groups (トラフィック IP グループ)** をクリックします。このページでは、トラフィック IP グループを新規作成したり、既存のトラフィック IP グループを編集したりすることができます。

グループのメンバにするトラフィックマネージャの名前を入力し、選択します。リストに、グループのトラフィック IP アドレスをスペースまたはカンマで区切って入力し、必要に応じて IP 分散モードを選択したうえで、**Create Traffic IP Group (トラフィック IP グループの作成)** をクリックします。

トラフィックの分配

同じトラフィック IP グループ内のトラフィックマネージャ間でどのようにトラフィックを分配するかは、IP 分散モードで決まります。

トラフィック IP グループは、トラフィック IP アドレスを一度に 1 台のトラフィックマネージャで有効にするか(シングルホストモード)、グループ内の全てのトラフィックマネージャで同時に有効にする(マルチホストモード)ように設定できます。一部の種類の製品では、ルートヘルスインジェクションも分散モードに選択できます。

シングルホストモード

グループ内の各 IP アドレスは、一度に 1 台のトラフィックマネージャでのみ有効になります。グループに複数の IP アドレスが含まれている場合、各 IP アドレスは加えられているトラフィックマネージャ全体で分散されます。1 つのトラフィックマネージャで障害が発生すると、そのトラフィック IP アドレスが、グループ内で稼働しているその他のトラフィックマネージャに引き渡されます。

トラフィック IP グループに **kepttogether** オプションを設定することにより、そのグループ内での IP アドレスの分散を防止することができます。その場合、全てのトラフィック IP アドレスが同じトラフィックマネージャで有効になります。この機能は、トラフィック IP グループと IP トランスペアレンシを組み合わせる場合に便利です。詳細については、43 ページの「クラスタでの IP トランスペアレンシの使用」を参照してください。

マルチホストモード

各 IP アドレスが同時に全てのマシンで有効になります。各 IP に送信されるデータは、グループ内の全てのトラフィックマネージャで受信され、各トラフィックマネージャは、受信したコネクションの等しい配分を担当します。マシンで障害が発生した場合、そのマシンで担当していた配分が残りのアクティブなマシンに分配されます。

マルチホスト IP アドレスは、ローカルネットワーク上でマルチキャストパケットを使用して動作します。トラフィックマネージャは、ローカルサブネットのマルチキャスト MAC アドレスを使用してトラフィック IP アドレスをアドバタイズします。これにより、スイッチおよびルータに、その IP へのトラフィックを全てのトラフィックマネージャに送信しなければならないことが示されます。使用される MAC アドレスは、別のマルチキャストグループ(239.0.0.0~239.255.255.255 の範囲の IP として表される)を提供することにより計算されます。このマルチキャストグループは、ネットワーク上でユニークである必要があります。

各トラフィックマネージャは、受信した全てのコネクションを受け取り、コネクションの送信元 IP アドレスに基づく計算を行い、そのコネクションを受け入れるか、(グループ内の別のトラフィックマネージャが受け入れるため)そのまま破棄するかを決定します。

この方法では、送信元 IP アドレスのみによって判断が行われるため、(クライアントが IP アドレスを変更しない限り)同じクライアントは同じトラフィックマネージャで処理されることになります。これには、トラフィックマネージャに保存されているセッション維持データと SSL セッションデータを利用できるというメリットがあります。

ただし、特にクライアントの送信元 IP アドレスが少数である場合に、トラフィックマネージャに分配されるトラフィックが不均等になる可能性があります。不均等に分配されると、負荷分散が行われるサービスの性能に影響を与えることがあります。このような状況が懸念される場合は、**consider source port** (ソースポート考慮)を設定し、送信元 IP アドレスとポートに基づいてトラフィックを分配するようトラフィックマネージャに指示します。これにより、トラフィックがクラスタ全体に容易に分配されるようになります。

セッション維持データをクラスタ全体で共有するときに多少の遅延が生じることがあるため、ソースポートを使用すると、一部のセッション維持手法の妨げになることがあります(CHAPTER 12 の「セッション維持とマルチホストトラフィック IP アドレスの併用」セクションを参照)。同様に、**consider source port** (ソースポート考慮)は、FTP 仮想サーバが使用しているトラフィック IP アドレスでは使用しないでください(CHAPTER 11 の「FTP」セクションを参照)。

注: トラフィックマネージャがマルチホスト IP に接続しようとする、そのセッションは常にローカルトラフィックマネージャ自体が受け取ります。

ルートヘルスインジェクション

トラフィックマネージャは、グループで管理されるトラフィック IP アドレスを広範なルーティングネットワーク全体に知らせることにより、フロントエンドマシンのフォールトトレランスを使用します。この手法を使用するためには、組織内のルータが次のいずれかのルーティングプロトコルを使用するように設定されている必要があります。

- **OSPFv2:** Open Shortest Path First バージョン 2
- **BGP:** Border Gateway Protocol

RHI は IPv4 アドレスにのみ使用できます。IPv6 はサポートされていません。

ルートヘルスインジェクション(RHI)に基づくトラフィック IP グループには、いずれも単一のアクティブなトラフィックマネージャ、またはアクティブ-パッシブ構成を使用したペアのトラフィックマネージャで構成される、最大 2 台のトラフィックマネージャが含まれます。

さらに、分散モードに RHI を選択する場合は、アクティブおよびパッシブなトラフィックマネージャに対してルーティングメトリクスを指定します。トラフィックマネージャがアップストリームのルーティングネットワークにルートのインジェクションを行う場合、これらのメトリクスによってマルチクラスタ(またはマルチデータ

センタ)シナリオ内におけるトラフィックマネージャのインスタンスの優先順位が決定します。

詳細については、46 ページの「ルートヘルスインジェクション」を参照してください。

パッシブマシン

トラフィックマネージャに「パッシブ」のマークを付けると、そのトラフィックマネージャでは、パッシブでないいずれかのトラフィックマネージャで障害が発生するまで、トラフィック IP グループ内のどの IP アドレスも有効にならないか(シングルホストモード)、何の負荷も処理されません(マルチホストモード)。

トラフィックマネージャを既存のトラフィック IP グループに追加すると、トラフィックの一部が新しいトラフィックマネージャに転送されることがあります。これにより、転送の瞬間に必然的に一部のコネクションが切断されます。このような状況は、トラフィックマネージャを「パッシブ」として追加することで対処できます。この場合、いずれかのピアで障害が発生するまで、そのトラフィックマネージャはトラフィックを全く受信しません。

トラフィック IP グループの無効化

トラフィック IP グループを無効化するには、**enabled** (有効化)チェックボックスの選択を解除します。このオプションは、グループを一時的に無効化しつつも、その後元の状態に戻すために設定を維持する場合に使用します。

グループが無効化されているときは、そのグループのどの IP アドレスも有効になりません。RHI トラフィック IP グループに対するルートアドバタイズメントは取り消されます。

インターフェースとサブネットのマッピング(トラフィック IP ネットワーク)

トラフィックマネージャのバージョン 7.0 までは、トラフィック IP アドレスのホストについて、インターフェースに同じサブネットの IP アドレスが設定されていることが前提条件となっていました。この前提条件には、トラフィックマネージャが(ルーティングテーブルとインターフェースの「ヘルス」に基づいて)どのデバイスがトラフィック IP アドレスをホストできるかを自動的に判断できるというメリットがあります。その一方、最終的に、実際には何のトラフィックも受信しないのに容量を占有する IP アドレスになるという欠点があります。トラフィックを受信させたいネットワークが小規模な場合(/29 など)、このことがストレスの原因になります。インターフェースとサブネットのマッピング機能を使用すれば、サブネットとインターフェースの明示的なマッピングを定義することができます。この機能を使用すると、

インターフェースに特定サブネットの IP アドレスを設定しなくても、そのサブネット内でトラフィック IP アドレスをホストすることができます。

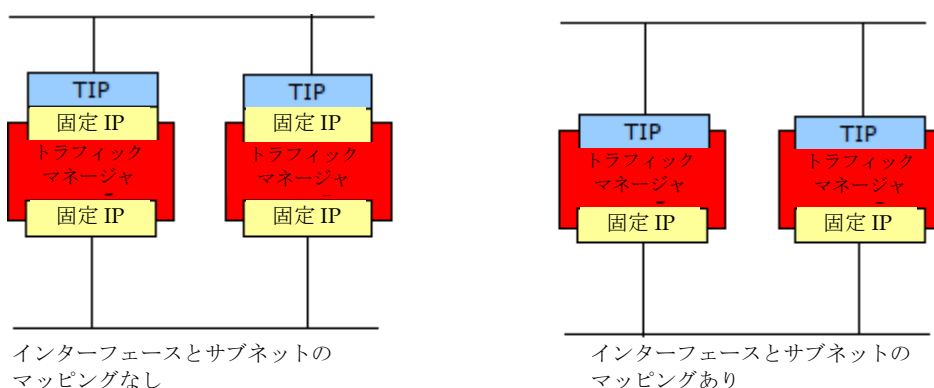


Fig 18. インターフェースとサブネットのマッピング

トラフィック IP ネットワークを設定するには、以下の手順に従います。

1. **Services** (サービス) > **Traffic IP Groups** (トラフィック IP グループ) ページに移動します。
2. **Traffic IP Networks** (トラフィック IP ネットワーク) セクションで、**Network Settings** (ネットワーク設定) リンクをクリックします。
3. **Add Network** (ネットワークの追加) フィールドに、IP アドレスをホストする IP サブネットを **CIDR 表記** で入力します(192.168.50.0/24 など)。
4. **Default Interface** (デフォルトインターフェース) ドロップダウンボックスから、このサブネットをホストするネットワークインターフェースを選択します。
5. **Update** (更新) をクリックします。

目的のインターフェースにトラフィック IP サブネットを設定すると、クラスタ内のその他のトラフィックマネージャで、同じサブネットに使用するインターフェースを設定することのできるダイアログボックスが表示されます。

フォールトトレランスの設定

クラスタ内のトラフィックマネージャは、アクティブな各ネットワークインターフェースを通して ICMP Ping を使用し、ネットワークと通信できることを定期的に確認します。この確認後、それぞれのヘルス状態(good (良好) または failed (障害発生)) を記述したメッセージをブロードキャストします。

クラスタ内のいずれかのトラフィックマネージャで障害が発生すると、その他のトラフィックマネージャが、障害が発生したトラフィックマネージャで管理されていた全てのトラフィック IP アドレスを引き継ぎます。

フォールトトレランスの設定

フォールトトレランスは、**System** (システム) > **Fault Tolerance** (フォールトトレランス) > **General** (一般) で設定することができます。

- **flipper!autofailback** - トラフィック IP アドレスをホストしているトラフィックマネージャで障害が発生すると、そのトラフィックマネージャが復旧するまで、破棄されたアドレスはクラスタ内の別のトラフィックマネージャで有効になります。

元のトラフィックマネージャのコネクションが復旧した時点で、トラフィック IP アドレスが自動的にそのトラフィックマネージャへ返されるようにするには、**Yes** に設定します。

No に設定すると、復旧に合わせて元のトラフィックマネージャは強制的に **保留** モードになります。このモードでは、トラフィック IP アドレスをいつ有効にするかについて指示されるまで、トラフィックマネージャは待機します。

トラフィックマネージャが保留状態にあると、管理 UI に警告が表示されます。保留モードのトラフィックマネージャを完全にアクティブな状態に戻すには、**Diagnose** (診断) > **Cluster Diagnosis** (クラスタ診断) ページ、または **Services** (サービス) > **Traffic IP Groups** (トラフィック IP グループ) ページを使用します。

注: クラスタ内のその他全てのトラフィックマネージャで障害が発生すると、保留モードのトラフィックマネージャは自動的にアクティブな状態に戻ります。この場合、アクティブな状態に戻ったトラフィックマネージャは、それ自体が管理している全てのトラフィック IP アドレスを引き継ぎます。また、既に実行中のサービスを中断させることがないトラフィックマネージャも、自動的にアクティブな状態に戻ります。

- **flipper!monitor_interval**: トラフィックマネージャが、ネットワーク上のその他のデバイスと通信できるかどうかを定期的に確認するために使用する時間間隔 (ミリ秒単位) です。確認結果は、マルチキャストアドレスに存在するその他のトラフィックマネージャに通知されます。間隔を短くすると、ネットワーク内の障害がより早く検出されるようになりますが、それによってシステムへの負荷は高まります。
- **flipper!monitor_timeout**: トラフィックマネージャが、コネクションを確認するために使用するデバイスの 1 つと通信を試みているときに、トラフィックマネージャにレスポンスを待機させる時間の長さ (秒単位) です。この時間内にレスポンスが受信されなかった場合、トラフィックマネージャは、そのデバイスとの通信は不可能であると見なします。この時間を短くすると、トラフィックマネージャ

による障害の検出が早くなりますが、低速のコネクションが障害であると誤って判断されることがあるため、システムの信頼性は低下します。

- **flipper!heartbeat_method:** クラスタ内では、各トラフィックマネージャは定期的なハートビートメッセージを送信し、全てのソフトウェアおよびネットワークサービスが稼働していることを通知しなければなりません。

トラフィックマネージャに対し、送信相手を指定した UDP メッセージをクラスタ内の各トラフィックマネージャに送信するように指示するには、ユニキャストを使用します。ネットワーク内のスイッチが、より効率的になり得るマルチキャストパケットの手法を処理できない場合は、この設定を使用してください。ユニキャストメッセージは、全く異なるデータセンタにあるトラフィックマネージャなど、別のサブネットに存在するマシンにも到達します。ユニキャストメッセージに対してトラフィックマネージャが監視する必要があるネットワークポートを指定するには、**Communication Port:** (通信ポート)を使用します。このポートは、クラスタ内のその他全てのトラフィックマネージャから通信できなければなりません。

トラフィックマネージャが接続されている全てのスイッチの全てのネットワークポートをトラバースするマルチキャストネットワークパケットを使用して、これらのメッセージを送信するようトラフィックマネージャに指示するには、マルチキャストを使用します。マルチキャストメッセージは、同じサブネットに存在するマシンにのみ到達します。**Multicast address and port:** (マルチキャストアドレスおよびポート)フィールドでは、全てのトラフィックマネージャに監視させるマルチキャストアドレスを設定してください(デフォルトでは 239.100.1.1:9090)。

- **flipper!use_bindip:** マルチキャストメッセージの送受信を管理インターフェースだけで行うかどうかを設定します。

トラフィックマネージャが、メイントラフィックに使用される別々の専用ネットワーク上で相互に通信する場合は、**Yes** に設定します。ただし、結果としてシステムの信頼性が損なわれる可能性があることに注意してください。

トラフィックマネージャに対し、全てのネットワークインターフェースを経由して接続性を通知するために使用しているマルチキャストメッセージを送信するように指示するには、**No** に設定します(デフォルト)。**No** に設定すると、1つのネットワークインターフェースで障害が発生しても、クラスタ内のその他のトラフィックマネージャはそのマシンで障害が発生しているとは見なすことはないため、システムの信頼性が高まります。

- **flipper!arp_count:** ネットワークインターフェースが有効なときにトラフィックマネージャが送信する ARP パケットの数です。トラフィックマネージャの新たなインターフェースを認識するために、ネットワーク上のその他のデバイスは ARP リクエストを 1 つだけ必要としますが、ARP パケットが行方不明になったり、失われたりすることがあります。インターフェースが有効な場合、トラフィ

ックマネージャはここで定義された数の ARP パケットを送信し、ブロードキャストの信頼性と速度を向上させます。

- **flipper!igmp_interval:** トラフィックマネージャが、マルチホストトラフィック IP グループに対する IGMP メンバーシップレポートパケットを一方的に送信するために使用する時間間隔(秒単位)です。この間隔を使用すると、スイッチおよびルータにトラフィックマネージャのマルチキャストサブスクリプションを忘れずに実行させることができます。この機能を無効化するには、間隔を 0 に設定してください。
- **flipper!verbose:** トラフィックマネージャに全ての接続テストを記録するかどうかを設定します。システムログに詳細情報を出力するには、Yes に設定します。

注: この設定は、診断目的のみに使用することをお奨めします。

- **flipper!frontend_check_addrs:** フロントエンドのコネクションを確認するために使用する IP アドレスです。トラフィックマネージャがこれらのアドレスに対する Ping の実行を適切に完了できない場合、トラフィックマネージャはそのネットワークのコネクションが切断されていると判断し、クラスタ内のその他のトラフィックマネージャにフェールオーバーします。

トラフィックマネージャがイントラネットの一部である場合など、トラフィックマネージャが外部接続を必要としないように設定されている場合、このフィールドを空白にすることで、外部接続の確認を無効化することができます。

トラフィックマネージャのフォールトトレランスチェック

各トラフィックマネージャは、それ自体のネットワークインターフェースが正常に動作しているかどうかをチェックします。このチェックは以下のように行われます。

- デフォルトゲートウェイに対して定期的に Ping を実行し、フロントエンドネットワークインターフェースが機能していることを確認します。
- 使用中のプール内の各バックエンドノードに対して定期的に Ping を実行し、バックエンドネットワークインターフェースが機能していることを確認します。

トラフィックマネージャは、デフォルトゲートウェイに対する Ping が失敗した場合、または現在使用しているどのプールのどのノードに対する Ping も失敗した場合、そのトラフィックマネージャ自体に障害が発生していると判断します。

フロントエンドチェックのオーバーライド

例えば、ゲートウェイが ICMP Ping に対してレスポンスを返さない場合などに、フロントエンドチェックのオーバーライドが必要になることがあります。フロントエンドチェックをオーバーライドするには、**flipper!frontend_check_addrs** を空の文字列に設定します。

ヘルスブロードキャスト

各トラフィックマネージャは、ローカルヘルスチェックの結果を、そのヘルス状態に関わらず定期的にブロードキャストします。

デフォルトでは、各マシンは、1 秒間に 2 回ハートビートメッセージをブロードキャストします。この動作は、**flipper!monitor_interval** で設定することができます。

ブロードキャストメッセージの送信方法は、マルチキャストとユニキャストから選択できます。**flipper!heartbeat_method** 設定でいずれかの方法を選択してください。

重要: ユニキャストを使用する場合、*State Sharing* (状態共有)を有効化する必要があります。これを行うには、**System** (システム) > **Global Settings** (グローバル設定) ページで **state_sync_time** をゼロ以外の値に設定します。

多くの状況では、ユニキャストが適しています。ただし、トラフィックマネージャクラスタが単一のネットワークセグメント内にある場合など、全てのトラフィックマネージャがマルチキャストメッセージを受信できるようにネットワークが設定されている場合は、代わりにマルチキャストを使用してください。両方の手法を試しても問題が解消されない場合は、サポート窓口に問い合わせて指示を受けるようにしてください。

注: Hyper-V ベースのトラフィックマネージャを使用している場合はデフォルトのユニキャスト手法を使用することをお奨めします。複数のインスタンスがまとめてクラスタ化されている場合、状況によってはマルチキャストを使用することでパケット損失につながる可能性があります。

ハートビート通信の問題をデバッグする場合は、tcpdump プログラムを使用してネットワーク上の全てのハートビートメッセージをキャプチャすることをお奨めします。

```
# tcpdump -i eth0 ip multicast
```

ご使用のトラフィックマネージャから送信されたハートビートメッセージのみをキャプチャするには、以下のプログラムを使用してください。

```
# tcpdump -i eth0 dst host 239.100.1.1 and dst port 9090
```

クラスタのヘルス判定

各トラフィックマネージャは、クラスタ内のその他全てのトラフィックマネージャから送信されたヘルスメッセージを監視し、以下のいずれかの条件を満たした場合にいずれかのピアで障害が発生したと判断します。

- ピアから「I have failed」ヘルスメッセージを受信した場合

- **flipper!monitor_timeout** で設定された値以内に、ピアから何のメッセージも受信しなかった場合
- ピアが、その子プロセスのいずれかでトラフィックが処理されなくなったことをレポートした場合。これは、いずれかの子プロセスが **flipper!child_timeout** で設定した値の時間内にレスポンスを返さない場合に発生します。

トラフィックマネージャは、ピアから「I am healthy」メッセージを受信し始めると、そのピアが復旧したと判断します。

フェールオーバー

クラスタ内の各トラフィックマネージャは、ピアのいずれかで障害が発生したと判断した場合、障害が発生したシステムに割り当てられていた一部または全てのトラフィックを引き継ぐことができます。これをどのように行うかは、トラフィック分散方法によって決まります。

トラフィック IP アドレストランスファ(シングルホストモード)

クラスタ内の各トラフィックマネージャは、どのマシンがアクティブかについて保持している情報に基づき、使用するトラフィック IP アドレスを判断します。クラスタは完全な決定的アルゴリズムによって、以下のように IP アドレスをマシンに分配します。

- このアルゴリズムは決定的であるため、いずれかのピアで障害が発生するか復旧したときに、トラフィックマネージャ相互でネゴシエーションを行う必要がありません。
- このアルゴリズムは、クラスタ内のアクティブなトラフィックマネージャにトラフィック IP アドレスを広く分散させ、トラフィックマネージャの障害または復旧時に IP アドレスの転送件数を最小限に抑えるよう最適化されています。

トラフィックマネージャでトラフィック IP アドレスが有効になると、そのトラフィックマネージャは、隣接するネットワークデバイスにその IP アドレスに対応する MAC アドレスが変更された可能性があることを通知する ARP メッセージをいくつか送信します。トラフィックマネージャは最大 10 件の ARP メッセージを送信し(**flipper!arp_count** 設定で調整可能)、これらのメッセージの送信頻度は **flipper!monitor_interval** 設定でコントロールできます(デフォルトでは 0.5 秒間隔でメッセージを送信)。

トラフィックマネージャは、それ自体のネットワークコネクションで障害が発生したことを検出すると、そのトラフィックマネージャで有効になっていたトラフィック IP アドレスを即座に無効にし、ピアに *I have failed* ヘルスメッセージをブロードキャストします。これは、クラスタ内のその他のトラフィックマネージャがこのトラ

フィックマネージャでの障害発生を認識し、そのインターフェースを有効にすることを期待して行われる動作です。

トラフィック IP アドレストランスファ(シングルホストモード)

トラフィック IP グループ内の各トラフィックマネージャは、パケットの送信元 IP (およびオプションでソースポート。詳細は 114 ページの「トラフィックの分配」を参照)に基づいて、各パケットを処理すべきかどうかを確定的に選択します。

トラフィックマネージャで障害が発生した場合、そのトラフィックマネージャに割り当てられていた負荷が残りのトラフィックマネージャに均等に分配されます。トラフィックマネージャは障害から復旧すると、負荷が均等になるようピアから負荷を受け入れるため、トラフィック IP グループ内の稼働しているマシンにトラフィックが常に均等に分散されます。

注: トラフィックマネージャソフトウェアには、マルチホスト IP 機能がデフォルトでは含まれていません。この機能は、追加カーネルモジュールとしてダウンロードしてインストールすることができ、Linux カーネルのバージョン 2.6.18 以上でサポートされます。サポートされているバージョンについての詳細は、Brocade の Web サイト(<http://www.brocade.com>)からダウンロードできるトラフィックマネージャのドキュメントを参照してください。

トラフィック IP アドレストランスファ(RHI モード)

トラフィックマネージャのフォールトトレランスチェックが失敗すると、トラフィックマネージャは RHI トラフィック IP グループで使用するアドレスを下げ、ネットワークのルートアドバタイズメントを取り下げます。

RHI トラフィック IP グループ内の専用のアクティブなトラフィックマネージャと通信不能であることがネットワークにより検出されると、専用のパッシブなトラフィックマネージャや、別のデータセンタ内で同じトラフィック IP アドレスをホストしているトラフィックマネージャなど、最も低いメトリクスを使用する、次に可用性の高いルートを代用してトラフィック IP アドレスに対するルーティングが決定されます。

障害からの復旧

障害の発生したトラフィックマネージャが復旧すると、そのトラフィックマネージャの割り当て分のトラフィックが他のトラフィックマネージャから転送されます。

分配されたトラフィックがトラフィックマネージャ間で転送されるたびに、そのトラフィックに使用されていたコネクションは切断されます。この動作は、トラフィックマネージャで障害が発生したために転送が行われる場合には避けられませんが、トラフィックマネージャの復旧時には望ましくない場合があります。

この場合、管理 UI の **System (システム) > Fault Tolerance (フォールトトレランス)** ページで **flipper!autofailback** 設定を無効化することができます。この設定を無効化すると、トラフィックマネージャは復旧時にトラフィックを受け入れません。代わりに、トラフィックマネージャが復旧し、IP アドレスを再び受け入れることができることを示すメッセージがユーザインターフェースに表示されます。

トラフィックマネージャを再びアクティブにするには、**Diagnose (診断)** ページに移動し、**Reactivate this Traffic Manager** (このトラフィックマネージャの再アクティブ化) リンクを選択します。

または、各トラフィック IP グループを編集し、復旧したトラフィックマネージャをパッシブに設定することもできます。トラフィックマネージャを全てのグループでパッシブに設定すると、そのトラフィックマネージャは分配されるトラフィックを受け入れる必要がなくなり、後にエラー状態が解除された状態で自動的に再度アクティブになります。また、分配されたトラフィックが転送されないため、トラフィックが失われることもありません。

フォールトトレランスのデバッグとモニタリング

全ての状態変化と IP アドレスの転送は、関連する各トラフィックマネージャのイベントログに記録されます。電子メールアラートが正しく設定されていれば (CHAPTER 21 を参照)、トラフィックマネージャは、状態変化と、その結果行われた IP アドレスの転送について記述した電子メールメッセージをシステム管理者に送信します。

詳細なデバッグを行うには、**flipper!verbose** 設定を有効化します。これにより、各トラフィックマネージャでは、接続テスト、送信したブロードキャストメッセージ、受信したブロードキャストメッセージが全て記録されるため、フォールトトレランスが期待どおりに動作しない原因を判断するときに役立ちます。

管理 UI の **Diagnose (診断)** セクションにある **Cluster Diagnosis (クラスタ診断)** ページも確認してください。このページには、正しく受信されていないブロードキャストメッセージがあるかどうかを示され、エラーが発生している場合はシステムステータスの概要が表示されます。

ルートヘルスインジェクションの状態に関する詳細は、管理 UI の **Diagnosis (診断) > Routing (ルーティング)** ページを参照してください。

BGP 接続性の設定

トラフィック IP グループは、ルーティングプロトコルとして BGP を使用して、IP 分散モードに RHI を使用するように設定することができます。トラフィック IP アドレスに対して RHI を有効化するには、**Services (サービス) > Traffic IP Groups (トラフィック IP グループ)** ページを使用します。

System (システム) > Fault Tolerance (フォールトトレランス) > BGP Route Health Injection (BGP ルートヘルスインジェクション)の設定によって、クラスタ内のトラフィックマネージャと、隣接するルーティングインフラストラクチャとの間の BGP 接続性が制御されます。

- **bgp!enabled:**BGP ルートヘルスインジェクションを有効化するかどうかを設定します。
- **bgp!as_number:**トラフィックマネージャクラスタが動作する自律システム(AS)です。10 進値を使用して、AS を指定します。この数値は、クラスタ内の全てのトラフィックマネージャ(またはマルチサイトマネージャクラスタ内の各ロケーション)設定。418 ページの「マルチサイトクラスタの管理」を参照してください)に適用されます。

BGP ルータ ID の設定

デフォルトでは、クラスタ内の各トラフィックマネージャは、それ自体の IPv4 デフォルトゲートウェイへの接続に使用されているアドレスおよびサブネットを使用して、BGP のルーティングドメインに組み込まれます。この動作をオーバーライドするには、**flipper!bgp_router_id** を使用して、個々のクラスタメンバに別の ID を設定します。

BGP ネイバーの管理

OSPFv2 とは異なり、トラフィックマネージャはその BGP ネイバーを自動的に検出できません。クラスタで BGP セッションを確立できるようにするには、初めに隣接するルータを定義してから、各クラスタメンバと対象ネイバー間に適用されるマッピングを作成する必要があります。

このマッピングで使用できる BGP ネイバーを管理するには、**Manage BGP Neighbors (BGP ネイバーの管理)**をクリックします。このページを使用して、新しい隣接する BGP 対応ルータを定義し、既存のネイバー定義のパラメータを変更します。

新しい各ルータに対して、以下の BGP コネクションパラメータを入力します。

フィールド	説明
BGP Neighbor name (BGP ネイバー名)	識別名
IP Address (IP アドレス)	ネイバーの IPv4 アドレス

フィールド	説明
AS Number (AS 番号)	このネイバーの AS 番号。これは、推奨される対応トラフィックマネージャと同じ AS にするか(セッションで iBGP が使用されます)、まったく異なる AS することができます(セッションで eBGP が使用されます)。
Keepalive (キープアライブ)	ヘルスメッセージ送信の間隔(秒単位)
Holdtime (ホールド時間)	最後に keepalive を受信してから、このネイバーで、マッピングされたトラフィックマネージャがアクティブではなくなったと見なされるまでの最小間隔(秒単位)
Advertisement Interval (アドバタイズメント間隔)	このネイバーに対して実行するアドバタイズメントの間隔(秒単位)
Authentication Password (認証パスワード)	(オプション)マッピングされたトラフィックマネージャが、このネイバーで使用する必要がある BGP ごとのセッションパスワード 注: 2.6.20 よりも前の Linux カーネルバージョンまたは関連オプション(CONFIG_TCP_MD5SIG)を有効化せずに構築されたカーネルでは、この設定がサポートされないことがあります。詳細については、システム管理者またはサポート窓口にお問い合わせください。

BGP ネイバーを定義した後に、**System (システム) > Fault Tolerance (フォールトトレランス)**をクリックしてこのページに戻り、個々のトラフィックマネージャに対するマッピングを作成します。

重要: BGP を使用する前に、隣接する BGP ルータを設定して、選択したトラフィックマネージャとのセッションを確立する必要があります。これは、実質的に、コネクションのトラフィックマネージャ側で最後に作成したネイバー定義に対応します。

Update (更新)をクリックしてマッピングと設定を適用します。このページの設定は、トラフィックマネージャクラスタの全てのメンバに複製されます。

トラフィックマネージャによって、全てのルータピアリングと接続失敗がイベントログにレポートされます。管理 UI のアプレットと **Diagnose (診断)**ページの両方に、現在のステータスが反映されます。

OSPFv2 の接続性の設定

トラフィック IP グループは、ルーティングプロトコルとして OSPFv2 を使用して、IP 分散モードに RHI を使用するように設定することができます。トラフィック IP アドレスに対して RHI を有効化するには、**Services** (サービス) > **Traffic IP Groups** (トラフィック IP グループ) ページを使用します。

System (システム) > **Fault Tolerance** (フォールトトレランス) > **OSPF Route Health Injection** (OSPF ルートヘルスインジェクション) の各設定により、トラフィックマネージャクラスタ内の OSPFv2 の接続性をコントロールできます。

- **ospfv2!enabled:** OSPFv2 ルートヘルスインジェクションを有効化するかどうかを設定します。
- **ospfv2!area:** トラフィックマネージャクラスタが動作する OSPFv2 エリアです。10 進フォーマットまたは IPv4 アドレスフォーマットを使用できます。
- **ospfv2!area_type:** トラフィックマネージャクラスタが動作する OSPFv2 エリアのタイプです。OSPFv2 では、特定エリア内の全てのルータが同じタイプを使用する必要があります。
- **ospfv2!hello_interval:** トラフィックマネージャが OSPFv2 の「Hello」パケットをネットワークに送信する間隔です。
- **ospfv2!router_dead_interval:** トラフィックマネージャがサイレントなルータダウンを宣言する前に経過しなければならない秒数です。

注: OSPFv2 を無効化したり、トラフィックマネージャの動作を停止させたりすると、隣接する全てのルータに **MaxAge** リンクステートアドバタイズメント(LSA)が送信されることによって、まずトラフィック IP アドレスのアドバタイズメントが停止されます。

トラフィックマネージャは、2 つの別個の OSPFv2 認証キー/シークレットのペア (「a」および「b」) の使用をサポートしています。このメカニズムによって、キー/シークレットの 1 つのペアを交換しなくななくなったときに、ネットワークインフラストラクチャ全体で認証を保持することができます。使用中の認証キーを更新するときは、新しいキーのプロビジョニングがネットワーク全体で完了するまで、古いキーをアクティブのまま保つことができます。

トラフィックマネージャは、0 (何ものなし) と 2 (MD5) のタイプの OSPFv2 認証をサポートしています。MD5 は、アクティブなキーが 1 つ以上設定されているときに使用されます。

- **ospfv2!authentication_key_id_a:** OSPFv2 認証のキー ID です。キーを無効化するには、0 に設定します。

- **ospfv2!authentication_shared_secret_a:** OSPFv2 認証の共有シークレット(MD5)です。キーを無効化するには、空白に設定します。
- **ospfv2!authentication_key_id_b:** OSPFv2 認証のキーID です。キーを無効化するには、0 に設定します。
- **ospfv2!authentication_shared_secret_b:** OSPFv2 認証の共有シークレット(MD5)です。キーを無効化するには、空白に設定します。

注: 両方のキーペアを 0 (無効)に設定すると、トラフィックマネージャは OSPFv2 認証を使用しません。

トラフィックマネージャは、汎用 OSPFv2 ルータとしては機能しなくなります。

OSPFv2 の IP アドレスの設定

デフォルトでは、クラスタ内の各トラフィックマネージャは、それ自体の IPv4 デフォルトゲートウェイへの接続に使用されているアドレスおよびサブネットを使用して、OSPFv2 のルーティングドメインに組み込まれます。この動作をオーバーライドするには、管理 UI の **flipper!ospfv2_ip** を使用して、個々のクラスタメンバに対して代替の IPv4 アドレスを設定します。

ここで指定する個々の IP アドレスは、各トラフィックマネージャによって、トラフィックマネージャが通知しているトラフィック IP 用の隣接するルータにインストールされている OSPFv2 ルートに対し、ネクストホップアドレスとして使用されます。

トラフィックマネージャのルーティングソフトウェアを無効化するには、**flipper!ospfv2_ip** を「0.0.0.0」に設定します。

隣接モニタリングの設定

トラフィックマネージャはルーティングネットワークのポーリングを 10 秒ごとに行い、隣接データを収集しています。このデータには、トラフィックマネージャがピアとして設定されているルータのリストが含まれています。OSPF の場合、このリストはネイバーリストと言います。

隣接モニタリングを実行するためには、トラフィックマネージャがトラフィックを受信することが想定される、隣接するルータの IP アドレスリストを使用して、クラスタ内の各トラフィックマネージャを設定します。このリストと発見されたネイバーリストが比較されることにより、ルーティングネットワークのヘルス判定が行われます。

想定されるネイバーを設定するには、スペースまたはカンマで区切られた IPv4 アドレスのリストを **flipper!ospfv2_neighbor_addrs** に入力します。各トラフィックマネージャが接続されているデフォルトの IPv4 ゲートウェイを指すには、デフォルト値の %gateway% を使用します。

特定のトラフィックマネージャに対する隣接モニタリングを無効化するには、**flipper!ospfv2_neighbor_addrs** を空白のリストに設定します。

想定されるルータの一部がピアに設定されていない場合、トラフィックマネージャはイベントログに警告を報告します。また、想定されるルータが 1 つもピアに設定されていない場合、トラフィックマネージャはエラーを報告します。管理 UI および Diagnose (診断) ページのアプレットには、どちらも最後のポーリング時の状態が反映されます。

注: OSPF の設定を変更すると、ピアの確立に伴う遅延により、警告またはエラーが一時的に表示されることがあります。

CHAPTER 7 トラフィックマネージャ管理インターフェースの主要機能

ホームページ

トラフィックマネージャ管理インターフェースのホームページには、主な設定、トラフィック、およびステータスの概要が動的に更新されて表示されます。

- **Status Applet** (ステータスアプレット)には、クラスタのヘルスステータス、トラフィックチャート、および概要がリアルタイムに表示されます。何らかの問題が検出されると、このアプレット内のステータスライトが緑色からオレンジ(重要性の低い問題)または赤(深刻な問題)に変化します。ステータスライトをクリックすると、**Diagnosis** (診断)ページに直接移動します。
- **Login Information Banner** (ログイン情報バナー)には、これらのクレデンシャルを使用して前回正常にログインしたときのレコードが表示されます。ここには、これまでに失敗したログイン試行も示されます。
- **Traffic Managers** (トラフィックマネージャ)セクションには、クラスタ内のトラフィック管理デバイスのステータスが表示されます。いずれかのデバイスで障害が発生すると、そのデバイスがハイライトされ、エラーメッセージが表示されます。
- **Services** (サービス)セクションには、管理しているサービスのリストと、各サービスで使用しているプロトコル、ポート、仮想サーバ名、および全プールのリストが表示されます³。サービスは、**Stop/Start** (停止/開始)ボタンで停止または開始することができます。仮想サーバまたはプールで何らかの問題が検出されると、該当する仮想サーバまたはプールがハイライトされます。
- **Event Log** (イベントログ)セクションには、クラスタ内のトラフィックマネージャシステムから送信された最近のイベントログメッセージが表示されます。**Examine Logs** (ログ確認)ボタンをクリックすると、イベントログ全体が表示されます。詳細については、396 ページの「イベントおよび監査ログ」を参照してください。

ホームページは自動的に更新され、新しいイベントログメッセージや、トラフィックマネージャ、仮想サーバ、またはプールのステータスの変化が示されます。

³ 注: リストには、設定で明示的にデフォルトプールとして参照されているプールまたは **TrafficScript** ルールや **RuleBuilder** ルールに設定されているプールのみが表示されます。例えば `pool.use($variable)` などを使用して動的に選択したプールは表示されません。

注: マルチクラスタ管理モードでは、設定されているロケーションが表示される *Locations* セクションなどが追加されるため、それに合わせてトラフィックマネージャのホームページのレイアウトが変わります。詳細については、CHAPTER 28 の「マルチサイトクラスタの管理」を参照してください。

Services (サービス) > Configuration Summary (設定概要)

多数の仮想サーバ、プール、およびルールを設定すると、設定はすぐに複雑になってしまいます。**Config Summary** (設定概要)には、設定した項目とその相互関係の概要が1ページにまとめて表示されます。

Config Summary (設定概要)を表示するには、**Services** (サービス)ボタンをクリックし、次に **Config Summary** (設定概要)タブをクリックします。このページには、サービスがテーブル形式で表示されます。初期状態では、サービスがポートおよび仮想サーバごとにグループ化され、各サービスの横にそのサービスで使用されている各項目が表示されます。

このテーブルは、特定の列や設定オブジェクトをキーにして並べ替え、その項目に依存する設定オブジェクトを分かりやすく表示することができます。例えば、**Pool** (プール)列をクリックすると、各プール名と、その横にそのプールに含まれるノード、そのプールを使用する全ての仮想サーバおよびルールが表示されます。

これにより、設定が複雑であっても、設定されている各項目がどこで使用されているかを正確にトラッキングできます。

カタログ

カタログは、トラフィックを管理するために使用できるオブジェクトのセントラルリポジトリです。以下のタイプのオブジェクトを保存できます。

注: 一部のカタログタイプは、ライセンスに応じて異なることがあります。詳細については、Brocade の Web サイト(<http://www.brocade.com>)を参照してください。

- **ロケーションカタログ:** グローバル負荷分散(GLB)、およびトラフィックマネージャのマルチサイトクラスタ管理機能に使用される、ロケーションベースの設定オブジェクトが保持されます。
- **DNS サーバ:** トラフィックマネージャの権威ドメインネームサーバ(DNS)機能で使用するゾーンファイルおよびゾーン設定が保持されます。
- **GLB サービスカタログ:** 地理的位置が異なるクライアントの負荷分散を行うためにトラフィックマネージャが使用する GLB サービス設定が保持されます。

- **Web Accelerator カタログ:** Brocade Web Accelerator によって使用されるアプリケーションのスコープとプロファイルが保持されます。
- **ルールカタログ:** TrafficScript および RuleBuilder ルールが保持されます。
- **Java Extensions カタログ:** Java Extensions と、Java Extensions が必要とする全てのサポート Java クラスファイルが保持されます。これらの Java Extensions は TrafficScript ルールから呼び出すことができます。
- **モニタカタログ:** プール内のノードが正常に動作していることを確認するためのヘルスマニタが保存されます。
- **SSL カタログ:** サーバおよびクライアント証明書、認証局、証明書失効リストなどの SSL リソースが保持されます。
- **認証コードカタログ:** リモート LDAP 認証サービスの定義が保持されます。これらのサービスに TrafficScript からアクセスして、ユーザに関する情報を検索したり、ユーザのパスワードを検証することができます。
- **Kerberos カタログ:** Kerberos のレムに加わるためにトラフィックマネージャが必要とする設定リソースが保持されます。
- **サービスプロテクションクラス:** トラフィックマネージャが不要なトラフィックをフィルタリングするためのポリシーおよびセキュリティ対策を定義するクラスが保持されます。
- **セッション維持クラス:** クライアントコネクションのセッション維持情報を管理するクラスが保持されます。
- **帯域クラス:** 帯域の使用を管理するための帯域割り当てが保持されます。
- **サービスレベルモニタリングクラス:** ノードのレスポンス時間と合意されたサービスレベルへの適合状況をモニタリングするクラスが保持されます。
- **レートシェイピングクラス:** 最大リクエストレートを実現するために、リクエストをキューイングおよびレートシェイピングするためのクラスが保持されます。
- **クラウドクレデンシャルクラス:** トラフィックマネージャがクラウドプロバイダ API と通信するために必要な情報が保持されます。
- **追加ファイル:** TrafficScript ルールからアクセスできる(resource.get())を使用)、またはプール設定でエラーメッセージを提供するために使用する追加のファイルとリソースが保持されます。

仮想サーバ、プール、およびルールはそれぞれカタログ内のオブジェクトを参照できます。カタログ内の項目を編集すると、その項目を使用している全てのサービスに変更内容が伝搬されます。

カタログにアクセスするには、トップナビゲーションバーにある **Catalogs** (カタログ) をクリックします。カタログ内の項目を編集または追加するには、該当する **Edit** (編集) リンクをクリックします。

各カタログタイプについては、後で詳しく説明します。

ライセンス管理

トラフィックマネージャのドキュメントと製品インターフェースは、ライセンスキーの使用を参照して、製品の機能を有効化または無効化することがあります。

Brocade(または指定されたサポート窓口)は、個別にライセンスキーを生成し、通常、ライセンスキーは特定の製品インスタンスまたはサービス契約に紐付けられます。

複数のトラフィックマネージャインスタンスからなるクラスタでは、クラスタ全体のライセンス要件を対象にした適切なキー(または一連のキー)が必要です。ライセンスがないトラフィックマネージャは、適切な警告バナーが付いて開発者モードで動作します。

重要: 1つのクラスタでライセンスの異なるトラフィックマネージャを使用することは可能ですが、推奨はされません。クラスタの1つ以上のメンバに対してライセンスがない機能を更新しようとして、クラスタの自動同期に失敗する可能性があるためです。

現在インストールされているライセンスキーを表示するには、**System** (システム) > **Licenses** (ライセンス) をクリックします。このページには、シリアル番号によって識別できる各ライセンスキーと、そのライセンスキーが有効なトラフィックマネージャが表示されます。製品、有効期限、ライセンスされる一連の機能など、特定のライセンスの全ての詳細を表示するには、折りたたみ表示の矢印をクリックします。

1つのトラフィックマネージャインスタンスに複数のライセンスがインストールされている場合、そのトラフィックマネージャでは、以下の順序の基準に基づいて使用するライセンスが選択されます。

- インストールされたライセンスは、開発者モードよりも優先されます。
- 認可されたライセンスは、認可されていないライセンスよりも優先されます。
- リモートに認可されたライセンス(Flexible License Architecture (FLA)タイプのライセンスなど)は、非リモートのライセンス(永続ライセンス)よりも優先されます。
- より機能の豊富なライセンスは、機能の少ないライセンスよりも優先されます。

ライセンスキーの追加と削除

インストールと設定が済んだ直後のトラフィックマネージャで最初に管理 UI を開くと、**Unlicensed** (ライセンスなし) ページが表示され、ライセンスキーをアップロー

ドするか開発者モードで続行するかを選択できます。トラフィックマネージャのハードウェアアプライアンス、仮想アプライアンス、および特定のクラウドインスタンスでは、**Initial Configuration (初期設定)**ウィザードを使用してライセンスをアップロードすることもできます。詳細については、ご使用の製品に最も相応しいバージョンの『**Installation and Getting Started Guide**』を参照してください。

初回以降は、**System (システム) > Licenses (ライセンス)**ページで個々のキーを追加または削除できます。新しいライセンスは、ページ下部の *Install new License Key* (新しいライセンスキーのインストール)セクションを使用してアップロードできます。**Choose File** (ファイルの選択)オプションをクリックして、ライセンスデータが含まれている標準テキストファイルの場所を指定し、**Install Key** (キーのインストール)をクリックします。この操作が正常に完了すると、リストに新しいライセンスが表示されます。

クラスタに存在する不要または無効なキーは、そのライセンスの右にあるチェックボックスを選択して **Remove Selected Keys** (選択したキーの削除)ボタンをクリックすると削除できます。キーを削除するとトラフィックマネージャのライセンスが無効になるような場合には、それが警告され、警告を無視すると削除が確定されます。削除を続行した場合、対象となるトラフィックマネージャは開発者モードに戻ります。

System (システム) > Global Settings (グローバル設定)

トラフィックマネージャソフトウェアでは、幅広いグローバル設定が可能です。グローバル設定にアクセスするには、**System (システム)**ボタンをクリックし、次に **Global Settings (グローバル設定)**タブをクリックします。グローバル設定は、個々のサービスまたはトラフィックには関連しない、トラフィックマネージャ全体の設定が対象となっており、以下のカテゴリに分かれています。

カテゴリ	説明
System Settings (システム設定)	各トラフィックマネージャのネットワークおよびシステムの設定を定義します。
EC2 Account Settings (EC2 アカウント設定)	トラフィックマネージャと Amazon EC2 API の相互関係をコントロールします。
Cache Settings (キャッシュ設定)	各トラフィックマネージャの共有キャッシュの動作をコントロールします。

カテゴリ	説明
FIPS 140-2 Configuration (FIPS 140-2 設定)	「FIPS モード」を有効化するかどうかを決定します。このモードでは、トラフィックマネージャは FIPS 140-2 準拠の暗号モジュールを使用してトラフィックを処理することができます。
SSL Configuration (SSL 設定)	SSL のサポートの動作をコントロールします。
SSL Hardware Support (SSL ハードウェアサポート)	トラフィックマネージャが使用する追加の SSL ハードウェアに適用されます。
Logging (ロギング)	システムイベントおよび仮想サーバのアクセスに対してトラフィックマネージャがロギングを実行する方法をコントロールします。
State Synchronization Settings (状態同期設定)	クラスタメンバがそれぞれの状態データを共有する方法をコントロールします。
Idle Connection Settings (アイドルコネクション設定)	ノードに対するアイドル HTTP コネクションの管理方法をコントロールします。
Java Extension Settings (Java Extension 設定)	Java Extension プロセスの開始および管理方法をコントロールします。
Login and Security Settings (ログインおよびセキュリティ設定)	ユーザのログイン動作をコントロールします(393 ページの「ログインセキュリティとログイン行動」を参照)。
Web Accelerator	Brocade Web Accelerator のグローバルプロパティをコントロールします。
Other Settings (その他の設定)	トラフィックマネージャのその他の設定をコントロールします。

各カテゴリの横にある矢印をクリックすると、そのカテゴリが展開され、オプションが表示されます。変更を完了したら、**Update (更新)**ボタンをクリックします。**Restore Defaults (デフォルトに復元)**ボタンを使用すると、全てのグローバル設定をシステムのデフォルト値に戻すことができます。

重要: これらの設定を変更するときは注意が必要です。デフォルト値は、幅広いシステムで良好な性能が得られ、ソフトウェアの安定性とセキュリティが維持されるよう選択されています。これらの設定を不適切な値に設定すると、システムの安定性とセキュリティが損なわれ、性能に影響を与える可能性があります。

System (システム) > Backups (バックアップ)

トラフィックマネージャのバックアップ管理機能を使用すると、設定をバックアップしたり、バックアップを相互に、また現在の設定と比較したり、以前のバックアップから設定を復元したりできます。

設定のバックアップを管理するには、**System (システム) > Backups (バックアップ)** をクリックします。

また、zconf コマンドラインユーティリティを使用すると、個々の設定オブジェクトについてさらにきめ細かいバックアップ/インポート/エクスポートを実行できます。詳細については、CHAPTER 27 を参照してください。

設定を大幅に変更する前に設定をバックアップすることをお奨めします。そうすることにより、以下のような操作が可能になります。

- 設定を比較し、変更した内容を正確に把握する。
- 変更を破棄して以前のバックアップを復元する。

この機能を使用すると、旧式の仮想マシンまたはクラウドインスタンスを新しいバージョンに置き換える場合などに、トラフィックマネージャマシン間で設定のバックアップをコピーすることもできます。

重要: 設定バックアップファイルは、そのファイルが作成されたトラフィックマネージャに固有であるため、クラスタ内のトラフィックマネージャ間で自動的に複製される設定には含まれません。特定のクラスタメンバを終了または削除した場合、そのメンバに保存した全ての設定バックアップは失われます。

バックアップの作成

設定のバックアップを作成するには、「**Create a Backup (バックアップ作成)**」フォームを使用します。作成したバックアップは、ローカルのトラフィックマネージャマシンに保存されます。

ご使用のトラフィックマネージャのデプロイで Web Application Firewall のライセンス認証および有効化が行われている場合、**App Firewall** ボックスを選択することにより、バックアップにオプションで Application Firewall の設定を含めることができます。

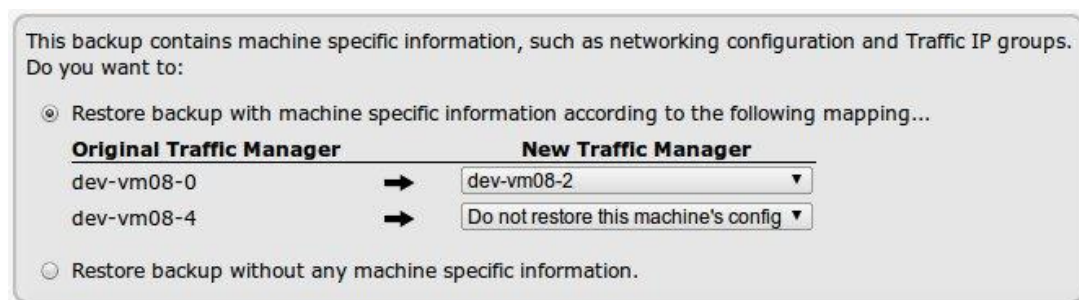
バックアップの復元

Backup Management (バックアップ管理) ページのテーブルでバックアップ名をクリックします。「**Restore Configuration** (設定復元)」オプションを使用して、現在の設定をバックアップの設定内容に置き換えます。

設定のバックアップには、トラフィック IP グループなど、バックアップ元となったトラフィックマネージャのマシン固有の情報が含まれています。そのため、この時点で、現在のトラフィックマネージャのローカルマシン設定を置き換えるかどうかを決定する必要があります。以下の 2 つの方法があります。

- バックアップに含まれるマシン固有の設定を使用する。

復元するバックアップの作成元トラフィックマネージャが同じ場合、ローカル設定をバックアップの設定で単純に上書きします。バックアップ元のトラフィックマネージャが異なる場合は、マッピングコントロールが提示されます。ここで、バックアップに保存されているマシン設定のマッピングを指定することができます。



This backup contains machine specific information, such as networking configuration and Traffic IP groups. Do you want to:

☒ Restore backup with machine specific information according to the following mapping...

Original Traffic Manager		New Traffic Manager
dev-vm08-0	➔	dev-vm08-2 ▼
dev-vm08-4	➔	Do not restore this machine's config ▼

☐ Restore backup without any machine specific information.

Fig 19. 2 台のトラフィックマネージャから成るクラスタの例

この例のバックアップには、2 台のトラフィックマネージャから成るクラスタの設定が保持されています。このツールでは、どのトラフィックマネージャの設定を復元し、どのトラフィックマネージャの設定を無視するかを決定できます。マシン設定間に作成できるのは 1 対 1 のマッピングのみであることに注意してください。

- マシン固有の情報を使用せずにバックアップを復元する。

バックアップに保存されているマシン固有の設定を無視し、現在のトラフィックマネージャのローカルマシン設定を維持します。

ご使用のトラフィックマネージャのデプロイで Web Application Firewall のライセンス認証および有効化が行われている場合、**Include Application Firewall configuration** (Application Firewall の設定を含める) をクリックすることにより、バックアップからオプションで Application Firewall の設定も復元することができます (設定のバックアップが存在する場合)。

バックアップのエクスポート

設定のバックアップは、ローカルのトラフィックマネージャマシンに保存されます。保管しておくために、設定のバックアップをエクスポートすることができます。

Backup Management (バックアップ管理) ページのテーブルでバックアップ名をクリックします。「**Export Configuration** (設定のエクスポート)」オプションを使用して、設定のバックアップをローカルマシンにダウンロードします。

重要: このバックアップは安全に保管してください。このバックアップには SSL 証明書などの機密情報が含まれています。

バックアップのインポート

以前エクスポートした設定のバックアップをインポートすることができます。設定のバックアップをローカルマシンからアップロードするには、**Import a Backup** (バックアップのインポート) セクションを使用します。

インポートしたバックアップは、トラフィックマネージャマシンの設定のバックアップリストに追加されます。バックアップをインポートするだけでは、現在の設定は置き換わりません。必要に応じて、アップロードしたバックアップを復元することにより、現在の設定を置き換えることができます。

System (システム) > Backups (バックアップ) > Partial Backups (部分バックアップ)

このセクションで利用できるバックアップ/復元機能は完全バックアップのものと同じですが、バックアップに設定全体のサブセットのみが含まれるよう調整することができます。このセクションには、**System (システム) > Backups (バックアップ)** ページのリンクからアクセスできます。

この機能には、CHAPTER 27 で説明する `zconf` コマンドラインユーティリティと同様の機能があり、特定のサービスを他のトラフィックマネージャクラスタにコピーする場合に役立ちます。通常のインポートとは異なり、インポートした設定は、既存の設定を置き換えるのではなく、既存の設定にマージされます。

このページは、インポートおよびマージプロセスを処理するためのセクションと、部分バックアップをエクスポートするためのセクションに分かれています。

Importing (インポート)

このセクションでは、部分バックアップまたは完全バックアップをインポートすることができます。表示される **Include Only** (指定の値のみを含む) テキストボックスに適切なフィルタを指定することにより、アップロードされたバックアップファイルから簡単に部分インポートを行うことができます。システムでは、このフィルタに一致するオブジェクトと、選択されたオブジェクトが依存するオブジェクトのみがインポートされます。例えば、選択された仮想サーバが使用しているプールもイ

ンポートされます。使用フォーマットの詳細については、以下の **Filter formats (フィルタフォーマット)** セクションを参照してください。

バックアップファイルをアップロードし、オプションでフィルタを指定すると、システムに対してどのような変更が行われるかを示す *Diff* (差異のリスト) が提示されます。この *Diff* の内容を見直し、間違いがないことを確認してください。 **Apply Partial Backup (部分バックアップの適用)** をクリックして変更内容を適用します。

重要: 部分バックアップをインポートするときに適切に処理を行わないと、設定に矛盾が生じる可能性があります。部分バックアップをインポートするときは、提示される *Diff* の内容を必ず見直してください。全てのバックアップおよび復元操作と同様に、部分バックアップを適用する前に、完全バックアップポイントを作成することを強くお勧めします。

Exporting (エクスポート)

このセクションは、設定の部分バックアップまたは完全バックアップをダウンロードするときに使用します。表示される **Include Only** (指定の値のみを含む) テキストボックスに適切なフィルタを指定することにより、部分バックアップを簡単に行うことができます。システムでは、このフィルタに一致する設定オブジェクトと、そのオブジェクトが依存するオブジェクトのみがエクスポートされます。使用フォーマットの詳細については、以下の **Filter formats (フィルタフォーマット)** セクションを参照してください。

Filter formats (フィルタフォーマット)

Include Only (指定の値のみを含む) フィルタは、以下のいずれかのフォーマットに従って記述された、スペースで区切られたエントリのリストです。

[CONFIGURATION TYPE] 例: vservers

または

[CONFIGURATION TYPE] / [NAME] 例: vservers/Intranet

または

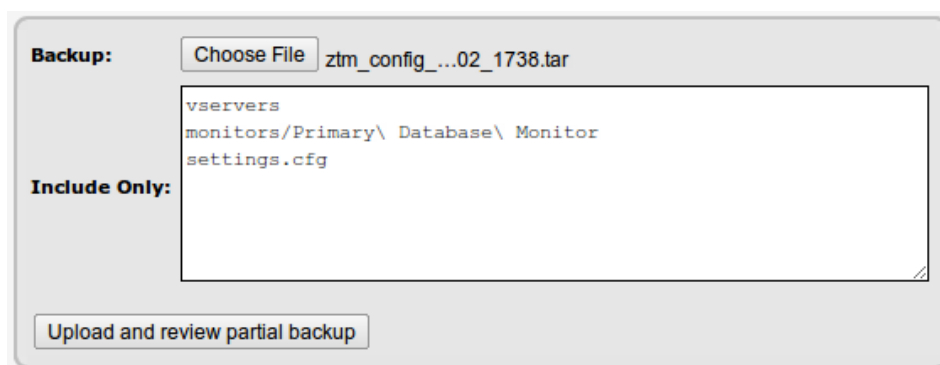
[CONFIGURATION FILE] 例: users

注: 設定名にスペースが含まれる場合は、各スペース文字の前にフォワードスラッシュ(\)を挿入してください。これにより、スペースが区切り文字としてではなく名前の一部として扱われます。例えば、Intranet Master Service の場合は Intranet\ Master\ Service と入力します。

フィルタでは、ワイルドカードとしてアスタリスク(*)を使用して、0 以上の文字を表すことができます。例えば、「net*」は「net」、「net.cfg」、および「network」と一致します。

注: 特殊な設定ファイル settings.cfg を含めることにより、グローバル設定(**System (システム) > Global Settings (グローバル設定)**)をバックアップできます。また、設定ファイル users を含めると、ローカルユーザ設定(**System (システム) > Users (ユーザ)**)をバックアップすることができます。

例えば、全ての仮想サーバ、「Primary Database Monitor」モニタ、およびグローバル設定をインポートする場合は、以下のようにフィルタを入力します。



Backup: Choose File ztn_config_...02_1738.tar

Include Only:

```
vservers
monitors/Primary\ Database\ Monitor
settings.cfg
```

Upload and review partial backup

Fig 20. 部分バックアップをインポートするための Include Only (指定の値のみを含む) フィルタ

以下の表に、フィルタに含めることのできる設定識別子を示します。

識別子	説明
actionprogs	アクションプログラム
actions	アラートアクション
activitymonitor	現在のアクティビティグラフデータ設定
appliance	アプライアンス管理
aptimizer/profiles	Web Accelerator プロファイル
aptimizer/scopes	Web Accelerator スコープ
auth	管理インターフェース/管理サーバ認証コード
authenticators	TrafficScript リモート認証コード
ApplicationFirewallConfig	Application Firewall の設定
auth	認証コード
bandwidth	帯域クラス

識別子	説明
cloudcredentials	クラウドクレデンシャル
custom	カスタム設定のセット
dnsserver/zonefiles	DNS サーバゾーンファイル
dnsserver/zones	DNS サーバゾーン
events	アラートイベントタイプ
extra	その他のファイル
flipper	トラフィック IP グループ
groups	ユーザグループ
jars	TrafficScript の Java Extensions
licensekeys	ライセンスキー
locations	マルチサイト管理およびグローバル負荷分散の地理的位置
monitors	モニタ
persistence	セッション維持クラス
pools	プール
protection	サービスプロテクションクラス
rate	レートシェイピングクラス
rules	TrafficScript ルール
scripts	モニタスクリプト
services	グローバル負荷分散(GLB)サービス
servlets	Java Extension Servlet
slm	サービスレベルモニタリングクラス
ssl/cas	SSL 認証局

識別子	説明
ssl/client_keys	SSL クライアント証明書/キー
ssl/dnssec_keys	DNSSEC キー
ssl/server_keys	SSL サーバ証明書/キー
supplementarykeys	追加ライセンスキー
vservers	仮想サーバ
zxtms	トラフィックマネージャ

部分バックアップへの **Web Application Firewall** の取り込み

Web Application Firewall (WAF)を部分バックアップに含めるには、グローバル設定ファイルも含める必要があります。さらに、バックアップを作成する前に、トラフィックマネージャの設定で WAF を有効化する必要もあります。

すなわち、WAF を有効化した状態で、以下を **Export** (エクスポート)セクションの「Include Only (指定の値のみを含む)」フィルタに追加します。

- settings.cfg
- ApplicationFirewallConfig

同様に、WAF 設定を **Partial Backups** (部分バックアップ)ページからインポートする場合も、**Import** (インポート)セクションの「Include Only (指定の値のみを含む)」フィルタに両ファイルを追加してください。

重要: この方法でアクティブなトラフィックマネージャの設定に settings.cfg ファイルを復元すると、WAF 固有のキー値だけでなく、全てのグローバル設定がオーバーライドされます。部分バックアップでこのファイルのコピーを復元する前に、まずファイルのバックアップを作成することをお奨めします。

アクティビティモニタリング

システム上の現在および過去のアクティビティをトラッキングすることのできるビジュアルトラフィックモニタリングツールがいくつか用意されています。これらのツールにアクセスするには、管理サーバインターフェースのトップバーにある **Activity** (アクティビティ)ボタンをクリックします。

Activity (アクティビティ) > Current Activity (現在のアクティビティ)

Current Activity (現在のアクティビティ) ページには、システム上のトラフィックアクティビティを示すグラフがリアルタイムに表示されます。このページに表示されるデータは幅広くカスタマイズできます。

標準データセットがいくつか用意されており、これらを選択すると、現在の帯域幅などのトラフィック値、トラフィックマネージャが処理している接続数、空きメモリや予備 CPU 時間などのハードウェア値などがプロットされます。この情報の多くは、仮想サーバ、プール、またはノードごとに分割することができます。また、グラフのサイズやタイムスケールを指定したり、軸タイプを線形軸と対数軸から選択したり、パイチャートを選択したりすることが可能です。さらに、グラフの対象を全てのマシンにすることも、一度に 1 台のマシンにすることもできます。

設定を選択したら、**Plot Data** (データをプロット) をクリックするとグラフが表示されます。データを .tsv (タブ区切り変数) ファイルとしてダウンロードして、解析に利用することもできます。

グラフのデータをさらにカスタマイズするには、**Change Data** (データ変更) ボタンをクリックします。これにより **Current Activity** (現在のアクティビティ) > **Edit** (編集) ページが表示され、このページでプロットするデータを指定することができます。

Global Values (グローバル値) セクションでは、モニタリングする統計値を選択し、システム全体のデータをプロットすることができます。また、**Virtual Server Values** (仮想サーバ値)、**Pool Values** (プール値)、および **Node Values** (ノード値) セクションでは、データをこれらの各インスタンスに分割してプロットすることができます。データ転送および接続カウントのほか、DNS および SNMP リクエストデータ、コンテンツ圧縮、および HTTP 書き換えを確認することができます。

Service Protection Class Values (サービスプロテクションクラス値) セクションでは、サービスプロテクションクラスによって拒否されたリクエストに関する情報をプロットできます。サービスプロテクションクラスの全てまたはサブセットのデータを表示できます。また、単一の IP アドレスによる接続カウントが高すぎる、リクエストに許可されていないコンテンツが含まれるなど、拒否理由ごとにデータを分割することもできます。

プロットの設定を選択したら、後で使用するために保存することができます。完了したら、**Apply** (適用) をクリックします。

SNMP による性能のモニタリング

簡易ネットワーク管理プロトコル(SNMP)は、ネットワークのモニタリングおよび管理に関するオープン規格です。SNMP を使用すると、ネットワーク上のデバイスをモニタリングし、性能データを収集することができます。

トラフィックマネージャは、複数の SNMP バージョン(v1、v2c および v3)をサポートしています。

トラフィックマネージャでは、HP OpenView™などの SNMP ツールを使用して性能をモニタリングできます。また、トラフィックフローに関する情報を抽出して、解析に利用することもできます。

注: トラフィックマネージャは、SNMP トラップと呼ばれる非同期 SNMP 通知を送信することもできます。詳細については、CHAPTER 21 を参照してください。

SNMP を設定するには、**System** (システム) ボタンをクリックし、次に **SNMP** タブをクリックします。ここでは、選択した SNMP バージョンに関連するトラフィックマネージャ *MIB* (マネジメントインフォメーションベース) をダウンロードできます。MIB は SNMP が使用する情報のデータベースで、以下の 2 つの SMI (管理情報構造) バージョンがあります。

- **SMIv1:** SNMPv1 クライアントのみ
- **SMIv2:** SNMPv2c および v3 クライアントのみ

いずれかのリンクをクリックしてブラウザに MIB を表示するか、右クリックしてコンテンツをコンピュータ上のファイルに保存します。

一般的な SNMP 設定

SNMP を有効化するには、snmp!enabled ラジオボタンを **Yes** (はい) に設定します。

SNMP サービスで監視するポートおよびバインド IP アドレス、サービスへのアクセスを許可するクライアントなど、一般的な設定を行うことができます。

SNMP の標準ポートは 161 ですが、非特権ポートを使用する場合は、1161 を使用することをお奨めします。

SNMPv1 および SNMPv2c の設定

以下の設定を行うことができます。

設定	説明
snmp!community	SNMP サーバが正常に動作するためには、適切なコミュニティストリングが必要です。コミュニティストリングはパスワードに相当し、これによって、認可された SNMP リクエストかどうかを識別します。ただし、コミュニティストリングは SNMP リクエストでプレーンテキストとして送信され、セキュアではないため、機密情報をコミュニティストリングとして使用しないでください。多くの SNMP クライアントでは、デフォルト値として「public」が使用されます。トラフィックマネージャは SNMP サービスへのアクセスを制限できるため、この秘密情報の強度は重要ではありません。 コミュニティストリングをブランク/空のままにすると、全ての SNMPv1 および SNMPv2c コマンドが拒否されます。

SNMPv3 の設定

このセクションでは、SNMPv3 コマンドを受け入れて、それに対するレスポンスを返すための認証およびプライバシー設定を指定できます。

注: パスワードから暗号化キーを生成するために使用される要素の 1 つにエンジン ID があります。トラフィックマネージャは、Zeus⁴エンタープライズ OID (7146)と、トラフィックマネージャが稼働しているマシンの最初の MAC アドレスを組み合わせることでこの値を生成します。このセクションには、接続しているトラフィックマネージャの正しいエンジン ID 値が表示されます。

全ての SNMPv3 通信は、以下に示すユーザ設定と照合され、セキュリティパラメータが正確に一致していないコマンドは拒否されます(ただし、認証されることがなく、空のユーザ名で送信される可能性のある snmpEngineID 検出メッセージは許可され、それに対するレスポンスが返されます)。

⁴ Zeus は、トラフィックマネージャの基盤となった製品の名前です。

以下の設定を行うことができます。

設定	説明
snmp!username	全ての SNMPv3 コマンドおよび通信に必要となる SNMPv3 ユーザのユーザ名です。ブランク/空に設定すると、全ての SNMPv3 コマンドが拒否されます。 トラフィックマネージャでは、単一の SNMPv3 ユーザアカウントが定義されていることが要求され、このアカウントが許可されます。このユーザの認証およびプライバシー設定は、以下に示すフィールドによって定義され、RFC3414 (http://www.ietf.org/rfc/rfc3414.txt)で規定される「User-based Security Model」に基づきます。
snmp!security_level	SNMPv3 ユーザによる、認証を使用するかどうか、またプライバシーを使用するかどうかの選択です。SNMPv3 コマンドはこの選択に従う必要があります。従わないコマンドは拒否されます。SNMPv3 レスポンスはこの選択に従います。 ここで Authentication (認証)を選択すると、コマンドおよびレスポンスは、snmp!auth_password から導き出されたキーを使用して暗号化された署名がなされ、この署名がメッセージに組み込まれます。不正に署名されたコマンドまたはレスポンスは検出され、拒否されます。これは、パスワードの機密性が維持されている限り、信頼されたパーティへのなりすましを防ぐことを目的としています。 ここでさらに Privacy (プライバシー)を選択すると、コマンドおよびレスポンスの一部が、snmp!priv_password から導き出されたキーを使用して DES で暗号化されます。これは、パスワードの機密性が維持されている限り、盗聴者によるメッセージコンテンツの読み取りを防ぐことを目的としています。

設定	説明
snmp!hash_alg	SNMPv3 ユーザによる、暗号ハッシュアルゴリズムの選択です。暗号ハッシュアルゴリズムは SNMPv3 コマンドで要求され、レスポンスに使用されます。 サポートされているハッシュアルゴリズムは MD5 および SHA-1 です。詳細については、RFC 3414 (http://www.ietf.org/rfc/rfc3414.txt)を参照してください。 snmp!security_level を「No Authentication, No Privacy (認証なし、プライバシーなし)」に設定した場合、この設定は使用されません。
snmp!auth_password	SNMPv3 ユーザの認証パスワードです。 これは共有秘密情報です。SNMP クライアントユーザアカウントに同じ認証パスワードを設定する必要があります。 SNMPv3 通信の認証キーは、このパスワードから導き出されます。 snmp!security_level を「No Authentication, No Privacy (認証なし、プライバシーなし)」に設定した場合、この設定は使用されません。
snmp!priv_password	SNMPv3 ユーザのプライバシー(メッセージ暗号化)パスワードです。 これは共有秘密情報です。SNMP クライアントユーザアカウントに同じプライバシーパスワードを設定する必要があります。 SNMPv3 通信のプライバシーキーは、このパスワードから導き出されます。セキュリティ上の理由から、認証パスワードと同じパスワードを使用しないことをお奨めします。 snmp!security_level を「No Authentication, No Privacy (認証なし、プライバシーなし)」または「Authentication only (認証のみ)」に設定した場合、このパスワードは使用されません。

性能およびイベントモニタリングのカスタマイズ

TrafficScript 関数の `counter.increment()` を使用すると、10 個の内蔵カウンタのいずれかを増加させることができます。

SNMP を使用してこれらのカウンタの値を読み込み、アクティビティモニタにグラフとして表示することができます。

これらのカウンタは、TrafficScript で検出したカスタムイベントを記録するために使用できます。以下に例を挙げます。

- TrafficScript ルールで認証を実行すると、認証の失敗件数をカウントできます。
- 特定のリクエストクラスのリソース使用量が特に多い場合に、リクエストが行われた回数のチャートを作成することができます。
- 例えば既知の Web ワームシグネチャについてセキュリティチェックを実行すると、そのシグネチャが検出された回数をカウントし、チャートとして表示することができます。

内蔵 SNMP カウンタの値を増加させるには、**System (システム) > Global Settings (グローバル設定) > Other Settings (その他の設定)**にある **snmp_user_counters** 値を変更します。

Activity (アクティビティ) > Historical Activity (ヒストリカルアクティビティ)

デフォルトでは、トラフィックマネージャは、過去 90 日間のシステムのアクティビティを記録します。データが保存される日数を変更するには、**System (システム) > Global Settings (グローバル設定) > Other Settings (その他の設定)**をクリックし、**statd!days** に格納された数値を変更します。0 の値は、データを永続的に保管することをトラフィックマネージャに指示します。

ヒストリカルシステムアクティビティを表示するには、**Activity (アクティビティ) > Historical Activity (ヒストリカルアクティビティ)**をクリックします。

仮想サーバ、プール、またはノードごとに表示される 1 分あたりのヒット数または 1 秒あたりの処理ビット数のグラフを選択します。トラフィックマネージャには、1 時間から 1 年まで、プロットするデータに対するタイムスケールの範囲が用意されています。ただし、利用できるオプションは、**statd!days** で設定した値によって影響を受けます。

システム内の全ての仮想サーバ、プール、ノード、または選択したサブセットに対してデータをプロットすることを選択します。また、チャートサイズと軸タイプも選択します。

Plot Data (データのプロット)をクリックし、現在の設定を使用してグラフを表示します。データを **.tsv (タブ区切り変数)ファイル** としてダウンロードして、解析に利用することもできます。 **Activity (アクティビティ) > Map (マップ)**

このマップには、トラフィック管理システムが処理しているリクエストの送信元の地理的位置が表示されます。

マップはインタラクティブに設計されており、マップ上を移動したり、インフラストラクチャに必要な詳細レベルにズームインしたりするためのツールが用意されています。マウスの左ボタンを押しながらマップを所要の目的地にドラッグすることにより、任意の地理的位置を中央に表示することができます。

マップ上に表示される小さいドットは、特定のサービスにリクエストを送信したユーザを表します。

Activity (アクティビティ) > Connections (コネクション)

注: このセクションで説明する機能は、それぞれライセンスによって管理されています。一部のライセンスでは**基本機能モード**のみが提供されます。このモードでは、単純なコネクション情報のみが表示され、カスタマイズオプションが少なく、利用できるフィルタは1つのみです。**フルモード**は、ここで説明する全ての機能をサポートし、*advanced real-time analytics* ライセンスオプションで提供されます。このオプションのご購入については、サポート窓口にお問い合わせください。

基本モード

Connections (コネクション) ページでは、トラフィックマネージャが処理している現在および最近のコネクションを、日付順で(最も新しいものから順に)モニタリングできます。これは、特定時点におけるコネクションのスナップショットです。**Refresh Snapshot** (スナップショットを更新) ボタンを押すまで、それ以降のコネクションは表示されません。

コネクションテーブルの左上にある **[+]** ボタンを使用すると、フィールド列の表示/非表示を切り替えることができます。このボタンをクリックすると、現在テーブルに表示されているフィールドのドロップダウンリストが表示されます。このリストで各フィールドをクリックすると、そのフィールドの表示/非表示が切り替わります。

表示されているデータに単純なフィルタを適用して、特定ノードへのコネクションのみを表示することができます(「To」列)。これは、**フルモード**で利用可能なフィルタ機能の単一フィールドバージョンです。次のセクションで説明する使用手順を参照してください。

フルモード

このモードでは、基本モードで提供される全ての機能に加え、コネクションの詳細およびカスタマイズを利用できます。ソート基準にする列のヘッダをクリックする

と、表示されているデータをソートすることができます。ソートされた列の横にある小さい矢印は、ソート順を示します。この矢印をクリックして、昇順と降順を切り替えることができます。

ノードフィルタに加え、テーブル内の任意のフィールドまたはデータでコンネクションをフィルタすることができます。これを行うには、ページの上部にあるフィルタセクションのドロップダウンボックスを使用するか、テーブル内のいずれかのデータ値を直接クリックします。後者の方法では、インラインの *add filter* ダイアログが表示され、選択したフィールドの値を共有していない全てのコンネクションを除外することができます。

例えば、特定の IP アドレスとポートから送信され、結果的に HTTP レスポンスコード 200 が生成されたリクエストを表示するとします。この場合、まず **From** 列で、使用する IP アドレスをクリックします。選択した IP が入力された状態でフィルタポップアップが表示されます。

+	Traffic Manager	From	To	VS	Pool	Resp. Code	Request
P	coeus.cam.zeus.com	10.100.1.14:37077	92.52.65.222:80	website	website	200	/favicon.ico
P	coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/bullet.gif
From equals 10.100.1.14:37074 Add Filter X							
P	coeus.cam.zeus.com	10.100.1.14:37076	92.52.65.222:80	website	website	200	/assets/default/Site/en/images/tri/triboo.gif
P	coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/images/gli/gliit.gif
P	coeus.cam.zeus.com	10.100.1.14:37072	92.52.65.222:80	website	website	200	/assets/default/Site/en/images/dom/dominos.gif
P	coeus.cam.zeus.com	10.100.1.18:3636	212.58.246.93:80	intranet	test	301	/assets/default/Site/en/images/vir/virgin.gif
P	coeus.cam.zeus.com	10.100.1.18:3620	212.58.246.93:80	intranet	test	301	/
P	coeus.cam.zeus.com	10.100.1.18:3499	212.58.246.92:80	test	test	301	/
P	coeus.cam.zeus.com	10.100.1.18:3438	212.58.246.92:80	test	test	301	/

Fig 21. 「From」 列へのフィルタの追加

Filter (フィルタ) ボタンをクリックして、表示されたロジックを適用します。このフィルタが、このページの上部に表示される *Active Filters* に追加されます。次に、**Resp. Code** (レスポンスコード) 列で値「200」をクリックして、この値以外を除外するフィルタを適用します。これにより、目的の接続のみが表示されます。

Connection Filters

[Remove](#)

Active Filters:

From	equals	10.100.1.14:37074	☐
Response Code	equals	200	☐

Add Filter: Select field to filter...

Update filters
Clear filters
Showing 5 connections

Refresh
Download

+ Traffic Manager	From	To	VS	Pool	Resp. Code	Request
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/bullet.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/images/vir/virgin.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/divider.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.222:80	website	website	200	/assets/img/logo.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/css/main.css

Fig 22. アクティブなコネクションのフィルタ

アクティブなフィルタを調整するには、変更を行った後に **Update Filters** (フィルタの更新) ボタンをクリックします。これにより、コネクションデータが再表示されます。フィルタを削除するには、削除する条件の横のチェックボックスを選択し、

Update Filters (フィルタの更新)をクリックします。**Clear Filters** (フィルタの消去)をクリックすると、全てのフィルタを削除できます。

Refresh (更新)ボタンを使用すると、結果のスナップショットを更新できます。また、**Download** (ダウンロード)ボタンを使用すると、結果の**.tsv** (タブ区切り値)ファイルを作成し、他のアプリケーションでの解析に利用できます。

さらに、各行の左側にある虫眼鏡アイコンをクリックすると、特定の接続について以下の追加の詳細が表示されます。

セクション	説明
Request tracing (リクエストトレーシング)	このセクションには、内部接続処理イベントの時系列が表示されます。リクエストの全所要時間、そのうち TrafficScript ルールの処理時間、およびその他の役立つ情報を表示できます。 この情報は、その接続を処理した仮想サーバで request_tracing!enabled キーが有効化されている場合のみ記録されます。詳細については、73 ページの「接続分析」を参照してください。
Request details (リクエストの詳細)	このセクションには、リクエストに含まれていた全てのリクエストヘッダが表示されます。
Response details (レスポンスの詳細)	このセクションには、全てのレスポンスヘッダ (TrafficScript または詳細キャッシュロギングによって追加されたものも含む)が表示されます。

注: 接続スナップショットのサイズは、**System** (システム) > **Global Settings** (グローバル設定) ページで拡張することができます。**recent_conns** キーの **Logging** (ロギング) セクションを確認してください。

Activity (アクティビティ) > Draining Nodes (ノードのドレイン)

例えばメンテナンスやアップグレードのために、ノードをシステムから削除したい場合、そのノードをドレインするよう設定できます。ドレインされたノードは引き続き既存のセッションを処理しますが、トラフィックマネージャはそのノードに接続を送信しなくなります。**Draining Nodes** (ノードのドレイン) セクションには、現在ドレインしているノードに関する情報が表示されます。

このページには、ドレイン中の各ノードについて、最後のコネクションからの経過時間および現在のアクティブな接続数が表示されます。**Reload This Page** (このページのリロード)をクリックすると、この情報を更新できます。

アクティブなセッションが全て終了した時点で、ノードを安全に削除することができます。これを行うには、**Pools (プール) > Edit (編集)** ページを使用するか、**Remove a Node** (ノードの削除)ウィザードを使用します。

詳細については、CHAPTER 5 の「ノードのドレインと無効化」セクションで、ノードのドレインに関する説明を参照してください。

Activity (アクティビティ) > View Logs (ログ表示)

全てのトランザクションをリクエストログに記録するよう仮想サーバを設定することができます。**Virtual Server (仮想サーバ) > Request Logging (リクエストロギング)** ページ (CHAPTER 4 の「Request Logging (リクエストロギング)」セクションを参照)で、記録する情報およびログファイルの書き込み先を設定できます。

Activity (アクティビティ) > View Logs (ログ表示) ページでは、リクエストログにレコードが追加されていく状況をリアルタイムに確認できます。

注: トラフィックマネージャ仮想インスタンスおよびクラウドインスタンスは自己完結型であり、自己管理される専用のログパーティションを備えています。ログファイルはこのパーティションに保存され、自動的にアーカイブおよび削除されます。そのため、ログファイルの保存場所を設定する必要はありません。

注: ログファイルは、**Activity (アクティビティ) > View Logs (ログ表示)** ページで表示でき、**Activity (アクティビティ) > Download Logs (ダウンロードログ)** ページでダウンロードできます。また、トラフィックマネージャコントロール API を使用して、トラフィックマネージャ仮想/クラウドインスタンス上のログファイルをダウンロードしたり管理したりすることもできます。

クラウドクレデンシャル

トラフィックマネージャソフトウェアをクラウド環境で使用する場合、トラフィックマネージャがクラウドプロバイダに対して API 呼び出しを行うために、クレデンシャルが必要になる場合があります。まず、オートスケーリングを使用するよう設定されたプールには、クラウド内でスケーリングメカニズムを使用するために API クレデンシャルが必要です。このセクションでは、選択したクラウドプロバイダ API の必要条件に基づいて、そのようなクレデンシャルをカタログオブジェクトに記録することができます。プールのオートスケーリングの詳細については、99 ページの「オートスケーリング」を参照してください。

トラフィックマネージャには、クレデンシャルセットのベースになる多数の内蔵クラウド API が用意されています。

- VMware vSphere
- Amazon EC2
- Rackspace クラウド

また、**Catalogs** (カタログ) > **Extra Files** (追加ファイル) で適切な実行可能スクリプトをアップロードすることにより、カスタムクラウド API を追加することもできます。詳細については、Brocade が提供するコミュニティ Web サイト (<http://community.brocade.com>) のトラフィックマネージャセクションを参照してください。

新しいクラウドクレデンシャルセットを作成するには、以下の手順に従います。

1. **Catalogs** (カタログ) > **Cloud Credentials** (クラウドクレデンシャル) をクリックします。
2. **Name** (名前) フィールドに名前を入力します。
3. **Cloud API** (クラウド API) で、使用するクラウド API を選択します。
4. 残りのフィールドは、選択した API によって異なります。通常は、ID、パスワード/パスコード、および追加の認証情報を入力するためのフィールドが表示されます。
5. **Create Cloud Credentials** (クラウドクレデンシャルの作成) ボタンをクリックします。

以下の表に、新しいクレデンシャルセットを作成するときに表示される各フィールドの詳細を示します。

設定	説明
Name	トラフィックマネージャの設定内でこのクレデンシャルセットを識別するための名前です。
Cloud API	選択したクラウドプロバイダ API です。
ID / Name (cred1)	API 呼び出しに使用するクラウドプロバイダアカウントのユーザ名または ID です。
Auth Key / Password (cred2)	ユーザ名/ID に関連付けられたパスワードです。

設定	説明
Token / Server (cred3)	一部のクラウドプロバイダでは、API 呼び出しを行うためにさらに認証トークンまたは追加項目が必要になります。VMware vSphere では、API 呼び出しを受け入れるために vCenter ホスト名/IP を必要とします。このような追加情報はここで指定できます。
update_interval	トラフィックマネージャはユーザに代わって、定期的にクラウドプロバイダに対して実行中の全インスタンスのステータスをクエリします。この動作は、外部システムによってノードが追加または削除されたときに最新の状態に保つために必要になります。 update_interval は、これらのステータス呼び出しを行う頻度(秒単位)を決定します。一部のクラウドでは、このような呼び出しを行える 1 分あたりの回数が制限されることに注意してください。
change_process_timeout	変更プロセスの最大所要時間(秒単位)です。例えば、クラウド API に対してノードの作成/削除リクエストが行われるときに、この設定により、そのリクエストが完了するまで待機する時間が指定されます。

Amazon EC2 クレデンシャルの IAM ロール

Amazon EC2 のデプロイでは、ローカルに保存されているクレデンシャルの代わりにアイデンティティおよびアクセス管理(IAM、Identity and Access Management)ロールを使用できます。EC2 クレデンシャルセットを作成している場合、アクセスキーID およびシークレットキーを指定するか、これらのフィールドをブランクにして、その代わりに IAM ロールを使用するかを選択します。

注: トラフィックマネージャは、ロールの有無に関係なく、割り当てられた IAM ロールよりもここで指定されたクレデンシャルを優先して使用します。

ロールの作成および管理方法など、IAM ロールに関する詳細については、AWS に関するドキュメント(<http://aws.amazon.com/documentation/>)を参照してください。

Web Application Firewall

Brocade Virtual Web Application Firewall は、最新世代のミッションクリティカルな Web アプリケーションに対する攻撃を検出し、これらのアプリケーションを保護するエンタープライズレベルの Web Application Firewall です。

Brocade Virtual Web Application Firewall を使用すると、中央からセキュリティのモニタリング、レポートの送信、およびアラートの報告を行うことができます。また、外部攻撃に対する Web アプリケーションおよびインフラストラクチャの保護をカスタマイズして適用することも可能です。

注: これは、適切なライセンスキーのアップグレードによってのみアクティブになる、トラフィックマネージャのオプション機能です。詳細については、サポート窓口にお問い合わせください。

概要

Application Firewall は、オプションで提供されるトラフィックマネージャのコンポーネントです。ライセンスを入手し、クラスタで有効化すると、トラフィックマネージャソフトウェアの総合的な機能に完全に統合されます。

トラフィックマネージャは、Web サービスのインフラストラクチャと総合的なコントロールを提供し、指示があった場合にトラフィックを Application Firewall に渡します。

本章では、Application Firewall の概要とトラフィックマネージャとの相互関係について説明します。Application Firewall の詳しい設定および使用手順については、『**Brocade Virtual Web Application Firewall User Guide**』およびオンラインヘルプを参照してください。

Application Firewall の有効化

重要: トラフィックマネージャのソフトウェアインスタンスと非ソフトウェアインスタンスが混在するクラスタで Application Firewall コンポーネントを有効化しないでください。この構成はサポートされていません。

Application Firewall を使用するためには、サポート窓口から適切なライセンスキーを取得します。新しいライセンスをインストールすると、トラフィックマネージャの管理 UI の **System (システム) > Application Firewall** ページがアクティブになります。このページを使用して、Application Firewall コンポーネントの有効化、無効化、および設定を行います。

他の設定変更と同様に、クラスタ内の 1 台のトラフィックマネージャで Application Firewall の有効化または無効化を行うと、その他全てのクラスタメンバの設定にも同等の変更が自動的に反映されます。

重要: Application Firewall を有効化すると、トラフィックマネージャソフトウェアが自動的にリスタートします。サービスへの影響を最小限に抑えることのできるタイミングでこの手順を行うことをお奨めします。

トラフィックマネージャの管理 UI で利用できる **Application Firewall** 機能

Application Firewall のコントロールおよび設定は、トラフィックマネージャのナビゲーションバーにあるボタンを使用して行います。このボタンを使用すると、トラフィックマネージャと Application Firewall のユーザインターフェースが切り替わります。

注: このボタンを表示させるには、Application Firewall コンポーネントを有効化する必要があります。

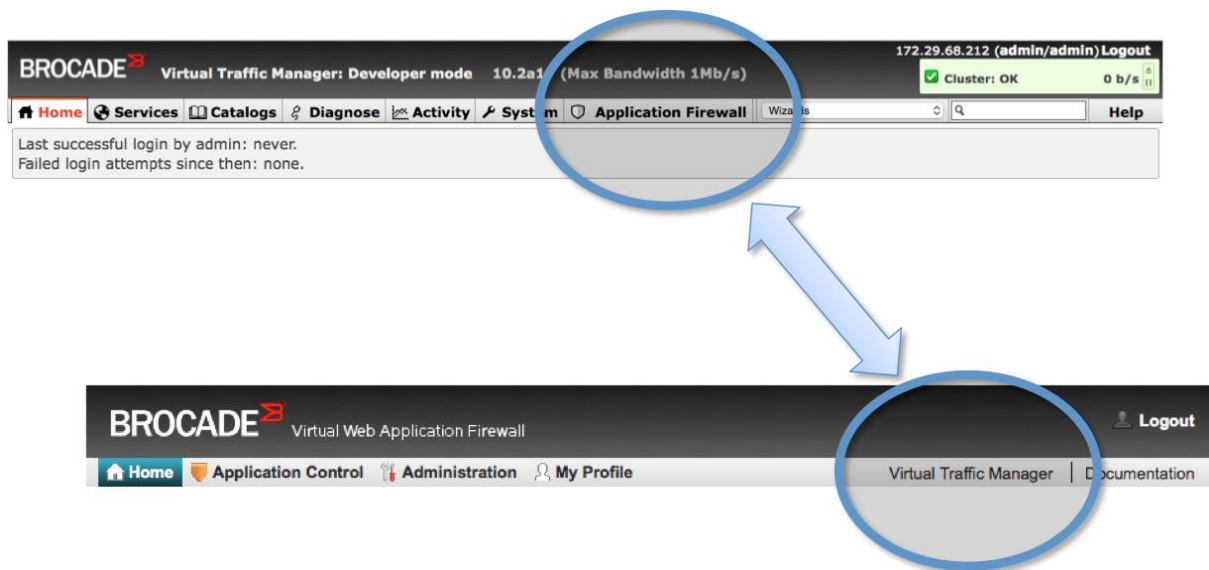


Fig 23. トラフィックマネージャと Application Firewall のメインナビゲーションバー間のリンク

メインのトラフィックマネージャ管理 UI には、Application Firewall の通信や動作を管理するための機能も多数用意されています。

- **System (システム) > Application Firewall** ページには、Application Firewall コンポーネントを有効化および無効化するためのセクションがあります。また、Application Firewall が使用するネットワークポートや実行するプロセス数など、低レベルの Application Firewall 設定も含まれています。
- **System (システム) > Traffic Managers (トラフィックマネージャ)** ページの Software Restart (ソフトウェアリスタート) セクションに「Restart Application Firewall... (Application Firewall のリスタート)」ボタンが追加されます。一部の設定変更には、トラフィックマネージャ自体のリスタートは不要でも、Application Firewall のリスタートが必要になる場合があります。
- **Services (サービス) > Virtual Servers (仮想サーバ) > Edit (編集)** ページの Basic Settings (基本設定) セクションには、各 HTTP サービスについて Application Firewall を有効化または無効化するためのコントロールがあります。

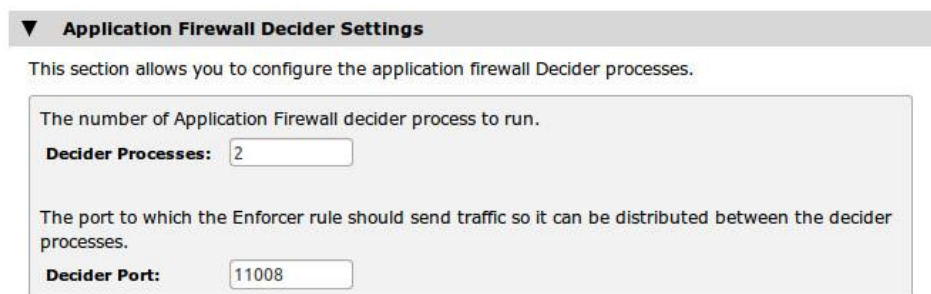
- ルールカタログには、新しい組み込み式 **TrafficScript** ルールである「**Application Firewall Enforcer**」があります。トラフィックマネージャはこのルールを使用して、Application Firewall サービスとの通信を維持します。
- トラフィックマネージャにより、Application Firewall サービス、特に現在の実行状態に関連するステータスメッセージがトラフィックマネージャのイベントログおよび監査ログに記録されます。特定の設定キーの変更およびモニタエラーに加え、ソフトウェアの停止、起動、リスタートが全て記録されます。
Application Firewall でも、サービスのセキュリティに関連する攻撃およびその他のイベント情報を記録する独自のローカルイベントログが保持されます。このログは、Application Firewall の管理 UI から表示できます。詳細については、『**Brocade Virtual Web Application Firewall User Guide**』を参照してください。
- 「Application Firewall」設定セクションが **Diagnose** (診断) ページに表示されます。このセクションには、クラスタ内で発生したファイアウォールの問題に関する詳細が表示されます。トラフィックマネージャと Application Firewall 間の通信が何らかの理由で中断した場合、エラーが発生したことが **Status Applet** (ステータスアプレット) に表示されます。このエラーインジケータをクリックすると、**Diagnose** (診断) ページに移動します。このページで、具体的にどのような問題が発生したのかを確認し、この状況に対処するための適切なアクションを決定することができます。

System (システム) > Application Firewall ページ

このページには、Application Firewall の性能および通信に関する設定オプションがあります。

注: 通常の運用では、このセクションの設定を変更する必要はありません。ここで説明する内容は、参考のための情報です。

Application Firewall には、Decider プロセスと Enforcer TrafficScript ルール間の通信専用のベースポートが必要です。



▼ **Application Firewall Decider Settings**

This section allows you to configure the application firewall Decider processes.

The number of Application Firewall decider process to run.

Decider Processes:

The port to which the Enforcer rule should send traffic so it can be distributed between the decider processes.

Decider Port:

Fig 24. Decider の設定

工場出荷時のデフォルトポートはほとんどのデプロイに適していますが、システムでこのポートを利用できない場合は、別の値を指定できます。このページで設定を変更すると、Application Firewall が自動的にリスタートされます。

Decider Processes (Decider プロセス)設定を使用すると、システムで利用可能な CPU コア数に応じて Application Firewall の性能を調整することができます。より多くのトラフィックを処理してシステムの性能を向上させるには、この値を最大 8 に増やしてさらなる Decider プロセスをインスタンス化します。ただし、空きリソース以上のプロセスを実行すると、性能が抑制され、サービスの低下につながります。このような状況では、プロセス数を減らすことによって、システムの全体的な性能を高めることができます。

Decider プロセスの数は、トラフィックマネージャクラスタ内で搭載 CPU コアの数 が最も少ないサーバの CPU 数と同じ数に設定することをお奨めします。例えば、クラスタ内に 2 台のトラフィックマネージャがあり、一方のトラフィックマネージャには 4 個の CPU コアが、もう一方には 2 個の CPU コアが搭載されている場合、Decider プロセスの数を 2 に設定してください。

▼ Local Application Firewall Ports

This section allows you to configure additional ports used by the application firewall on this machine, this might be necessary to avoid port conflicts with existing services. To change these settings on another cluster member, you must log in to that cluster member's Administration Server.

The base port from which the Application Firewall decider processes should run. Ports will be used sequentially above this for each additional decider process that runs.

Internal Decider Base Port:

The Application Firewall Administration Server port, this port is only open on localhost.

Admin Server Port:

The Application Firewall XML Master port, this port is used on all IP addresses.

Admin Master Port:

The Application Firewall XML Slave port, this port is used on all IP addresses.

Admin Slave Port:

The Application Firewall Updater GUI Server Port, this port is used on localhost only.

Updater Admin Server port:

The Application Firewall Updater External Control Center Port, this port is used on localhost only.

External Updater Control Center Port:

The Application Firewall Updater GUI Backend Port, this port is used on localhost only.

Internal Updater Control Center Port:

The Application Firewall REST Internal API port, this port should not be accessed directly

Internal REST API Server Port:

The Application Firewall internal communication base port. The Application Firewall will use ports sequentially above this for internal communication. These ports are bound only to localhost.

Internal Communication Base Port:

Fig 25. 追加の通信ポート

これ以外に、Application Firewall には内部通信用にいくつかのポートが必要です。通常の場合は、デフォルト値を変更する必要はありませんが、ポート番号の競合を解決する場合は、このページで必要なポート設定を別の値に変更してください。

Enforcer と Decider

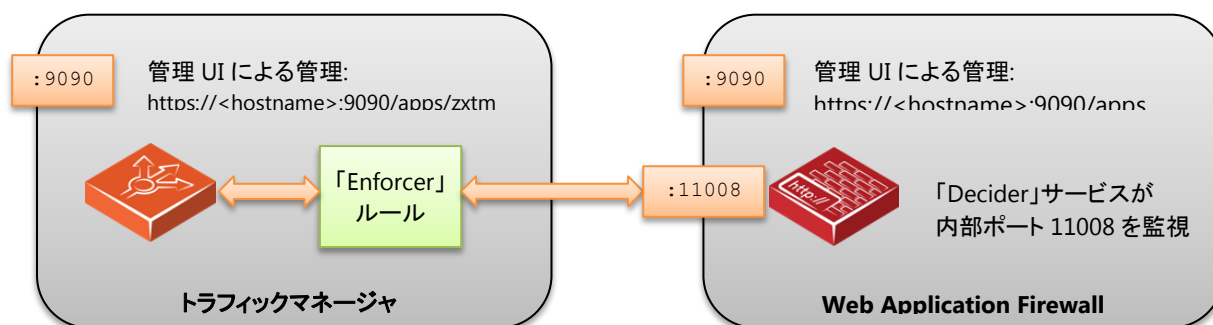


Fig 26. トラフィックマネージャの Enforcer TrafficScript ルールにより、Decider サービスによるチェックのためリクエストがデフォルトローカルポート 11008 に転送されます。Decider から「許可」または「拒否」の決定が返されると、Enforcer ルールによってこの決定が適用されます。

Enforcer は、トラフィックマネージャと Application Firewall 間の通信を可能にする、組み込み式の TrafficScript ルール(名前は「Application Firewall Enforcer」)です。全ての HTTP リクエストとレスポンスをキャプチャし、詳細なチェックのため Application Firewall で実行中の Decider サービスに転送します。

Decider は、設定データベースに保存されている一連のルールを使用して HTTP リクエストを評価し、実行するアクションを決定します。その後、Enforcer がその決定を実行し、各リクエストまたはレスポンスは、その決定に従って受け入れられるか、変更されるか、拒否されます。

Enforcer ルールについての詳細は、159 ページの「Enforcer ルール」を参照してください。

Enforcer ルール

トラフィックマネージャは、TrafficScript ルールシステムを介して Application Firewall と通信します。TrafficScript ルールシステムでは、Web アプリケーションとの間で送受信されるトラフィックによって生成されるリクエストとレスポンスに特定のロジックが適用されます。Application Firewall を有効化すると、

「Application Firewall Enforcer」(Enforcer)という名前の特殊なルールが作成され、トラフィックマネージャが HTTP トラフィックを Decider サービスに転送するように指示されます。このルールを仮想サーバに追加するには、**Virtual Server** (仮想サーバ) > **Edit** (編集) ページで **Application Firewall** 設定を有効化します。

Enforcer ルールは、他のルールと同様に有効化、無効化、削除、または編集することはできませんが、移動して TrafficScript ルールの実行順序を変更することは可能

です。これは、リクエストまたはレスポンスを先に処理する必要がある場合に便利です。例えば以下の **TrafficScript** のように、**Enforcer** の実行より前に特定のリクエストをホワイトリストに指定するルールを追加する場合などが挙げられます。

```
connection.data.set( "enforcer.whitelist", 1 );
```

他のルールと同様に、**Enforcer** はルールカタログに保存されます。前述したように、このルールは削除できませんが、**Save As** 機能を使用してコピーを作成することができます。

注: このようなコピーは、元の **Enforcer** ルールと同じように見え、提供されるオプションも同じですが、従来のルールと同様に使用する必要があります。すなわち、通常の方法で仮想サーバ、プール、およびルールに割り当てる必要があります。仮想サーバの *Application Firewall* 設定を使用して割り当てるのは、元の **Enforcer** ルールのみです。

Enforcer には、設定可能なデータ項目数を設定することもできます。ルールの設定ページにアクセスするには、そのルール名をクリックするか、*Edit* (編集) リンクをクリックします。そのままでは実際の **TrafficScript** テキストは表示されません。**Rule text** (ルールテキスト) リンクをクリックして表示します。**TrafficScript** テキストはバイナリフォーマットであるため、人間が読み取れるのは一部のみです。

ルールテキストの一番上に、設定可能なパラメータが多数表示されます。変更が行われると、トラフィックマネージャは自動的にルールテキストを更新します。

- **allowonerror:** Decider にアクセスできない場合に全てのトラフィックを許可するには、1 に設定します(デフォルト: 0)
- **backend_timeout:** Decider を待機する秒数です。この時間が経過するとタイムアウトになります(デフォルト: 5)
- **bypass_enforcer_max: enforcer_max_body_size** よりも大きいリクエストが *Application Firewall* をバイパスすることをブロックするには、0 に設定します。(デフォルト: 1)
- **bypass_file_types:** *Application Firewall* で免除するファイル拡張子のスペース区切りリストです。(デフォルト: 「css js png jpg gif」)
- **debug:** デバッグモードを有効化するには、1 に設定します。このオプションを使用すると、問題の追跡に役立つ詳細なエラーが出力されます(デフォルト: 0)
- **discard_on_reject:** エラーメッセージを送信せずに、拒否されたコネクションを閉じるには、1 に設定します。
- **enforcer_max_body_size:** *Application Firewall* によって処理されるバイト単位の最大ボディサイズです。この上限よりも大きいリクエストは、

bypass_enforcer_max の設定に応じてブロックまたはバイパスされます。(デフォルト: 2097152)

注: 通常は、これらの設定を変更する必要はありません。変更すると、ルールが正しく機能しなくなる可能性があるため、どのような結果になるかが完全に分かっている場合のみ変更してください。

ユーザ管理

トラフィックマネージャと Application Firewall 間には、ユーザおよびグループアクセスが維持されています。Application Firewall の新規ユーザには、トラフィックマネージャの管理 UI の **admin** (管理) グループのメンバと同じ認証レベルが設定されます⁵。そのため、通常の管理タスクを実行するときに、追加のアクセス権限またはログイン処理は必要ありません。いずれかの UI で **Logout** (ログアウト) をクリックすると、ユーザは両方のアプリケーションからログアウトすることができます。

重要: トラフィックマネージャからユーザを削除しても、そのユーザは Application Firewall から自動的に削除されません。この操作は、管理者が手動で行う必要があります。

Demo (デモ) や **Monitoring** (モニタリング) など、トラフィックマネージャの他のユーザグループのメンバは、**System** (システム) > **Users** (ユーザ) > **Groups** (グループ) > **Edit Group** (グループの編集) ページでパーミッションが正しく設定されている場合のみ、Application Firewall の様々な UI 機能にアクセスできます。トラフィックマネージャの **System** (システム) セクションには、そのための以下の設定があります。

- **Application Firewall:** トラフィックマネージャの管理 UI の **System** (システム) > **Application Firewall** ページに対するアクセスをコントロールします。
- **Application Firewall > Administration (管理):** (前述のツールバーボタンからの) Application Firewall の管理 UI に対するアクセスをコントロールします。

注: Application Firewall の管理 UI でユーザアカウントをあらかじめ定義しておくことにより、管理ユーザを自動的に作成するメカニズムをオーバーライドすることができます(制限付きアクセスを指定したり、非管理グループを使用したりする場合など)。これにより、Application Firewall の UI では、トラフィックマネージャ内で同じ名前を持つユーザに対して指定したアクセスのみが与えられます。

⁵ 厳密には、Application Firewall のユーザレコードは、そのユーザがトラフィックマネージャの管理 UI から Application Firewall の管理 UI にアクセスするまで作成されません。同じユーザ名が既に存在する場合、新しいユーザは作成されず、代わりに既存のユーザレコードが使用されます。

ソフトウェアの更新

Application Firewall にはアップデートと呼ばれる機能があり、Application Firewall ソフトウェアコンポーネントへの更新処理を任せることができます。アップデートへは、Application Firewall の管理 UI の **Administration (管理) > Cluster Management** (クラスタ管理) タブからアクセスできます。詳細については、『**Brocade Virtual Web Application Firewall User Guide**』およびオンラインヘルプを参照してください。

CHAPTER 8 TrafficScript ルール

本章では、トラフィックマネージャで使用される TrafficScript ルールの仕組みについて説明します。新しいルールを作成して適用する方法について説明し、例をいくつか示します。

注: 一部のトラフィックマネージャ製品では、TrafficScript ルールまたは XML 機能がサポートされていないか、サポートが限定的(RuleBuilder のみ)である場合があります。完全なサポートは、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

Brocade が提供するコミュニティ Web サイトのトラフィックマネージャセクションには、TrafficScript ルールによって様々なネットワークおよびアプリケーションの問題を解決する方法について取り上げた、豊富な例とドキュメントが掲載されています。これは、<http://community.brocade.com> にあります。

概要

独自のトラフィック管理ルールを使用することにより、トラフィックマネージャの動作をカスタマイズすることができます。これらのルールは、*TrafficScript* と呼ばれるスクリプト言語を使用して作成します。

トラフィックマネージャは、新しいコネクションまたはネットワークリクエストを受信するたび、ノードからレスポンスを受信するたび、およびトランザクションの論理的完了時に TrafficScript ルールを実行します。これらのルールにより、そのコネクションで受信するデータや外部に送信するデータ、リモートクライアントアドレスなどその他の要素をチェックすることができます。

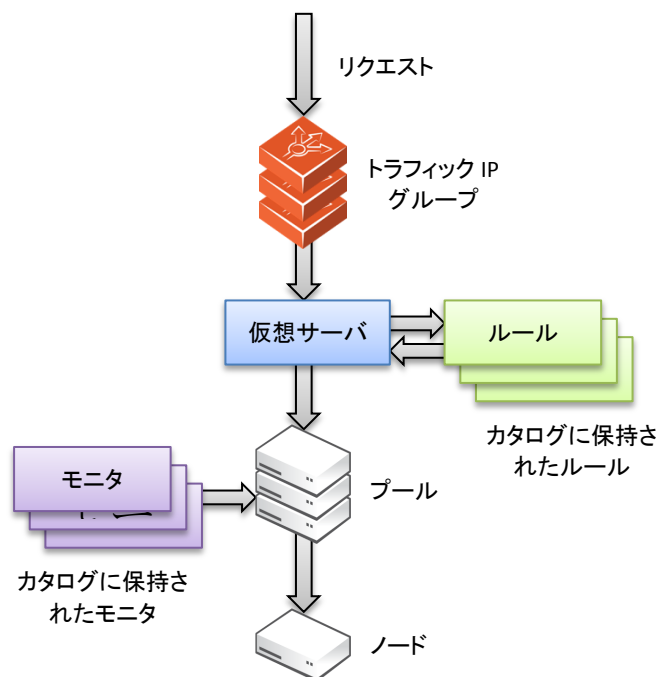


Fig 27. TrafficScript ルールによるトラフィックの解析、変更、モニタリング

TrafficScript ルールでは、リクエストやレスポンスを変更したり(例: HTTP リクエスト内の URL やヘッダの書き換え)、セッション維持パラメータを設定したり、トランザクションを選択的に記録したり、最適なプールへのリクエストのルーティング方法についてトラフィックマネージャに通知したりできます。

このように、それぞれのホストの必要条件に応じて設計されたルールを使用することで、トラフィックマネージャによるトラフィックの管理方法を正確にコントロールすることができます。

TrafficScript の例

以下の TrafficScript ルールは、HTTP リクエストに対して使用できます。このルールはリクエストを以下のように処理します。

- `www.example.co.uk` に対するリクエストを `www.example.com` に書き換える。
- `.jsp` ページに対するリクエストを一連のアプリケーションサーバ(プール名: `JSPServers`)にルーティングする。
- 「`/secure`」で始まる URL に対するリクエストは営業時間中のみ許可する。

```
# Rewrite host header if necessary
if( http.getHostHeader() == "www.example.co.uk" ) {
    http.setHeader( "Host", "www.example.com" );
}
```

```
}

$path = http.getPath();

# Give .jsp requests to the "JSPServers" pool
if( string.endsWith( $path, ".jsp" ) ) {
    pool.use( "JSPServers" );
}

# Deny access to /secure outside office hours
if( string.startsWith( $path, "/secure" ) ) {
    if( sys.time.hour() < 9 || sys.time.hour() >= 18 ){
        connection.discard();
    }
}
}
```

以下の TrafficScript ルールは、HTTP レスポンスに対して使用できます。このルールはレスポンスを以下のように処理します。

- ステータスコードが 404 または 5xx の場合、リクエストを最大 3 回再試行する。
- レスポンスに `www.example.co.uk` への参照が含まれる場合、これらの参照を `www.example.com` に変更して書き換える。

```
$code = http.getResponseCode();
if( $code == 404 || ( $code >= 500 && $code != 503 ) ) {
    # Not retrying 503s here, because they get retried
    # automatically before response rules are run
    if( request.getRetries() < 3 ) {
        # Avoid the current node when we retry,
        # if possible
        request.avoidNode( connection.getNode() );
        request.retry();
    }
}

# We're only going to process text/html responses, so
# break out of the rule if the response is of a
# different type...
if( http.getResponseHeader( "Content-Type" ) !=
    "text/html" ) break;

# getResponseBody automatically de-chunks and
```

```
uncompresses
# the response if required
$response = http.getResponseBody();

if( string.contains( $response, "www.example.co.uk" ) )
{
    $response = string.regexsub( $response,
                                "www.example.co.uk", "www.example.com",
                                "g" );
    http.setResponseBody( $response );
}
```

TrafficScript に関するドキュメント

TrafficScript 言語の構文および関数は、『**Brocade Virtual Traffic Manager: TrafficScript Guide**』に詳しく記載されています。以下の Brocade の Web サイトで、トラフィックマネージャ製品ページからこのガイドにアクセスできます。

<http://www.brocade.com/en/products-services/application-delivery-controllers/virtual-traffic-manager.html>

TrafficScript 編集ページにも、利用可能な関数へのクイックリファレンスリンクがあります。リファレンスを表示するには、ルールの編集時に **TrafficScript Reference** (TrafficScript リファレンス) リンクをクリックします。

ルールの作成方法については、167 ページの「カタログでのルールの作成」を参照してください。

トラフィックマネージャでのルールの使用

通常、仮想サーバは、リクエストを処理する適切なプールを選択するためにルールを使用します。ルールにより、リクエストの任意の部分をチェックし、場合によっては変更し、どのプールでリクエストを処理するかを決定できます。

プールに対してセッション維持情報を指定するためにルールを使用できます。ルールでは、リクエストをチェックした後に、`connection.setPersistenceKey()` 関数を使用して、セッションを維持するための文字列を設定できます。この文字列は、リクエストが属するセッションを識別するためにユニバーサルセッション維持メソッドにより使用されます。

サーバからのレスポンスをチェックして変更したり、一時的なエラーが検出された場合にリクエストを再試行したり(可能な場合)するために、ルールを使用できます。

ルールでは、仮想サーバまたはプールによってコネクションに割り当てられたクラスをオーバーライドすることにより、各コネクションに対して動作をカスタマイズして指定できます。例えば、低速なリソースへのコネクションには、長い許容レスポンス時間を設定することができます。この方法で割り当てることのできるクラス

として、サービスレベルモニタリング、セッション維持、および帯域制御があります。

サービスプロテクションクラスでルールを使用することができます。仮想サーバにサービスプロテクションクラスが関連付けられている場合、受信パケットがチェックされます。このクラスでルールを使用することにより、パケットが既知のワームやウイルスに一致するかどうかを調べることができます。このルールは、仮想サーバの主要処理を実行する前に実行されます。

セッション維持およびサービスプロテクションでのルールの使用については、それぞれ CHAPTER 12 および CHAPTER 15 で説明します。

トラフィックマネージャでのルールの使用

TrafficScript ルールは、ルールカタログに保存されます。必要に応じて、ルールカタログを使用してルールを作成したり、外部ソースからルールをアップロードしたり、ルールを変更または複製したり、不要なルールを削除したりできます。

ルールカタログにアクセスするには、**Catalogs** (カタログ) > **Rules** (ルール) をクリックします。

新しいリクエストまたはレスポンスを受信するたび、またはトランザクションの完了時に、カタログに保存されている 1 つ以上のルールを実行するよう仮想サーバを設定することができます。このようにカタログからルールを使用することで、複数の仮想サーバが同じルールを使用でき、ルールに対する変更は全ての仮想サーバで有効になります。

ルールを使用するには、まず新しいルールを作成してカタログにアップロードします。次に、仮想サーバがそのルールを使用するように設定します(**Services** (サービス) > **Virtual Servers** (仮想サーバ) > **Rules** (ルール) をクリックして、表示される設定を使用してルールを追加します)。

カタログでのルールの作成

新しいルールを作成する

1. **Catalogs** (カタログ) > **Rules Catalog** (ルールカタログ) をクリックします。
2. 「Create new rule (新しいルールの作成)」セクションで、ルールの名前を入力します。
3. TrafficScript でルールを作成する場合は、**Use TrafficScript Language** (TrafficScript 言語を使用) を選択します。RuleBuilder を使用する場合は、**Use RuleBuilder** (RuleBuilder を使用) をクリックします。

4. Create Rule (ルールの作成)をクリックします。

RuleBuilder の使用

注: ご使用のライセンスキーによっては、RuleBuilder の一部のオプションおよび機能、および TrafficScript の使用が制限される場合があります。ここで説明する機能の使用中に問題が生じた場合は、サポート窓口にお問い合わせください。ご使用のソフトウェアまたはライセンスをアップグレードするための最善の方法をご案内します。

RuleBuilder では、リクエストのチェック条件と、その条件の一部または全てが満たされた場合のアクションを選択することができます。

クライアントの IP アドレスとポート、HTTP パラメータ、クッキー、またはリクエストされた URL に基づく条件を追加できます。条件を作成するときには必ず、その項目のパラメータを入力するための追加のテキストボックスが表示されます。

The screenshot displays the RuleBuilder interface for a rule named "Content-based routing". The interface is divided into several sections:

- Header:** "Rule: Content-based routing" and a "RuleBuilder Reference" link.
- Notes:** A text area for adding notes.
- Conditions:** A section titled "Conditions" with a dropdown set to "Any" and the text "of the conditions must be met before executing the rule's actions:". It contains two conditions:
 - URL Path starts with /servlet
 - OR
 - URL Path ends with .jsp
- Actions:** A section titled "Actions" with the text "The following actions will be executed:". It contains one action:
 - Choose Pool: Servlet Runners
- Buttons:** "Apply Changes" and "Update" buttons at the bottom.
- Right Panel:** A sidebar with "Conditions" and "Actions" tabs. Under "Conditions", it lists "Requests and Responses" with sub-items: Remote IP Address, Local IP Address, Remote Port, HTTP only, SIP only, and RTSP only. Under "Actions", it lists "Responses Only" with sub-items: Response Body, HTTP only, SIP only, and RTSP only.

Fig 28. RuleBuilder のメイン画面

次に、ルールで実行する一連のアクションを選択できます。最終アクションの実行前に、リクエストの一部を変更したり、トラフィックマネージャによりログメッセージを書き込むことができます。最終アクションには、指定プールの選択やコネク

ションの破棄などがあります。複数の最終アクションを実行することはできません。また、最終アクションを決定するために、必ずしもルールは必要ありません。

例えば、以下のようなルールを作成するとします。

IF	URL Path	starts with	/servlet
OR	URL Path	ends with	.jsp
THEN	Choose Pool: Servlet Runners		

このルールでは、URL パスが/servlet で始まっていないか.jsp で終わっていないリクエスト(静的 HTML コンテンツのリクエストなど)は無視されます。このルールを適用する仮想サーバでは、別のルールを使用して、その他のページに対するリクエストを処理したり、リクエストをデフォルトプールに割り当てたりできます。

作成したルールに問題がなければ、**Update** (更新)をクリックします。

ルールを TrafficScript に変換することもできます。TrafficScript 言語では、選択できる条件やアクションの幅および構造が大きく広がります。**Preview Rule as TrafficScript** (ルールを TrafficScript として見る)をクリックすると、どのような TrafficScript ルールになるかを確認できます。**Convert Rule** (ルール変換)ボタンを使用すると、ルールが永続的に変換されます。

トラフィックマネージャのルールを初めて使用する場合は、まず **RuleBuilder** を使用して単純なルールを実装することをお奨めします。そのルールに相当する TrafficScript ルールを調べ、TrafficScript の構文を理解します。複雑なルールを作成する必要がある場合は、最初に **RuleBuilder** で単純なバージョンを作成し、**RuleBuilder** の機能の限界に達した時点でルールを TrafficScript に変換し、さらに編集を行ってください。

特殊変数

RuleBuilder のテキストボックスには、TrafficScript 関数に展開されるいくつかの特殊変数を入力することができます。これらの変数を使用すると、より動的なルールを柔軟に作成することができます。

例えば、HTTP ヘッダ「X-Forwarded-For」を「%REMOTE_IP%」という値に設定するルールを作成した場合、このヘッダは、トラフィックマネージャに接続しているクライアントの IP アドレスに設定されます。

特殊変数は以下のとおりです。

- **%REMOTE_IP%**: クライアント接続のリモート IP に置き換わります (TrafficScript 関数の `request.getRemoteIP()`)。

- `%REMOTE_PORT%`: クライアント接続のリモートポートに置き換わります(TrafficScript 関数の `request.getRemotePort()`)。

TrafficScript ルールの作成

TrafficScript 編集ページには、ルールの説明テキストおよびルール自体を入力し、コンパイル前に構文をチェックすることのできるフォームが表示されます。ルールを別のテキストエディタで作成してから、このページに貼り付けることもできます。

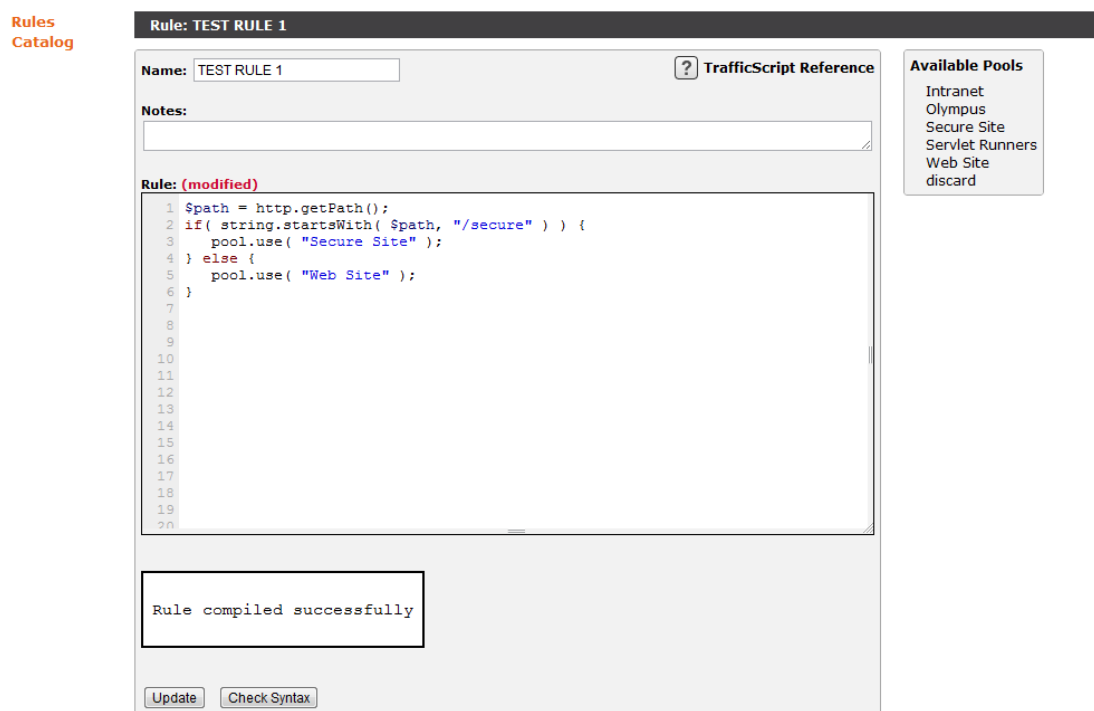


Fig 29. TrafficScript エディタでのルールの編集

TrafficScript Reference (TrafficScript リファレンス)リンクをクリックすると、関数のクイックリファレンスが表示されます。また **Check Syntax** (構文チェック)ボタンを使用すると、ルールをチェックできます。完了したら、**Update** (更新)ボタンをクリックします。

注: 一部の TrafficScript 関数と RuleBuilder の条件およびアクションは、特定のリクエストルール、レスポンスルール、またはトランザクション完了ルールでのみ有効です。例えば、リクエストのパラメータを変更する関数は、レスポンスルールで使用しても何の効果もありません(リクエストは既にノードに送信されているため)。同様に、接続に対して悪影響を伴う関数を使用しようとするトランザクション完了ルール(`http.setResponseHeader()`、`connection.discard()`など)は許可されません。各 TrafficScript 関数と RuleBuilder の条件およびアクションについては、オンラインヘルプまたは『**TrafficScript Guide**』を参照してください。

カタログへのルールのアップロード

トラフィックマネージャでは、あらかじめ作成しておいた TrafficScript ルールをカタログにアップロードできます。ルールはプレーンテキストでなければなりません。また、トラフィックマネージャはファイル名全体(拡張子も含む)をルール名に使用します。

注: アップロードされたルールファイルに対し、トラフィックマネージャは構文チェックを実行します。警告およびエラーは、**Rule Catalog** (ルールカタログ) ページ、**Diagnose** (診断) ページ、およびイベントログに表示されます。

カタログへルールをアップロードする

1. **Catalogs** (カタログ) > **Rules Catalog** (ルールカタログ) をクリックします。
2. 「**Upload an existing rule** (既存ルールのアップロード)」セクションで **Choose file** (ファイルの選択) をクリックして、ローカルファイルシステムからルールファイルを選択します。
3. アップロードするファイルと同じファイル名の既存ルールに上書きするかどうかを選択します。
4. **Upload Rule** (ルールのアップロード) をクリックします。

仮想サーバへのルールの適用

新たに追加されたルールは、まだトラフィックマネージャで使用されていません。ルールを使用するには、そのルールを仮想サーバ設定に追加します。

仮想サーバがルールを使用するように設定する

1. 仮想サーバの **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページにアクセスするには、**Services** (サービス) ボタンをクリックしてから **Virtual Servers** (仮想サーバ) タブをクリックし、仮想サーバの名前をクリックします。
2. **Rules** (ルール) リンクをクリックします。これにより、現在その仮想サーバで使用されているリクエストルール、レスポンスルール、およびトランザクション完了ルールのリストが表示されます。
3. リストに新しいルールを追加するには、各セクションの **Add rule:** (ルールを追加) オプションを使用します。ドロップダウンボックスからルールを選択し、**Add rule** (ルールを追加) をクリックします。
4. ルールは指定された順序で実行されます。最初のルールによってトランザクションについての最終決定が行われなかった場合、2 番目のルールがテストされ、以下同じように各ルールが順にテストされます。リスト内でルールを移動するには、各ルールバーの左側にある **グラブハンドル** を使用します。

多くのプロトコルでは、ルールを一度のみ実行するか(最初のリクエストまたはレスポンスに対して実行)、またはプロトコルダイアログの全てのリクエストとレスポンスに対して実行するかを指定できます。

RTSP および SIP では、トラフィックマネージャを経由する個々のリクエストまたはレスポンスに対して自動的にルールが実行されます。

HTTP はリクエストとレスポンスが一对一对応しており、*keepalive* コネクションで行われるリクエストは個別に処理されるため、HTTP 仮想サーバにはこのオプションは必要ありません。

新しいルールの効果をテストするには、テスト仮想サーバでそのルールの有効化および無効化を切り替えます。

ルールの例

コンテンツタイプによるルーティング

この例では、HTTP リクエスト内の URL パスをチェックします。URL パスの値に基づいて、リクエストを処理するプールを選択します。

リクエストの各グループに対して、それぞれ windows、sun、および linux というプールを設定します。以下のルールでは、コンテンツのタイプに応じてネットワークトラフィックが振り分けられます。

```
$path = http.getPath();
if( string.endsWith( $path, ".jsp" )) {
    pool.use( "sun" );
} else if( string.endsWith( $path, ".asp" )) {
    pool.use( "windows" );
} else {
    pool.use( "linux" );
}
```

時刻に基づくアクセス制限

この例では、特定のサービスへのアクセスを営業時間中(月～金曜日の午前 9 時～午後 6 時)のみ許可します。それ以外の時間帯に発生したコネクションは全て破棄します。

```
$dayofweek = sys.time.weekDay();
$hourofday = sys.time.hour();
```

```
# $dayofweek: Sunday is 1, Saturday is 7
# $hourofday: office hours are between 9am and 5:59pm
if( $dayofweek == 1 ||
    $dayofweek == 7 ||
    $hourofday < 9 ||
    $hourofday >= 18 ) {
    connection.discard();
}
```

実際には、警告せずにそのままコネクションを破棄するよりも、アクセス制限されたトラフィックをサーバの別の「エラープール」に送信する方が適切なことがあります。この場合、コネクションを切断する前に適切なエラーメッセージを返すようエラープール内のサーバを設定します。これを行うための手順は、負荷分散の対象となるプロトコルによって異なります。

カスタマのプライオリティ設定

この例では、HTTP リクエスト内のクッキーをチェックします。クッキーの値を使用して、選択するプールを決定します。一方のプールにはプレミアムユーザ用に予約されたマシンが含まれているため、もう一方のプールより高速です。

ある企業で、カスタマベースを「gold」会員と「silver」会員の2つに分けています。この企業では、yellow、green、blue、black、およびpurpleの5台のサーバを運用しており、「gold」カスタマのリクエストを優先的に処理にしようとしています。

ここで、yellow、green、およびblueマシンで構成される「silver」カスタマ用のstandardプールと、5台全てのサーバで構成される「gold」カスタマ用のpremiumプールの、2つのプールを作成します。すなわち、blackおよびpurpleは、「gold」カスタマのみが利用できるサーバです。

このサイトでは、クッキーログインシステムを使用しており、カスタマタイプはクッキーに符号化されています。トラフィックマネージャは、会員レベルを識別して、トラフィックを適切なプールに送信することができます。

```
$cookie = http.getHeader( "cookie" );
if( string.contains( $cookie, "gold" )) {
    pool.use( "premium" );
} else {
    pool.use( "standard" );
}
```

サービスレベルの管理

この例では、プレミアムカスタマに「premium」サービスレベルモニタリングクラスのタグを付け、そのトラフィックを「premium」プールに送信します。

プレミアムカスタマ以外のカスタマは、premium SLM クラスが許容値内で機能しているときは premium プールを共有できますが、premium SLM クラスの実行速度が遅い場合は「standard」プールに送信されます。

```
# $isPremium could be based on the presence of a
# login cookie, the contents of the shopping cart,
# or even the remote address of the client.

if( $isPremium ) {
    connection.setServiceLevelClass( "premium" );
    pool.use( "premium" );
} else {
    if( slm.conforming( "premium" ) == 100 ) {
        pool.use( "premium" );
    } else {
        pool.use( "standard" );
    }
}
```

XML トラフィックに基づくルーティング

TrafficScript では、XML ドキュメントのクエリに使用される業界標準言語である XPath を使用した XML ドキュメントの解析がサポートされます。

XML ドキュメントは、Web サービスなど SOAP ベースのプロトコルで使用され、ユーザが介入しなくても複雑なデータを自動的に交換したり解釈したりすることができます。

XML ドキュメントは、ノード⁶のツリー構造で表され、各ノードはデータ、または他のノードを保持することができます。XPath はこれらのノードを検索し、XML ドキュメントから特定のデータを抽出できます。トラフィックマネージャはこのデータを基に、トラフィックのルーティングに関する決定を行うことができます。

XPath 1.0 の仕様は <http://www.w3.org/TR/xpath> で参照できます。

⁶ これらのノードは、トラフィックマネージャのバックエンドノードとは関係ありません。

例: Google 検索リクエスト

Google™検索エンジンは、検索クエリの SOAP リクエストを受け入れる Web サービスインターフェースを備えています。Brocade Communications の検索リクエストは、以下の XML ボディデータを含む HTTP POST から成ります。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <namespace1:doGoogleSearch
xmlns:namespace1="urn:GoogleSearch">
      <key xsi:type="xsd:string">googleUniqueID</key>
      <q xsi:type="xsd:string">Brocade Communications</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">>false</filter>
      <restrict xsi:type="xsd:string"/>
      <safeSearch
xsi:type="xsd:boolean">>false</safeSearch>
      <lr xsi:type="xsd:string"/>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </namespace1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP ボディには「doGoogleSearch」ノードが含まれていることに注意してください。このノードには、検索リクエストのパラメータが保持されています。

インターネットサービスは doGoogleSearch リクエストを実行またはプロキシでき、このサービスへのトラフィックは、トラフィックマネージャを使用して管理することができます。

例えば、指定された最大結果件数によっては、doGoogleSearch リクエストを分割することが必要になる場合があります。maxResults が 100 より大きい場合、リクエストは googleLarge プールに送信され、それ以外の場合は google プールに送信されます。

TrafficScript ルールでは、xml.XPath.MatchNodeSet() および xml.XPath.MatchNodeCount() 関数を使用して、SOAP リクエストボディをクエリし、XML ノードをテストすることができます。

```
# Read the entire body of the SOAP/HTTP request
$body = http.getBody();

# XML parameters lie in the "urn:GoogleSearch" XML
# namespace:
$googlens = "xmlns:googlens=\"urn:GoogleSearch\"";

# Test for the presence of a "doGoogleSearch" node.
# If present, get the value of the "maxResults"
# parameter and choose the pool

if( xml.XPath.MatchNodeCount( $body, $googlens,
    "//googlens:doGoogleSearch" ) ) {

    $maxResults = xml.XPath.MatchNodeSet(
        $body, $googlens,
        "//googlens:doGoogleSearch/maxResults/text()" );

    if( $maxResults >= 100 ) {
        pool.use( "googleLarge" )
    } else {
        pool.use( "google" );
    }
}
```


CHAPTER 9 TrafficScript での認証のサポート

注: 製品のバージョンによっては、TrafficScript がサポートされないため、この機能を利用できない場合があります。完全なサポートは、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

概要

トラフィックマネージャでは、外部サービスに対するリモートユーザ認証がサポートされます。ユーザが、トラフィックマネージャ経由で提供されるサービスに接続するたびに、ユーザのクレデンシャルを、リモートサーバに保持されているレコードと照合して検証することができます。

これを行うには、まず認証コードカタログ(**Catalogs (カタログ) > Authenticators (認証コード)**)でリモート認証サービス定義を作成する必要があります。この方法で作成した認証コードには、TrafficScript ルール内で `auth.query()` 関数を使用することによりアクセスできます。このルールは、認証されるサービス进行处理する仮想サーバに追加できます。

認証コードの設定

現在、トラフィックマネージャでは LDAP 認証のみがサポートされています。

リモート LDAP 認証レコードは、管理 UI の **Catalogs (カタログ) > Authenticators (認証コード)** ページで作成できます。新しい認証コードを作成するには、このページのボックスに名前、ホスト、およびポートを入力します。**Create Authenticator (認証コードの作成)** をクリックするとレコードが作成され、残りの LDAP 設定キーにアクセスできるようになります。設定キーは以下のとおりです。

基本設定

設定	説明
名称	この認証コードに付ける識別名です。この名前は、認証を行う TrafficScript ルール内の <code>auth.query()</code> 関数の呼び出しで使用されます。 ここで認証コードの名前を変更すると、この認証ルールを参照している全てのルールが、新しい名前を使用するよう自動的に更新されます。
Host	接続先の LDAP サーバのホスト名または IP アドレスです。
Port	接続先の LDAP サーバのポートです。
Note	この認証コードの説明です。

LDAP 設定

設定	説明
ldap!bind!dn	「バインド」ユーザの識別名(DN)です。バインドユーザは、LDAP サーバへのアクセスおよび認証されるユーザに属するレコードの検索に使用されます。 バインドユーザには、LDAP サーバ上のユーザレコードの検索および読み取り許可が必要です。 バインドユーザが指定されていない場合、トラフィックマネージャは、認証されるユーザを検索するために匿名ログインを試行します。
ldap!bind!password	ldap!bind!dn 設定で指定されたバインドユーザのパスワードです。

設定	説明
ldap!filter	ベース DN にあるユニークなユーザレコードを抽出するために使用するフィルタです。文字列%u は、認証コードを呼び出すときに指定されたユーザ名に置き換えられます。 一般的な LDAP フィルタの例として、Active Directory の sAMAccountName=%u、一部の Unix LDAP スキーマの uid=%u などがあります。
ldap!filter!basedn	トラフィックマネージャが、認証されているユーザのレコードを検索するためのベース識別名(DN)です。 この LDAP 認証コードで認証される全てのユーザのエントリは、ここで指定する DN の下になければなりません。 一般的なベース DN として、OU=users、DC=mycompany、DC=local などがあります。
ldap!attr	トラフィックマネージャは、認証されるユーザのレコードを LDAP サーバで見つけると、そのレコードから追加情報を取得できます。この情報は、ユーザが属しているグループに基づくアクセス制限など、認証されるユーザに対して追加のチェックを実行するために使用できます。 スペースまたはカンマで区切った属性名のリストを ldap!attr 設定に指定することにより、ユーザのレコードから特定の属性を取得することができます。 ユーザのレコードから全ての属性を取得するには、ldap!attr を「*」に設定します。 この設定をブランクにすると、サーバから追加属性が読み込まれません。 ユーザのレコードから読み込んだ属性は、認証をリクエストした TrafficScript 関数の戻り値で使用できます。

設定	説明
ldap!ssl	この設定は、LDAP サーバへのコネクションを SSL で暗号化するかどうかを決定します。SSL コネクションの確立方法は、ldap!ssl!type 設定で指定できます。
ldap!ssl!type	この設定は、LDAP サーバへの SSL コネクションの確立方法を指定します。ldap!ssl が Yes (はい) に設定されている場合のみ使用されます。 使用できる方法は以下のとおりです。 ▪ LDAPS: トラフィックマネージャは、LDAP メッセージの送信前に、LDAP サーバに対してセキュアコネクションを確立します。 ▪ Start TLS: トラフィックマネージャは、LDAPv3 トランスポートレイヤセキュリティ拡張を使用して、サーバに対してセキュアコネクションを確立します。
ldap!ssl!cert	トラフィックマネージャは、リモート LDAP サーバに SSL で接続するときに、この設定で指定された認証局によってサーバの証明書が署名されていることを確認します。 サーバが、ここで指定された認証局によって署名されていない証明書を送信した場合、認証コードを使用している TrafficScript 関数にエラーが返されます。 認証局が指定されていない場合、サーバの証明書は有効になりません。

TrafficScript ルールの設定

仮想サーバでリモート認証を使用するには、認証ルールをリクエストルールとして仮想サーバに割り当てる必要があります。このルールでは、以下の引数を指定して `auth.query()` 関数を呼び出す必要があります。

```
auth.query( authenticator, user, [password] );
```

この関数は、設定された `authenticator` に対して `user` に関する情報をクエリし、`password` が指定された場合は、その値がそのユーザのレコードに設定されているパスワードと一致していることを確認します。その後、検証結果に従って設定された `OK` および `Error` の 2 つの値を含むハッシュを返します。結果には、クエリの対象となったユーザの識別名など、認証コードによって返された追加情報を含めることができます。詳細については、『**TrafficScript Reference**』を参照してください。

TrafficScript ルールは、管理 UI の **Catalogs** (カタログ) > **Rules** (ルール) ページで作成します。詳細については、CHAPTER 8 を参照してください。

以下の例は、`auth.query()` 関数を使用して、作成済みの「`ldap`」という認証コードに基づいてユーザを検証する方法を示しています。

```
# Verify the user's password using an LDAP
# authenticator called 'ldap'
$auth = auth.query( "ldap", $user, $pass );
if( $auth['Error'] ) {
    log.error(
        "Error with authenticator 'ldap': " .
        $auth['Error']
    );
    connection.discard();
} else if( !$auth['OK'] ) {
    # Unauthorised
    http.sendResponse( "403 Permission Denied",
        "text/html", "Incorrect username or password",
        ""
    );
}

# Allow through members of the 'admin' group using
# the 'group' attribute returned by the authenticator
if( $auth['group'] != "admin" ) {
    http.sendResponse( "403 Permission Denied",
        "text/html",
```

```
        "You do not have permission to view this page",  
        ""  
    );  
}
```

仮想サーバの設定

適切な設定を行ってルールを設定したら、認証を有効化する仮想サーバにそのルールを割り当てる必要があります。

1. 管理 UI の **Services** (サービス) > **Virtual Servers** (仮想サーバ) セクションに移動し、認証を有効化する仮想サーバを選択します。
2. **Rules** (ルール) セクションをクリックします。
3. **Request Rules** (リクエストルール) で、認証ルールを選択して「*Add Rule*(ルールを追加)」をクリックします。

CHAPTER 10 Java Extensions

本章では、利用可能な Java のサポートについて説明します。TrafficScript から Java を呼び出すことができるため、さらに強力なルーティングと容易なトラフィック管理が可能になります。

詳細については、『*Brocade Virtual Traffic Manager: Java Development Guide*』を参照してください。

Java について

Java はプラットフォームに依存しないオブジェクト指向の開発言語で、幅広い開発者、ライブラリ、およびアプリケーションに利用されています。トラフィックマネージャでは、TrafficScript で Java Extensions を利用することで、さらに柔軟にトラフィックを操作することができます。これらの Java Extensions は TrafficScript ルールから呼び出すことができます。

トラフィックマネージャ Java API を使用することにより、ユーザおよびインターネットサービスプロバイダは、トラフィックマネージャの機能を拡張することができます。

TrafficScript で Java API を使用すると、以下のような機能を利用できます。

- **コンテンツ圧縮** - 専用の Java ライブラリを使用することにより、XML/HTML の処理が向上します。
- **追加ライブラリ** - 付加価値ソリューションとして提供され、トラフィックマネージャの「プラグイン可能な」拡張機能として機能する ISV ライブラリにより、新たな機能を利用できるようになります。
- **認証** - 外部 Web サーバを使用しなくても、RADIUS/TACACS+/LDAP などを利用できるようになります。

Java Extension の呼び出し

Java Extensions は、TrafficScript または RuleBuilder ルールから呼び出します。

Java Extension をトラフィックマネージャにインポートすると、その Java Extension と同じ名称の RuleBuilder 基本ルールが作成されます。このルールによりインポート元の Extension が呼び出されるため、ルールでトラフィックをチェック、変更、およびルーティングできるようになります。この基本ルールは、仮想サーバが実行するリクエストルールまたはレスポンスルールのリストに追加できます。

多くの場合、特定の Java Extension を全てのリクエストやレスポンスに対して呼び出す必要はありません。例えば、ある Java Extension は特定タイプのリクエストま

たはレスポンスのみを処理します。Java Extension は、以下のように条件を指定して RuleBuilder または TrafficScript ルールから呼び出すことができます。

```
if( http.getPath() == "/serverstatus" ) {  
    java.run( "ServerStatus" );  
}
```

トラフィックマネージャで Java を使用するための設定

このセクションでは、トラフィックマネージャで Java Extensions を作成、設定、および実行するためのプロセスについて説明します。このプロセスの詳細については、『**Java Development Guide**』を参照してください。

必要条件

トラフィックマネージャで Java Extensions を使用するには、Sun Java Run-time Environment (JRE)バージョン 1.5 以上をインストールする必要があります。これより古いバージョンはトラフィックマネージャでサポートされません。

トラフィックマネージャ仮想/クラウドインスタンスには、Java Extensions を実行するために必要な Java ソフトウェアが含まれています。

Java Extension のコンパイル

Java Extensions をトラフィックマネージャ用にコンパイルするには、以下のリソースが必要です。

- **Java Development Kit (JDK):** Java コンパイラが含まれています。このコンパイラは <http://java.sun.com> からダウンロードできます。
- **Java Servlet API ライブラリ:** このライブラリは `$ZEUSHOME/zxtm/lib/servlet.jar` にあります。また、**Java Extensions catalog** (Java Extensions カタログ)のリンクからダウンロードすることもできます。
- **トラフィックマネージャ Java Extensions API ライブラリ:** このライブラリは `$ZEUSHOME/zxtm/lib/zxtm-servlet.jar` にあります。また、**Java Extensions catalog** (Java Extensions カタログ)のリンクからダウンロードすることもできます。

Java Extension をコンパイルするには、以下のコマンドを実行します。

```
$ javac -cp servlet.jar:zxtm-servlet.jar MyServlet.java
```


これにより、*MyServlet.class* というクラスファイルが作成されます。上記のコマンドでは、コンパイル元のディレクトリに *serolet.jar* および *zxtm-serolet.jar* がコピーされていることを前提としています(このディレクトリにコピーされていない場合は、これらのファイルのパスも指定する必要があります)。

Java Extension を、その他の必要なクラスと一緒に単一の jar ファイルにパッケージ化することもできます。トラフィックマネージャは、使用する Java Extensions の jar ファイルを自動的に検索します。

トラフィックマネージャへの Java Extensions のロード

Java Extension クラスと jar ファイルをトラフィックマネージャの **Java Extensions** カタログに追加する必要があります。

Extension をアップロードするには、**Catalog (カタログ) > Java Extensions** ページに移動し、**Upload (アップロード)** セクションでクラスまたは jar ファイルを指定します。Extension が、Java の配布パッケージに含まれていない他の Java jar ファイルに依存する場合、これらのファイルもカタログにアップロードする必要があります。

Java Extensions Catalog

✓ File 'Counter.jar' was uploaded successfully.

Java Extensions Catalog

A Java Extension can manipulate connections, in a similar way to a TrafficScript rule. Extensions are invoked from TrafficScript. Please see the user manual for full instructions on building Java Extensions. You will also need to download the **Java Servlet API** and **Zeus Java Extensions API** files.

Java Extensions Catalog [? Java API documentation](#)

Java Extensions that are usable from a TrafficScript rule. Click on the name of the extension to view more details and edit its initialization parameters.

Extension	Path	Used By	Select (all / none)
com.zeus.TestServlets.Counter	Counter.jar	Unused	☐
com.zeus.TestServlets.PoolPicker	PoolPicker.jar	Unused	☐

☐ Confirm operation

Java Libraries & Data Catalog

Any uploaded non-Java Extensions files are shown here, including other Java class/jar files.

No additional files have been uploaded.

Upload Extension / Data File

Choose to upload either a jar file containing your Java Extension code, a single class file or data files that your Java Extensions are going to use. Class files will automatically be put in the correct directory depending on their package.

File: No file chosen

☒ Automatically create TrafficScript rule

☐ Overwrite if file already exists: ☐

Fig 30. Java Extensions Catalog (Java Extensions カタログ)ページ

トラフィックマネージャの Java Extension Runner の設定

トラフィックマネージャには、Java Extensions をホストする Java Extension Runner という Java ヘルパアプリケーションが含まれています。トラフィックマネージャがこのヘルパアプリケーションを初期化および実行する方法をコントロールするには、

管理インターフェースの **System** (システム) > **Global Settings** (グローバル設定) > **Java Extension Runner** セクションの設定を使用します。

Java 実行可能ファイルを指定するには、インターフェースの **[[**セクションの「**java!command**」フィールドに、実行可能ファイルの名前(デフォルトパスにない場合はそのパスも)と、JRE の実行時に使用するオプションを指定します。

- **java!command:** コマンドライン引数を含む Java 実行可能ファイルのパスです。
- **java!classpath:** Java クラスが保存されているフォルダのコロンで区切ったリストです。
- **java!lib:** Java Extensions で使用されるライブラリを自動的に検索するフォルダを指定します。トラフィックマネージャは、このディレクトリから Java Extensions をロードするわけではありません。
- **java!max_conns:** 同時に許容される Java リクエストの最大数を定義します。この数を超えるリクエストはキューイングされ、以前のリクエストが完了するまで処理されません。この設定は CPU コアごとに行います。マルチコアプロセッサの場合は、この設定に実際のプロセッサ数を乗じる必要があります。
- **java!session_age:** Java セッションを維持するタイムアウトを定義する値(秒単位)です。

設定を確認するには、**Diagnose** (診断) ボタンを押して **Cluster diagnosis** (クラスタ診断) タブに移動し、「**Java extensions**」の行が緑色で表示されていることを確認します。これで、トラフィックマネージャで Java Extensions を実行することができます。

CHAPTER 11 プロトコルのサポート

トラフィックマネージャは、TCP コネクションまたは UDP データグラムに含まれるアプリケーショントラフィックの負荷分散と処理を行います。TrafficScript、負荷分散、セッション維持、帯域制御などの多くの機能は、全てのプロトコルタイプに同じように適用することができます。

適切な場合は、特定プロトコルに固有の追加の TrafficScript 関数を含めることで、そのプロトコルでのトラフィックの管理およびコントロールを容易に行えるようになります。

基本的な TCP プロトコル

「Generic Server First」、「Generic Client First」、および「Generic Streaming」プロトコルは、トラフィックマネージャで利用できる最も基本的な L7 プロトコルタイプです。これらのプロトコルタイプは、トラフィックが特定フォーマットに適合していることが想定されないため、カスタムプロトコルや単純な TCP コネクションを管理する場合に役立ちます。

Server First プロトコル

Server First は最も単純な TCP プロトコルタイプです。トラフィックマネージャはクライアントからコネクションを受け取ると、直ちに TrafficScript リクエストルールを実行し、その後すぐにバックエンドサーバに接続します。以降、トラフィックマネージャは、両方のコネクション(クライアント側とサーバ側)を監視し、受信したトラフィックを一方のコネクションからもう一方のコネクションへと渡します。

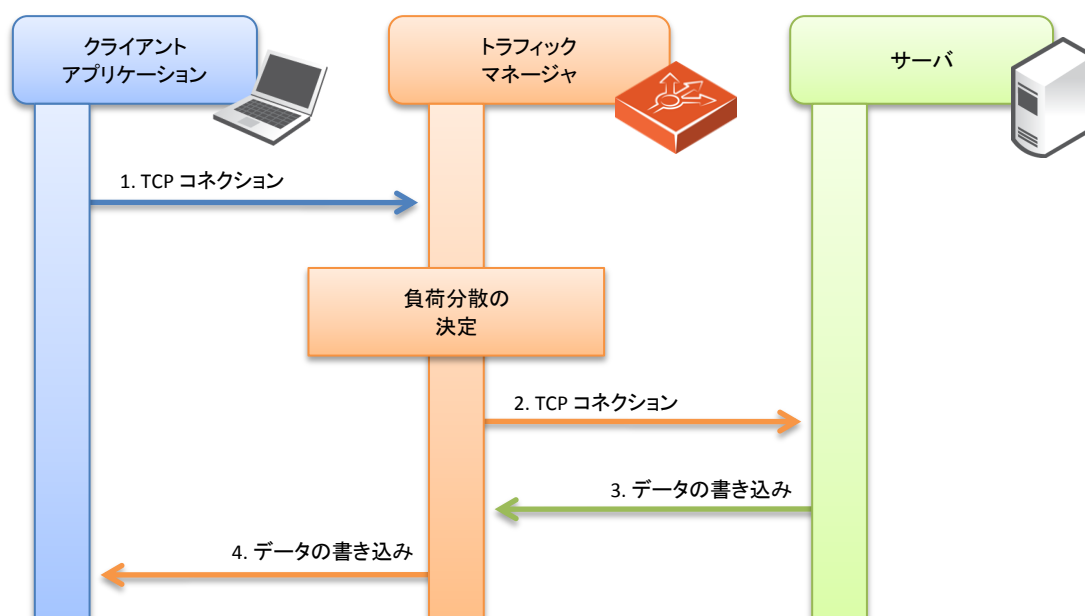


Fig 31. Server First プロトコルの負荷分散

Server First プロトコルには、Time (サーバがレスポンスデータを書き込んでから直ちに接続を切断)などのワンショットプロトコルや、MySQL (サーバがパスワード入力のためのダイアログを開く)などの複雑なプロトコルがあります。

「Generic Server First」仮想サーバプロトコルタイプは、接続の確立後にサーバからやり取りを開始する場合に最適です。実際には、多くのプロトコルは設計上「Generic Client First」の方が適しています。

Client First プロトコル

Client First は Server First を改良したもので、接続したクライアントが、サーバからのレスポンスの前にリクエストを送信するプロトコルに適しています。「Generic Client First」プロトコルタイプでは、クライアントのリクエストをチェックして、使用サーバノードの選択に使用するサーバプールを選択することができます。

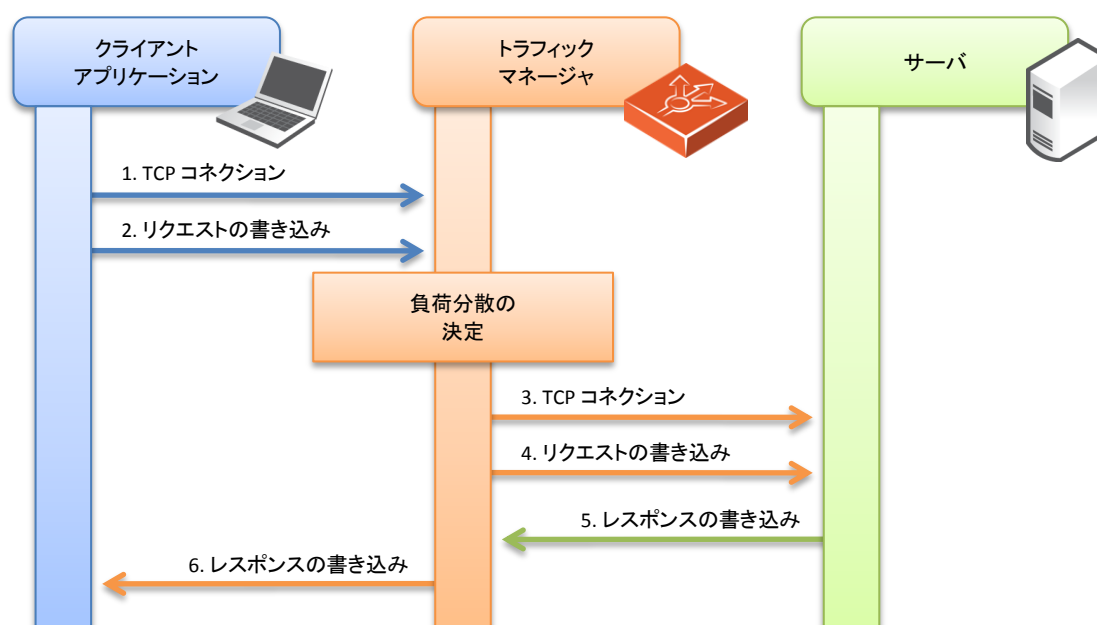


Fig 32. Client First プロトコルの負荷分散

トラフィックマネージャには、クライアント接続が確立され、データを受信した時点で初めてアラートが報告されます。その後、トラフィックマネージャは、Server First の方式に従い、TrafficScript ルールを実行し、バックエンドに接続し、データを双方向に渡します。

ルール処理の詳細

トラフィックマネージャは、TrafficScript リクエストルールとレスポンスルールを交互に実行します。クライアントから最初のデータを受信したときにリクエストルールを実行します。リクエストルールで `request.getLine()` や `request.get()` などの関数を使用されている場合、クライアントからさらにデータを読み取るためにリクエストルールが中断することがあります。

リクエストルールが完了すると、トラフィックマネージャは、以降クライアントから受信する全てのデータをサーバに転送します。また、サーバからデータ(すなわち「レスポンス」)を受信すると、直ちにレスポンスルールを実行します。これらのレスポンスルールも中断することがあります。

以降、トラフィックマネージャは、サーバからの全てのデータをクライアントに転送します。その後、クライアントからのデータを待機し、リクエストルールを再度実行します。このように、トラフィックマネージャは、リクエストデータを待機してリクエストルールを実行し、レスポンスデータを待機してレスポンスルールを実行する処理を交互に繰り返します。

「Run Every (毎回実行)」ではなく「Run Once (一度実行)」として設定されているルールは、最初のサーバデータまたはクライアントデータに対してのみ実行され、以降やり取りされるデータについては実行されません。

「サーババナー」を使った **Server First**

「サーババナー」を使った **Server First** は、「**Server First**」を別の形で最適化したもので、SMTP など、接続時にバナーをブロードキャストするサーバ用に調整されています。「サーババナーを使った **Server First**」では、クライアントのリクエストをチェックし、それに基づいて負荷分散の決定を行うことができます。

クライアントが接続すると、トラフィックマネージャは、直ちに設定されている

「**Server First** バナー」をクライアントに書き込み、その後通常の **Client First** 接続と同じように処理を進めます。また、サーバが送信する最初の行またはデータ(改行で終了)をスラップして破棄します。

このプロトコルタイプでも、TrafficScript ルールを使用してクライアントのリクエストデータをチェックしてから負荷分散の決定を行うことができます。

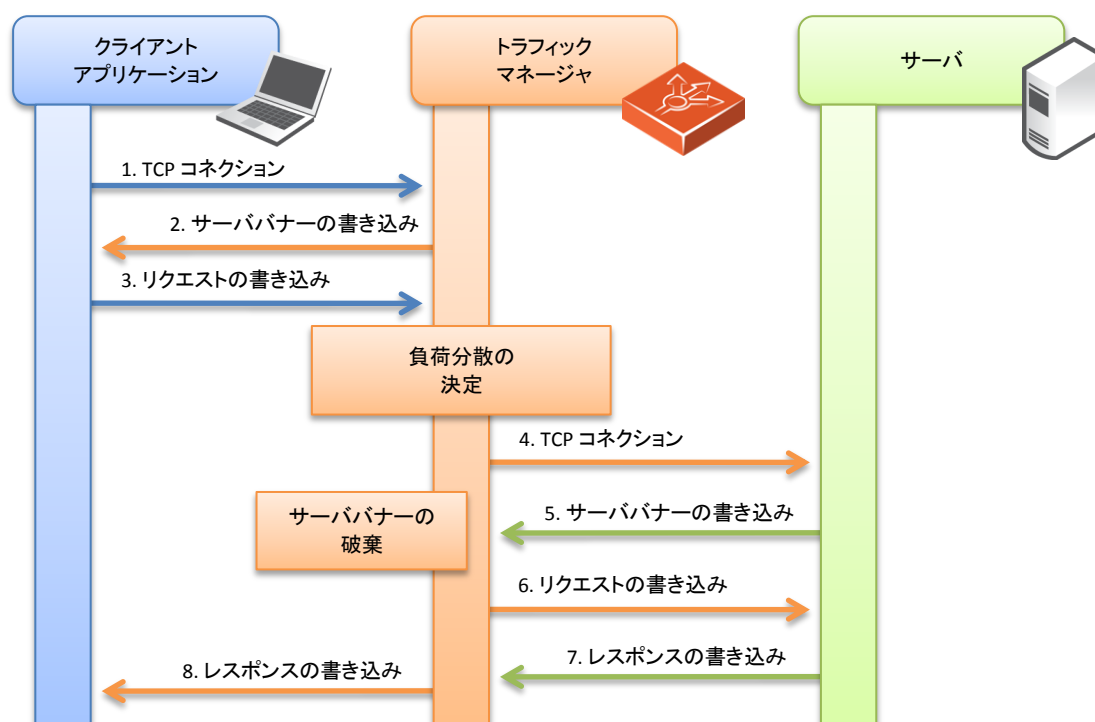


Fig 33. サーババナーを使った Server First の負荷分散

この動作を設定するには、「Generic Server First」プロトコルタイプを選択し、仮想サーバの設定の **Connection Management** (コネクション管理) 設定でバナーメッセージを設定します。

Generic Streaming プロトコル

サーバがクライアントから送信された全てのメッセージに対してレスポンスを返す必要のないプロトコルを使用しているか、TrafficScript でのデータ処理にさらに柔軟性が必要な場合、「Generic Streaming」オプションを選択できます。このサーバタイプでは、コネクションの確立後、クライアントまたはサーバのどちらから最初のメッセージを送信することができます。また、そのコネクションでデータを受信するたびに TrafficScript ルールを呼び出すことも可能です。

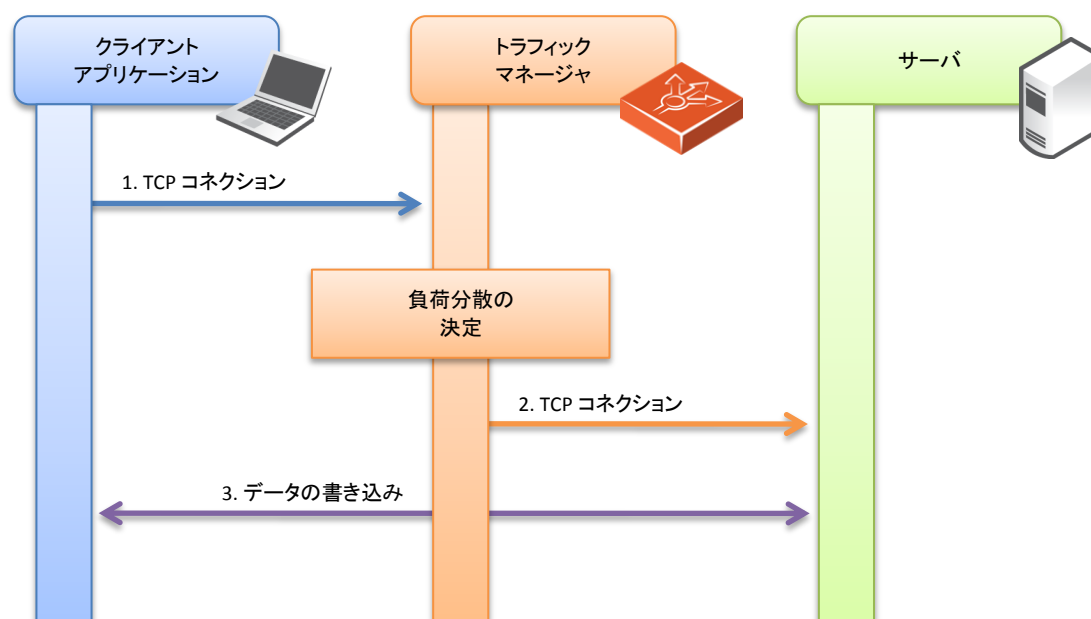


Fig 34. Generic Streaming プロトコルを使った負荷分散

このオプションは、リクエスト-レスポンスというセマンティックが存在しないプロトコルに適しています。コネクションのどちら側も最初のメッセージを書き込むことができ、レスポンスは必ずしも必要でない、または想定されていません。クライアントがコネクションによってデータを書き込むたびに TrafficScript リクエストルールが実行され、同様にサーバがコネクションによってデータを書き込むたびにレスポンスルールが実行されます。

HTTP

トラフィックマネージャの HTTP 仮想サーバプロトコルタイプは、様々な点が HTTP トラフィック用に最適化され、特化されています。

- トラフィックマネージャは、クライアント側とサーバ側のコネクションを個別に管理します。サーバへの確立済み TCP コネクションおよび新たな TCP コネクションの数を減らすために、可能な場合は、サーバ側で **keepalive** コネクションを再利用します。これにより、サーバが処理しなければならない同時接続数が最小限に抑えられ、性能および処理能力の面で大きなメリットがもたらされます。

HTTP POST などの機能を使用する場合やコンテンツ長が欠落している場合は、**keepalive** コネクションを安全に使用できません。トラフィックマネージャはこのような状況を検出し、適宜新たにコネクションを確立します。また、NTLM 認証では、**keepalive** が明示的に有効化されていることが必要になります。

- トラフィックマネージャでは、トラフィック管理ルールにおいて HTTP トランザクションのリクエスト-レスポンスという単純な性質に専念できるように、**keepalive** の使用やパイプラインングが管理者から見えなくなっています。

全ての HTTP トランザクションは、keepalive コネクションによるものかどうかに関わらず、個別に処理されます。

- トラフィックマネージャは、gzip および deflate コンテンツ圧縮やチャンクトランスファエンコーディングなどの HTTP 符号化を自動的に処理します。例えば、TrafficScript 関数の `http.getResponseBody()` または Java Extension によって HTTP レスポンスをチェックする場合、トラフィックマネージャは、レスポンスを容易に操作できるよう自動的にデチャンクし、解凍します。
- トラフィックマネージャには、HTTP リクエストおよびレスポンスの処理を容易に行えるよう特化された HTTP TrafficScript 関数が豊富に用意されています。例えば、基礎となる符号化や、これらの操作を複雑にしている多様なフォーマットおよびプロトコルの詳細を理解しなくても、これらの関数を利用することで、HTTP クッキーの操作、HTTP ヘッダ(ホストヘッダなど)の読み取り、URL の処理などを行うことができます。

トラフィックマネージャでは、HTTP/2 プロトコルのクライアント側サポートも提供されます(RFC7540 と RFC7541 を参照してください)。HTTP/2 は、HTTP/1.x に置き変わるもので、特に単一の TCP コネクションでトランザクション多重化をサポートすることにより、待ち時間の長いコネクションでページロード時間を短縮するように設計されています。

トラフィックマネージャは、HTTP/2 をサポートしていますが、以下の注意点があります。

- 一部のブラウザは、暗号化されていないコネクションで HTTP/2 をサポートしていません。HTTP/2 を使用してサービスにアクセスできるユーザ数を最大にするには、該当する仮想サーバに対してトラフィックマネージャの SSL 復号化機能を有効化することをお奨めします。
- セキュアなコネクション上の HTTP/2 では、TLS 1.2 以降、SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256 暗号、P256 楕円曲線が要求され、コネクションの確立後の再ネゴシエーションはサポートされません。これらのオプションは、デフォルトで有効ですが、以前のリリースからトラフィックマネージャをアップグレードした後に無効になることがあります。廃止された暗号または旧式の TLS バージョンを使用しているクライアントは、HTTP/2 が有効化された仮想サーバに接続するときに、HTTP/1.x リクエストのみを送信できます。
- トラフィックマネージャは、HTTP/2 トランザクションで Brocade Web Accelerator コンテンツ最適化をサポートしていません。
- トラフィックマネージャでは、クライアントから受信した HTTP/2 データが処理前に HTTP/1.1 に変換され、仮想サーバのプールノードに転送されます。これにより、バックエンドノード自体に HTTP/2 のサポートを要求せずに、トラフィックマネージャで HTTP/2 機能を引き続き使用できます。

- TrafficScript 関数は、HTTP/2 コネクションで透過的に動作します。注:
 - クライアントコネクションで使用されている HTTP バージョンを調べるには、TrafficScript 関数「`http.getClientVersionNumber`」を使用します。
 - HTTP 仮想サーバで「`connection.discard`」を使用する場合は、注意することをお奨めします。HTTP/2 では単一のコネクションでトランザクションを多重化できるため、コネクションを破棄すると、継続中のトランザクション全てが中断されます。
 - TrafficScript 関数「`connection.close`」は、全ての HTTP/1.x サービスと HTTP/2 サービスで廃止されました。この関数を使用する全ての TrafficScript ルールを、「`http.sendResponse`」と「`http.discardClientKeepalive`」を使用するように変更することをお奨めします。詳細については、『*Brocade Virtual Traffic Manager: TrafficScript Guide*』（Brocade の Web サイト：<http://www.brocade.com> から入手可能）を参照してください。

クライアントで使用されている HTTP バージョンを調べるには、**Activity (アクティビティ) > Details (詳細)** でコネクションを選択するときに「Request Tracing (リクエストトレーシング)」または「Request Details (リクエストの詳細)」のいずれかを確認します。また、`%r` または `%H` ロギングマクロを使用して、HTTP/2 の使用についてレポートするか、HTTP/2 リクエストが実行されるたびに増加する SNMP カウンタ「`virtualserverTotalHTTP2Requests`」を参照します。

SSL

SSL プロトコルを使用してトラフィックを管理するようトラフィックマネージャを設定すると、トラフィックマネージャは、自動的にトラフィックを復号化しません。SSL プロトコルタイプは、SSL トラフィックを変更しない SSL パススルーに使用します。

「SSL セッション ID」セッション維持タイプを使用してプール内で SSL コネクションの負荷分散を行うと、同じ SSL セッション ID のコネクションを同じバックエンドサーバに送信することができます。一般に、SSL サーバでは相互に SSL セッション ID が共有されないため、使用を検討することが重要です。そのため、クライアントが SSL セッション ID を再利用しようとしても、新しいサーバに送信され、再度 SSL セッションについてのネゴシエーションが必要になります。この処理には大きな処理能力が伴い、SSL サーバファームに負荷がかかります。実際、SSL セッション ID セッション維持を使用しない場合、クラスタに SSL サーバを追加すると、SSL セッションのネゴシエーションを頻繁に行わなければならないとなり、性能が低下します。

注: SSL トラフィックを復号化し、内部プロトコル(HTTP など)を使用して処理するには、内部プロトコルを使用するよう仮想サーバを設定し、仮想サーバの設定で SSL 復号化を有効化する必要があります(SSL の設定の詳細については、CHAPTER 4 および CHAPTER 13 の「SSL 復号化」セクションを参照)。

SSL ハンドシェイクの保護

SSL コネクションのネゴシエーションを行うときにクライアントが試行するピアハンドシェイクメッセージのサイズに、制限を設定することができます。これは、サービス妨害(DoS)攻撃の予防に効果があります。

管理 UI で、**System (システム) > Global Settings (グローバル設定)** ページに移動します。SSL Configuration (SSL 設定) セクションに、**ssl!max_handshake_message_size** という設定キーがあります。

この設定の値によって、SSL コネクションで受け入れられる SSL ハンドシェイクメッセージの最大サイズが決まります。ハンドシェイクメッセージのサイズがこの設定を超えると SSL コネクションに失敗し、イベントログには DoS 攻撃の可能性があるとして記録されます。デフォルト値は 10KiB で、連続して 5 つの証明書を受け入れることができるので、通常の SSL 設定ではこれで十分です。この値に 0 を指定すると、トラフィックマネージャで処理される SSL ハンドシェイクメッセージのサイズは無制限になります。

SSL コネクションの再ネゴシエーション保護

SSL 3.0 と TLS (トランスポートレイヤセキュリティ) のどちらのプロトコルでも、クライアントまたはサーバがセキュアコネクションの再ネゴシエーションを開始できます。これを利用して、コネクションを管理する新しい暗号化パラメータを設定することができます。

再ネゴシエーションの際には、クライアントとサーバとの対話にプレフィックスとしてトラフィックを挿入しようとする攻撃者によって、セキュアコネクションが傍受に対して脆弱になる可能性があります。中間者(MitM)攻撃として知られるこのメカニズムは、攻撃者から送信される初期データを本来のクライアントと同じエンティティから送信されたように見せかけてターゲットサーバを欺くことで、攻撃者側から新しいクライアントコネクションを確立するために使用できます。

また、再ネゴシエーションをサービス妨害(DoS)攻撃に使用することも可能です。クライアントよりサーバに対して高い負荷をかけることになるからです。クライアントは、高い頻度で再ハンドシェイクを繰り返しリクエストすることで、サーバに極端に大きな処理負荷をかけます。

RFC-5746 (<http://tools.ietf.org/html/rfc5746>) では、このような動作を実行しているクライアントに対し、暗号化した再ネゴシエーションを試行することによって MitM 攻撃を防ごうとする TLS 拡張が導入されました。ただし、この拡張が導入されたか

らといって必ずしも DoS 攻撃を予防できる、あるいは攻撃から身を守れるとは限りません。

トラフィックマネージャには、再ネゴシエーションハンドシェイクの処理方法を選択できる設定(ssl!ssl3_allow_rehandshake)があります。この設定は、管理 UI の **System** (システム) > **Global Settings** (グローバル設定) ページの **SSL Configuration** (SSL 設定) セクションにあります。使用できるオプションは以下のとおりです。

設定	説明
Always allow	クライアントが RFC-5746 をサポートしていない場合でも、再ネゴシエーションを常に許可します。ただし、第三者による介入に対しては全く保護されなくなります。この設定は、システムが MitM または DoS 攻撃を受けやすくなるため、 お奨めできません 。
Allow safe re-handshakes [デフォルト設定]	クライアントが RFC-5746 をサポートする場合と、コネクションを介してデータがまだ送信されていない場合がこれに該当します。前述した攻撃は先頭に付けられたプレーンテキストで構成されているため、データの受信前であれば、再ハンドシェイクは安全です。クライアントの観点から言えば、このオプションの用途は限定的です。再ハンドシェイクは実際には最初のハンドシェイクであるため、コネクションが攻撃を受けているかどうか、クライアントには判断できません。したがって、このオプションは主に、アップグレード前でまだ RFC-5746 をサポートしていないクライアントに後方互換性を持たせることを目的としています。この設定では再ハンドシェイクが許可されるため、システムはやはり前述の DoS 攻撃を受けやすい可能性があります。
Only if client uses RFC 5746 (Secure Renegotiation Extension)	再ネゴシエーションハンドシェイクは、RFC-5746 で定義されている セキュアな再ネゴシエーション がクライアントでサポートされている場合にのみ許可されます。この設定では再ハンドシェイクが許可されるため、システムはやはり前述の DoS 攻撃を受けやすい可能性があります。

設定	説明
Never allow	再ネゴシエーションハンドシェイクで全ての試行を拒否します。これは、MitM と DoS のどちらの攻撃に対しても保護されるため、最もセキュアなオプションです。ただし、システムの一部がクライアント側の再ネゴシエーションに依存している場合には、サービスエラーが発生する可能性があります。

SSL 3.0/TLS 再ハンドシェイクが許可されている場合、`ssl!ssl3_min_rehandshake_interval` 設定を使用して、単一の SSL 3.0/TLS コネクションのハンドシェイク間で許可される最低時間間隔(ミリ秒単位)を定義することができます。この時間間隔は、開始ハンドシェイクが始まると開始され、再ハンドシェイクの開始時点で必ずリセットされます。ハンドシェイクの最低間隔を無効化するには、0 (ゼロ)に設定します。

トラフィックマネージャが SSL サーバとして機能している(仮想サーバで `ssl_decrypt` が「Yes (はい)」に設定されている)場合、再ネゴシエーションがトラフィックマネージャ自体により `TrafficScript` 関数 `ssl.requirecert()` を用いて実行されると、その結果生じる再ネゴシエーションはレート制限の対象にはなりません。

SMTP (Simple Mail Transport Protocol)

SMTP は、機能が 1 つ追加された Server First プロトコルです。セキュリティを強化するために、SSL クライアントは、SMTP コネクションを SSL (TLS とも言う)で暗号化するようにリクエストすることができます。

SMTP プロトコルを使用するように仮想サーバを設定した場合、そのプロトコルは通常の Generic Server First プロトコルのように動作します(詳細については、187 ページの「Server First プロトコル」を参照してください)。Server First バナーで仮想サーバを設定することをお奨めします。詳細については、189 ページの「サーババナー」を使った Server First」を参照してください。

SMTP 仮想サーバで SSL を有効化すると、トラフィックマネージャでは、`smtp!expect_starttls` という設定が **Virtual Server** (仮想サーバ) > **Edit** (編集) > **SSL** ページに追加されます。これは以下のように設定します。

- **No** (いいえ): 仮想サーバは、通常の方法で SSL トラフィックを復号化します。言い換えれば、コネクションの最初に、トラフィックが SSL で暗号化されていると想定します。
- **Yes** (はい): 暗号化されていないかのように、プレーンテキストとしてトラフィックが処理されます。トラフィックマネージャは、クライアントからの

「STARTTLS」SMTP コマンドを監視し、このコマンドを受信すると、SSL ハンドシェイクを開始してクライアントのコネクションを SSL にアップグレードします。

トラフィックマネージャは、プール設定で SSL 暗号化が有効化されていない限り、全てのトラフィックを暗号化されていない状態(プレーンテキスト)でバックエンド SMTP サーバに転送します。SSL 暗号化を有効化した場合、トラフィックマネージャは、STARTTLS SMTP コマンドを使用せずに、最初からコネクションを暗号化します。

FTP

FTP プロトコルは、クライアントからサーバにファイルを転送するために使用します。FTP セッションは以下のコネクションから成ります。

- クライアントからリモートサーバのポート 21 (一般的な設定)へのコントロールコネクション。このコネクションは、トラフィックマネージャによって受け取られ、プロキシされます。
- コントロールコネクションによって送信されたコマンドの結果、即座に確立される 1 つ以上のデータコネクション。これらのコネクションは、サーバからクライアントに対して確立される場合(「アクティブモード」)、クライアントからサーバに対して確立される場合(「パッシブモード」)があります。

トラフィックマネージャの「FTP」仮想サーバタイプでは、追加のデータコネクションを確立するためにリクエストを傍受して管理する完全な FTP プロキシが行われます。トラフィックマネージャは、コントロールコネクションをプロキシするときと同様にデータコネクションをプロキシし、アクティブモードとパッシブモードの両方のデータ転送をサポートします。

このプロキシでは、long (RFC 1639)バージョンおよび extended (RFC 2428)バージョンを含む、一般的な FTP データコネクションコマンドが全て完全にサポートされます。ただし、RFC 4217 暗号化コントロール命令文(「AUTH TLS」、「AUTH SSL」、「CCC」)はサポートされず、コントロールストリームから削除されます。

注: **consider source port** (ソースポート考慮)は、FTP 仮想サーバが使用しているトラフィック IP アドレスでは使用しないでください(CHAPTER 6 の「トラフィック IP グループの作成」セクションを参照)。

FTP セキュリティ

FTP はセキュアなプロトコルではありません。パブリックインターネットで一般的な「匿名」ファイル転送には、パスワードが使用されず、パスワードが必要な場合もプレーンテキストで送信されるため、盗聴される可能性があります。このような状況が懸念される場合は、SCP、WebDAV、SSL-wrapped FTP (本章の「SSL-wrapped FTP」セクションを参照)など他のプロトコルを使用してください。

もう 1 つの懸念事項は、クライアントとサーバのデータコネクション準備を行うダイアログを盗聴者がのぞき見し、コネクションに入り込んでトラフィックの監視側とのコネクションを容易に開始できることです。

例えば、アクティブな FTP コネクションを確立するときに、クライアントはサーバに対して、データコネクションを確立するためにサーバが接続すべきローカルクライアントポートを通知します。このとき、盗聴者がサーバより先にクライアントのローカルポートに入り込んで接続し、クライアントから送信されるデータをキャプチャする(またはクライアントに偽のファイルを返す)可能性があります。

FTP プロトコルには、コネクションを認証してこのような状況を防ぐためのセキュリティ対策が組み込まれていませんが、現代の FTP クライアントおよびサーバの多くは、受信するコネクションの送信元 IP アドレスをチェックし、通信相手であるリモート FTP エージェントから送信されたものかどうかを確認します。トラフィックマネージャでは、この動作がデフォルトで有効化されます。**Virtual Server (仮想サーバ) > Connection Management (コネクション管理)**に移動し、このページの **FTP-Specific Settings (FTP 固有設定)** セクションで以下の設定を確認してください。

- **ftp_force_client_secure** (デフォルトは「yes」): 全てのパッシブデータコネクションの接続元がクライアントの IP アドレスであることを確認します。
- **ftp_force_server_secure** (デフォルトは「yes」): 全てのアクティブデータコネクションの接続元がサーバの IP アドレスであることを確認します。
- **port_range**: データコネクションに使用する宛先ポートの範囲を指定します。これらのポートは、例えばクライアントがトラフィックマネージャに対してパッシブコネクションを確立できるように、介在するファイアウォールによって許可されていない場合があります。

FTP ソースポート

FTP 仕様では、FTP サーバデータコネクションに、FTP サービスポートより 1 つ小さい番号のソースポートを使用することが推奨されています。一般的な FTP サービスポートはポート 21 であることから、推奨されるデータソースポートはポート 20 です。

ところが実際には、大部分の FTP クライアントおよびサーバでこの推奨内容が無視されています。これは、このようなポートの使用がセキュリティに影響するにも関わらず、保証や認証のための追加措置が提供されていないためです。デフォルトでは、トラフィックマネージャは、トラフィックマネージャから開始する FTP データコネクション用に、高いポート番号をランダムに選択します。

ソースポートの指定

ftp_data_source_port 仮想サーバ設定(**Connection Management (コネクション管理) > FTP Settings (FTP 設定)**セクション)を使用して、使用するソースポートを指定することができます。この操作は、一般に、アップストリームでファイアウォールを

使用して、外部に送信されるコネクションのうち明示的ポリシー(ソースポートと IP など)によって許可されたもの以外をブロックする場合に必要になります。

Unix/Linux 環境の FTP ソフトウェアは、ポート 20 など低い番号のポート(1024 未満)からリクエストを送信するための特定のパーミッションを必要とします。デフォルトでは、トラフィックマネージャは、早い段階でこれらの許可を破棄するため(最小権限の原則)、**ftp_data_source_port** を低い値に設定しようとするとう失敗します。低いポート番号を使用するには、**Global Settings** (グローバル設定)ページの **System Settings** (システム設定)セクションで **ftp_bind_data_low** 設定を有効化します。これにより、トラフィックマネージャはパーミッションの制限を放棄するため、低い番号のポートにバインドできるようになります。

現代の Linux プラットフォームでは、低い番号のポートへのバインド権限のみが維持され、Unix 系製品では、完全なルート権限が維持されます。

SSL-wrapped FTP (FTPS)

トラフィックマネージャは、RFC 4217 以前の FTPS (「Implicit FTPS」または「SSL-wrapped FTP」と呼ばれます)の使用をサポートしています。この実装では、コントロールポートは最初から暗号化されます。また、一般にデータコネクションも暗号化されますが、必要に応じてプレーンテキストで動作させることも可能です。

SSL-wrapped FTP 用の仮想サーバプロトコルタイプは「FTP」です。この場合、仮想サーバとプールは、コントロールチャネル通信を復号化または暗号化するよう個別に設定することができます。

コントロールチャネルのコントロール

- **ssl_decrypt:** クライアント側コネクションで SSL 復号化を有効化するには、**Virtual Server** (仮想サーバ) > **SSL Decryption** (SSL 復号)ページで **ssl_decrypt** 設定を有効化します。SSL 復号化の設定の詳細については、CHAPTER 13 の「**SSL 復号化の設定**」セクションを参照してください。
- **ssl_encrypt:** バックエンドサーバへの送信前にコントロールコネクションを暗号化するには、**Pool** (プール) > **SSL Settings** (SSL 設定)ページで **ssl_encrypt** 設定を有効化します(CHAPTER 13 の「**SSL 暗号化**」セクションを参照)。

トラフィックマネージャの FTP プロキシでは、確立済みの SSL-wrapped FTP セッションに RFC 4217 の暗号化コントロール命令文(「AUTH TLS」、「AUTH SSL」、「CCC」)が干渉できないよう、これらの命令文が無効化されます。

データチャネルのコントロール

トラフィックマネージャは、データチャネルを復号化することも、変更せずにそのままにすることもできます。この動作は、**Virtual Server** (仮想サーバ) > **SSL Decryption** (SSL 復号)の **ftp!ssl_data** 設定でコントロールします。

- `ftp!ssl_data` 設定を有効化すると、仮想サーバは、コントロールポートに使用されているものと同じ公開証明書を使用して全てのデータコネクションを復号化します。この場合、`ssl_decrypt` を使用して、SSL を復号化するよう FTP 仮想サーバも設定しなければならないことに注意してください。
- 無効化した場合、仮想サーバはデータを変更せずにそのまま渡します。バックエンドサーバがデータコネクションで SSL の使用を希望している場合、クライアントは、トラフィックマネージャを中間プロキシとして使用して、バックエンドサーバと直接ネゴシエーションを行うことができます。

サーバ側では、以下のいずれかの場合に、トラフィックマネージャがデータコネクションを暗号化することに注意してください。

- 仮想サーバの `ssl_decrypt` 設定が無効化され(仮想サーバが SSL を復号化しない)、プールの `ssl_encrypt` 設定が有効化されている(プールはデータチャネルのサーバ側を暗号化)場合。この場合、クライアントとのデータ通信はプレーンテキストで行われます。
- 仮想サーバの `ssl_decrypt` および `ftp!ssl_data` 設定の両方が有効化され(仮想サーバが SSL を復号化する)、プールの `ssl_encrypt` 設定が有効化されている(プールはデータチャネルのサーバ側を暗号化)場合。この場合、クライアントとのデータ通信は暗号化されます。

SSL-wrapped FTP の使用例

使用例1: サーバとクライアントが **SSL-wrapped FTP** をサポート

この使用例では、関与する全てのパーティが SSL-wrapped FTP をサポートして使用し、トラフィックマネージャは、クライアントをサーバに接続する負荷分散プロキシとして機能します。トラフィックマネージャは、コントロールコネクションをチェックできるよう内部的に復号化し、その後再度暗号化します。また、データコネクションについても復号化して再度暗号化することがオプションで可能です。

- SSL 復号化(仮想サーバ: `ssl_decrypt`)および SSL 暗号化(プール: `ssl_encrypt`)を有効化します。これにより、トラフィックマネージャは、コントロールチャネルをチェックしてプロキシし、データコネクションを動的に確立することができます。
- オプション: `ftp!ssl_data` を有効化します。無効化した場合、クライアントとサーバは、データコネクションで暗号化を使用するかどうかについてネゴシエーションを行うことができます。有効化した場合、トラフィックマネージャは、データ転送に SSL を使用するようクライアントにリクエストします。

注: `ftp!ssl_data` を有効化すると、以下の2つのメリットが得られます。1つ目のメリットとして、全てのコネクションが暗号化され、トラフィックマネージャの仮想サーバ証明書と照合して認証されるため、証明書の管理が容易になります。2つ目のメリットとして、トラフィックマネージャは全てのクライアントに対し、データコネクションに SSL コネクションを使用するよう強制します。

ftp!ssl_data を使用しない場合、トラフィックマネージャとバックエンドサーバ間で証明書を同期する必要があり、クライアントおよびサーバがプレーンテキストで(暗号化されていない)データコネクションするのを防ぐことができません。ただし、これらのメリットには、トラフィックマネージャシステムで追加の SSL 処理が生じるという代償を伴います。

使用例2: クライアントが **SSL-wrapped FTP** をサポートし、サーバは非サポート

この設定は、SSL をサポートしていない FTP サーバや、トラフィックマネージャと FTP サーバが非常に近接しているためトラフィックマネージャおよび FTP サーバ間のネットワークコネクションをセキュアにする必要がない状況で使用できます。

- SSL 復号化(仮想サーバ: `ssl_decrypt`)は有効化しますが、SSL 暗号化(プール: `ssl_encrypt`)は有効化しません。この場合、トラフィックマネージャは、クライアントからの SSL-wrapped FTP を復号化し、プール内の FTP サーバにプレーンテキストで渡します。
- 推奨: **ftp!ssl_data** を有効化します。無効化した場合、クライアントとサーバでは常にプレーンテキスト(暗号化されていない)コネクションによってデータ転送が行われます。有効化した場合、トラフィックマネージャは、データ転送に SSL を使用するようにクライアントに強制します。

この使用例の方法では、サーバを変更することなく、FTP サービスを SSL-wrapped FTP に簡単にアップグレードすることができます。

Real Time Streaming プロトコル

Real-Time Streaming Protocol (RTSP)を使用すると、クライアントは、オーディオおよびビデオサーバなどの連続メディアの単一または複数の時間同期ストリームを確立し、コントロールすることができます。RTSP に基づくアプリケーションは、マルチメディアサーバにおいてビデオのようなりモートコントロールパネルとして機能します。同一カンファレンス内で Session Initiation Protocol (SIP)と RTSP を組み合わせ使用するサーバもあります。

RTSP は、QuickTime、MPlayer、VLC、Windows Media Player、RealPlayer などのクライアントアプリケーションで使用されています。

トラフィックマネージャは、RTSP トラフィックを認識して管理できるため、ユーザは、このタイプのトラフィックを管理するメディアサーバのプールで負荷分散を行うことができます。

下図に、RTSP コネクションの一般的なレイアウトを示します。RTSP クライアントは、トラフィックマネージャを介して RTSP サーバへの TCP コネクションを確立し、このコネクションによって RTSP リクエストを送信します。これはコントロールチャネルです。メディアデータ自体は、UDP で個別に送信することも(一般的なシナリオ)、同じコネクションを使用して TCP で多重化することもできます。

または、HTTP を使用してデータ配信することも可能です。基礎となる転送プロトコルとして UDP または TCP を使用する場合、クライアントとサーバが接続の確立時にネゴシエーションを行うことに注意してください。この設定はトラフィックマネージャからは行いません。

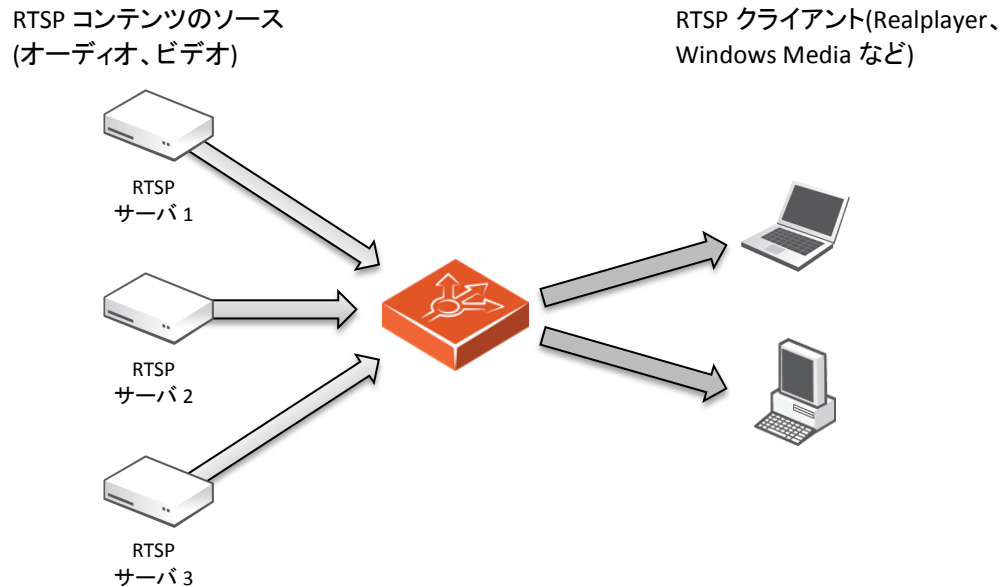


Fig 35. 標準的な RTSP コンテンツトラフィックの負荷分散を行うトラフィックマネージャ

RTSP 自体は、メディアの転送のみをコントロールします。メディアトランザクション全体では、以下のようなプロトコルが関与します。

- **RTP:** Real-time Transport Protocol。データを転送するために使用されます。
- **RTCP:** Real Time Control Protocol。RTP ストリームに関する追加情報を提供するために使用されます。
- **RDT:** Real Data Transport。データストリームを転送する、Real Networks 社独自のデータ配信フォーマットです。

機能的には、RTSP の処理にトラフィックマネージャを使用すると、以下のようなメリットが得られます。

- サーバの負担が軽減され、RTSP サーバのプールにわたる負荷分散が可能
- TrafficScript ルールを使用することで、性能のカスタマイズおよび向上が可能
- サーバでの障害発生時にセッション維持を提供

RTSP サービスの設定

デフォルトでは、RTSP はポート 554 で動作します。RTSP サービスをトラフィック マネージャで管理するには、ポート 554 (または、適切な場合は別のポート) で監視する仮想サーバを作成し、プロトコルタイプ「RTSP」を選択します。

UDP でコンテンツをストリーミングする場合、仮想サーバでは追加のポートを使用する必要があります。使用ポートの範囲を指定する必要がある場合、RTSP 仮想サーバの **Connection Management** (コネクション管理) 設定を使用します。RTSP-**Specific Settings** (RTSP 固有設定) セクションで、ポート範囲を変更できます。このとき、可能な限り広い範囲を入力することをお奨めします。

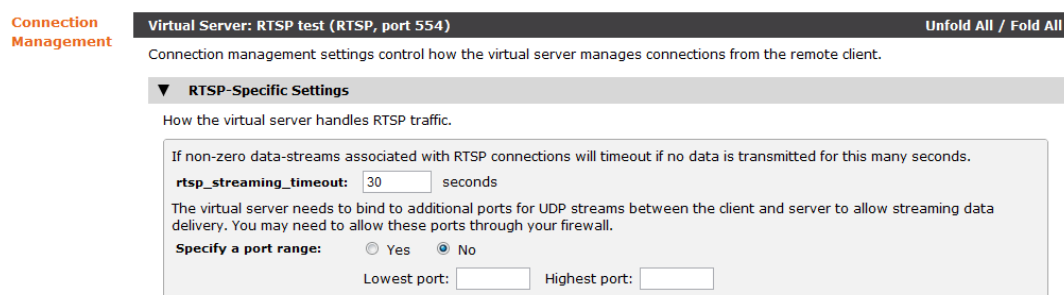


Fig 36. RTSP のポート範囲の設定

必要なポートの合計数は、転送方法と使用サーバ、クライアントが使用するプレイヤー、ストリーミングするファイルのフォーマットによって異なります。例えば、アクティブなクライアントが 100 台含まれ、クライアントあたり最大 8 個のポートが必要な場合、合計 800 個のポートが必要になります。

Session Initiation Protocol

Session Initiation Protocol (SIP) は、単一または複数の参加者とのセッションを開始、コントロール、変更、または終了することのできるアプリケーションレイヤコントロールプロトコルです。通常、SIP は、電話やビデオ通話、マルチメディア配信やカンファレンスのストリーミングに使用されます。SIP プロトコルでは、以下のような機能が提供されます。

- ユーザロケーション
- ユーザの可用性
- ユーザの機能
- セッションの設定
- セッションの管理

SIP の機能

このセクションでは、SIP の機能について詳しく説明します。

- **ユーザロケーション:** ユーザが SIP デバイス(SIP 電話など)をアクティブ化すると、デバイスによりその現在位置が SIP サーバに登録されます。以降、サーバのドメインでそのユーザとコンタクトを取ることができます。各ユーザは、異なる場所にある複数のデバイスを同時にサーバに登録することができます。

誰かがそのユーザとコンタクトを取ろうとすると、サーバにメッセージが届きます。サーバは、そのユーザについて登録されている場所から、メッセージの転送先として1つまたは複数の場所を選択します。

- **ユーザの可用性:** SIP デバイスはメッセージを受信した場合に、いくつかの方法でレスポンスを返すことができます。レスポンスは、例えばユーザが着呼に応答するなど、ユーザ入力によって決めることができます。また、例えばデバイスが使用中の場合に「ビジー」レスポンスを返すなど、レスポンスを自動的に生成することも可能です。このレスポンスをどのように処理するかはサーバが決定し、例えばレスポンスを呼び出し元に返したり、元のメッセージを別の場所に送信したりできます。

ユーザがサーバにデバイスを登録していない場合は、使用不可と見なされます。

- **ユーザの機能:** SIP デバイスは、他の SIP デバイスをクエリし、そのデバイスが持つ機能を学習することができます。その後、両方のデバイスでサポートされている機能を使用して通信を行うことができます。
- **セッションの設定:** SIP を Session Description Protocol と組み合わせて使用し、2台のデバイス間に直接通信チャネルを確立することができます。この通信チャネルでは、通話の場合は音声データなどセッションデータが送信されます。このとき、SIP チャネルは、通話を切断するリクエストなどのコントロール情報を送信するために使用されます。通常、セッションデータは Real-Time Transport Protocol (RTP)を使用して送信されます。
- **セッションの管理:** SIP は、デバイス間のセッションの終了およびセッションの転送をコントロールします。確立されたセッションのパラメータを変更することもできます。

SIP は、IPv4 および IPv6 をサポートし、TCP および UDP で動作します。

トラフィックマネージャと SIP プロトコル

トラフィックマネージャは、SIP プロキシサーバに SIP トラフィックを負荷分散し、可用性、信頼性、および拡張性に優れた SIP インフラストラクチャを構築します。下図に、トラフィックマネージャが含まれる SIP ネットワークの例を示します。一般に、SIP 電話はユーザの個々のロケーションに設置されますが、トラフィックマネージャ、SIP プロキシサーバ、およびロケーションデータベースは、SIP サービスプロバイダが所有します。

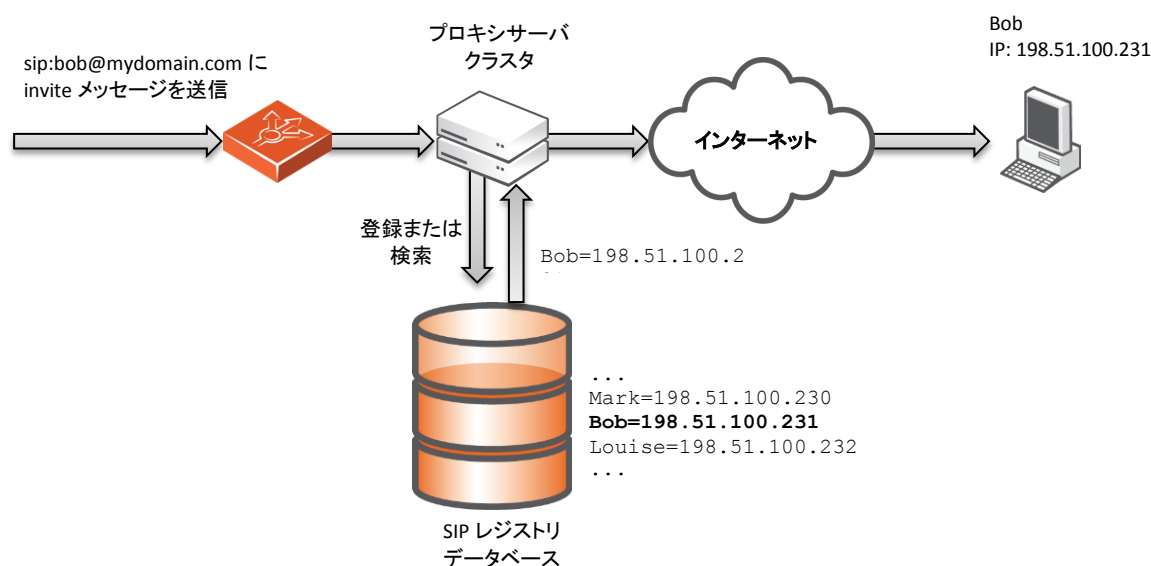


Fig 37. トラフィックマネージャを使用した SIP による IP 通話の処理の詳細

トラフィックマネージャは、SIP トラフィックの処理に関して以下の機能を提供します。

- カスタマイズ可能な負荷分散アルゴリズムを使用して、SIP プロキシサーバに SIP リクエストを負荷分散します。
- セッション維持を維持するため、同じセッションに属する全てのリクエストが同じプロキシサーバに送信されます(デフォルトでは、トラフィックマネージャは SIP 「Call-ID」 ヘッダに残ります)。
- RuleBuilder および TrafficScript ルールを使用して、トラフィックマネージャを通過する SIP リクエストおよびレスポンスパケットを変更します。
- 受信したリクエストをプロキシサーバに送信せずにレスポンスを送信します。
- 使用中のプロキシサーバのヘルスをモニタリングし、障害発生時にはトラフィックを他のプロキシサーバにルーティングします。
- 全てのクライアントがプライベートネットワーク内に存在する場合、全ての SIP データおよびセッションデータのゲートウェイとして機能します。
- DoS 攻撃および悪意のあるデータから保護します。

プロキシサーバでトラフィック管理をサポートするための設定

トラフィックマネージャは、Oracle/Weblogic SIP Server や SIP Express Router (SER)など一連の SIP プロキシサーバの前段階で使用するのが最も効果的です。この構成で動作させるには、プロキシサーバは、ドメインクライアントが利用するトラフィックマネージャを管理対象として認識する必要があります。例えば、

sip:user@example.com がトラフィックマネージャの IP に名前解決される場合、プロキシサーバは example.com ドメインを管理する必要があります。こうすることで、プロキシサーバは、リクエストをそのドメインに転送せずに処理するようになります。プロキシサーバがリクエストをこのドメインに転送してしまうと、トラフィックマネージャとプロキシサーバ間でリクエストがループすることになります。

または、プロキシサーバがドメインをサポートしていない場合は、管理 UI の **Virtual Servers** (仮想サーバ) > **Edit** (編集) > **Connection Management** (コネクション管理) ページに移動し、「rewrite_request_uri」オプションを有効化します。このオプションを有効化すると、受信する各リクエストの URI を、リクエストの送信先プロキシサーバの URI で置き換えるようトラフィックマネージャに指示が出されます。

トラフィックマネージャでの SIP サービスの設定

新しい SIP サービスは、**Manage a new service** (新規サービスを管理) ウィザードで作成できます。サービスの適切な名前を指定し、SIP over TCP または SIP over UDP プロトコルのいずれかを選択します。デフォルトの SIP ポートは、TCP および UDP のどちらのサービスの場合も 5060 です。このウィザードにより、指定した詳細に基づいて新しい SIP 仮想サーバとプールが設定されます。

SIP 仮想サーバを正常に動作させるには、この仮想サーバが、SIP リクエストの送受信に使用される全ての IP アドレスを監視する必要があります。これを行うには、**Virtual Servers** (仮想サーバ) > **Edit** (編集) > **Basic Settings** (基本設定) ページに移動し、**Listening on** (監視) 設定を見つけます。「All IP addresses (全 IP アドレス)」が選択されていることを確認します(これがデフォルトです)。

2 台の仮想サーバを作成し、それぞれに UDP プロトコルと TCP プロトコルを割り当てることにより、同じポートで両方のプロトコルを使用することもできます。

重要: SIP インフラストラクチャ内でトラフィックマネージャを使用する場合、クライアントへの SIP リクエストの送信時に、SIP プロキシサーバへの登録時とは異なるコネクションが使用されます。そのため、SIP リクエストが電話まで通過できるようクライアントのファイアウォールを設定する必要があります。一部のファイアウォールではこの設定が自動的に行われますが、その他のファイアウォールは、ポート 5060 のリクエストをユーザの電話に転送するよう設定する必要があります。

トラフィックマネージャでの SIP 動作モード

SIP 仮想サーバは、**SIP Routing** (SIP ルーティング) と、**SIP Gateway** (SIP ゲートウェイ)、および **Full Gateway** (フルゲートウェイ) の 3 つの動作モードで実行できます。この設定は、コントロール情報(SIP メッセージ)およびセッションデータ(一般には音声およびビデオ通信の RTP データ)にトラフィックマネージャがどの程度関与するかに影響します。

SIP サーバの動作モードを定義するには、トラフィックマネージャ管理インターフェースでサーバを編集し、**Connection Management** (コネクション管理) > **SIP-specific settings** (SIP 固有設定) をクリックします。

▼ SIP-Specific Settings

How the virtual server handles general SIP traffic.

The virtual server should discard a SIP transaction when no further messages have been seen within this time.
sip_transaction_timeout: seconds

When timing out a SIP transaction, send a 'timed out' response to the client and, in the case of an INVITE transaction, a CANCEL request to the server.
sip_timeout_messages: ☒ Yes ☐ No

The mode that this SIP virtual server should operate in.
sip_mode: ☐ SIP Routing ☒ SIP Gateway ☐ Full Gateway ...

Replace the Request-URI of SIP requests with the address of the selected back-end node.
sip_rewrite_uri: ☐ Yes ☒ No

Should the virtual server follow routing information contained in SIP requests. If set to **No**, requests will be routed to the chosen back-end node regardless of their URI or Route header.
sip_follow_route: ☒ Yes ☐ No

The action to take when a SIP request with body data arrives that should be routed to an external IP.
sip_dangerous_requests: ☒ Send the request to a back-end node
☐ Send a 403 Forbidden response to the client
☐ Forward the request to its target URI (dangerous)

SIP clients can have several pending requests at one time. To protect the traffic manager against DoS attacks, this setting limits the amount of memory each client can use. When the limit is reached new requests will be sent a 413 response. If the value is set to 0 (zero) the memory limit is disabled.
sip_max_connection_mem: bytes

Fig 38. SIP 仮想サーバの設定

各動作モードについて以下に説明します。

- **SIP Routing** (SIP ルーティング) - このオプションは、SIP プロキシサーバに SIP セッションを負荷分散する場合に選択します。最初の SIP リクエストとそれに対する全てのレスポンスはトラフィックマネージャを経由するため、RuleBuilder および TrafficScript ルールでチェックし、操作することができます。

トラフィックマネージャは、リクエストに **Record-Route** ヘッダフィールドを追加しません。その結果、同じセッションに属する以降のリクエストはトラフィックマネージャを経由せず、最初のリクエストが送信されたサーバに直接送信されます。

すなわち、一般的な SIP セッションでは、INVITE リクエストとそれに対する全てのレスポンスはトラフィックマネージャを経由し、このセッションを管理する BYE リクエストなどの以降のリクエストはトラフィックマネージャを経由しません。

- **SIP Gateway** (SIP ゲートウェイ) - このオプションは、特定のセッションに属する全ての SIP リクエストをチェックする場合に選択します。トラフィックマネージャ

ジャは、リクエストがトラフィックマネージャを経由するときに **Record-Route** ヘッダフィールドを追加するため、同じセッションに属する以降のリクエストは、トラフィックマネージャを経由してルーティングされます。これらの全てのリクエストとそれに対するレスポンスは、**RuleBuilder** および **TrafficScript** ルールで操作することができます。

一般的な SIP セッションでは、**INVITE** や **BYE** などの全てのコントロールメッセージがトラフィックマネージャを経由しますが、セッションデータ自体は経由しません。これは、トラフィックマネージャでの SIP 仮想サーバのデフォルト動作モードです。

- **Full Gateway (フルゲートウェイ)** - このオプションは、全てのセッションデータにトラフィックマネージャを経由させる場合に選択します。このモードは、全ての SIP クライアントがトラフィックマネージャと同じネットワーク上に存在し、トラフィックマネージャがインターネットへのゲートウェイとして機能している場合に便利です。

このモードでは、**INVITE** や **BYE** など全てのコントロールメッセージがトラフィックマネージャを経由します。トラフィックマネージャは、リクエストまたはレスポンスのボディデータに含まれる **Session Description Protocol** 情報を変更し、セッションデータもトラフィックマネージャを経由するようにします。

セッションデータに使用するポート範囲を指定できます。また、セッションデータコネクションでデータが送信されないときに、トラフィックマネージャがこのコネクションを切断するまでの時間を指定することもできます。

SIP のその他の設定

Connection Management (コネクション管理) ページにも、トラフィックマネージャの動作に影響を与える設定がいくつか表示されます。これには以下の設定があります。

- **sip_transaction_timeout** - SIP リクエストとレスポンスはトランザクションにグループ化されます。SIP のトランザクションには、**INVITE** トランザクションと非 **INVITE** トランザクションの 2 つのタイプがあります。**INVITE** トランザクションは、**INVITE** リクエストとそれに続く一連のレスポンスから成り、最後のレスポンスが「**200 OK**」レスポンスでない場合は、その後に **ACK** リクエストが続きます。一方、非 **INVITE** トランザクションは、リクエストとそれに続く 0 個以上のレスポンスから成ります。クライアントは同じコネクションで複数のトランザクションを送信できます。

特定のトランザクションに属するメッセージが、**sip_transaction_timeout** に指定された時間にわたって検出されないと、トラフィックマネージャはそのトランザクションに関連付けられたリソースを再利用します。

トランザクションが完了しないままタイムアウトになると、トラフィックマネージャは、**408** リクエストタイムアウトレスポンスをクライアントに送信します。

- **sip_rewrite_uri** - このオプションを設定すると、トラフィックマネージャに対し、受信したリクエストの URI を、トラフィックマネージャが選択したバックエンドノードの URI に置き換えるよう指示が出されます。例えば、受信したリクエストの URI が「sip:bob@example.com」で、トラフィックマネージャが example.com ドメインを管理しているとします。また、バックエンドノードの IP が 192.0.2.1 で、その SIP サーバがポート 5070 で実行されている場合、バックエンドノードに送信されるリクエストは sip:bob@192.0.2.1:5070 になります。

この設定は、SIP プロキシがユーザロケーションの検索時にユーザのドメインをチェックしない場合に便利です。

- **sip_follow_route** - このオプションを設定すると、トラフィックマネージャに対し、受信したリクエストに含まれるルーティング情報に従うよう指示が出されます。Route ヘッダが存在する場合、リクエストはそれに相当する宛先に送信され、Route ヘッダが存在しない場合は、URI で指定された宛先に送信されます。URI がトラフィックマネージャを指している場合、リクエストは通常どおりプールに送信されます。

この動作は、**sip_dangerous_requests** 設定でオーバーライドできます。例えば、受信したリクエストにボディデータと Route ヘッダが含まれる場合、危険なリクエストをバックエンドノードに送信するよう **sip_dangerous_requests** が設定されていれば、Route は無視され、リクエストはプールに送信されます。

sip_follow_route を No に設定すると、全てのリクエストがプールに送信され、選択したノードに対応する Route ヘッダが追加されます。

- **sip_dangerous_requests** - SIP リクエストは、それぞれ独自のルーティング情報を持ちますが、この情報が、中間マシンを介してリモートコンピュータを攻撃するために悪用される可能性があります。トラフィックマネージャは、次の宛先が外部 IP で、リクエストにボディデータが含まれている場合、そのリクエストを危険なものを見なします。

トラフィックマネージャは、危険性が疑われるリクエストを以下の方法で処理することができます。

- 1) リクエストをバックエンドノードに送信できます。このオプションは、SIP プロキシサーバが、トラフィックマネージャの IP に名前解決されないドメインを管理している場合に便利です。これは、危険性が疑われるリクエストに対するデフォルト設定です。
- 2) 403 Forbidden レスポンスをクライアントに返すことができます。このオプションは、トラフィックマネージャのドメインにアドレス指定されていない全てのリクエストを拒否する場合に便利です。
- 3) リクエストをそのターゲット URI に転送できます。このオプションを選択すると、SIP リクエストは通常どおりにルーティングされます。この設定は、外部に送信される SIP トラフィックを仮想サーバで処理する場合に使用します。

- **sip_max_connection_mem** - クライアントは、同じコネクションで複数の SIP リクエストを送信し、それに対するレスポンスを任意の順序で受信することができます。クライアントでレスポンス待ちのアクティブなリクエスト数が過剰にならないようにするために、トラフィックマネージャは、各コネクションに割り当てることのできる最大メモリ量を制限できます。

この設定では、トラフィックマネージャが各コネクションに許容するメモリ量を指定します。このメモリ量を超えると、新しいリクエストは「413 Request Entity Too Large」レスポンスで拒否されます。この値を 0 に設定すると、メモリ制限が解除されますが、トラフィックマネージャが DoS 攻撃を受けやすくなります。

UDP ベースの SIP サーバとの通信

一般に、UDP ベースの SIP サーバは、TCP ベースの SIP サーバと同様にリクエストに対してレスポンスを返します。すなわち、元のリクエストを受信したポートからレスポンスを送信します。この動作が必ずしも必要なわけではありませんが、SIP サーバには、リクエストを受信したものとは異なるポートや IP からレスポンスを返すものがあります。

トラフィックマネージャでは、**Pools (プール) > Edit (編集) > Connection Management (コネクション管理)** ページで「**udp_accept_from**」を設定することにより、この動作を確実に行うことができます(UDP サービスでプールが使用される場合)。この設定を使用して、トラフィックマネージャが UDP リクエストに対するレスポンスをどのように受信するかを以下のオプションから選択できます。

- リクエストの送信先の IP アドレスとポートのみ。
- リクエストの送信先の IP アドレスのみ。ただし任意のポートを使用可能。
- 特定の IP アドレスセット。ただし任意のポートを使用可能。
- 任意の IP アドレスと任意のポート。

特定の IP アドレスセットからのみレスポンスを受け入れるよう選択した場合、表示されるボックスに CIDR マスク(198.51.0.0/16 など)を入力します。

これらの設定では、セキュリティと互換性との間に様々なレベルのトレードオフが生じます。複数の IP アドレスからのレスポンスを許容すると、不正なレスポンスを受け入れる可能性が高くなり、セキュリティリスクが増大する可能性があります。特定の SIP では、適切に通信するためにこの機能が必要になる場合があります。必要条件に最も適したオプションを選択してください。

関連する設定が SIP (over UDP)ヘルスモニタにあります。詳細については、267 ページの「内蔵ヘルスモニタ」を参照してください。

クライアントから UDP リクエストを受信した後にサーバノードに許可される最大時間を制限するには、「`udp_response_time`」を必要な秒数に設定します。「0」の値(デフォルト)に設定すると、タイムアウトカウンタが無効化されます。つまり、レスポンスを送信しないサーバは、障害が発生したと自動的にマークされます。

重要: IPv4 アドレスに送信されたリクエストでは、IPv4 アドレスからのみレスポンスが返されることが想定されます。一方、IPv6 アドレスに送信されたリクエストでは、IPv4 または IPv6 形式でのレスポンスが許容されます。

CHAPTER 12 セッション維持

本章では、トラフィックマネージャにより、クライアントとバックエンドサーバ間のセッション維持(スティッキーセッションとも言う)がどのように提供されるかについて説明します。

セッション維持とは

クライアントから送信されるリクエストの多くのクラスは、バックエンドサーバのプールで負荷分散することができます。これにより、同じクライアントから送信された複数のリクエストを、サービスを中断することなくバックエンドサーバに分配できます。ただし、ローカルに保存されているクライアントの情報に依存するサーバアプリケーションなどは例外で、この方法では容易に負荷分散できない場合があります。これには以下のようなアプリケーションがあります。

- ファイルをディスクに保存してユーザのログインをトラッキングする Web メールアプリケーション
- クライアントの買い物かごの中身をディスクまたはメモリに保存する買い物かご
- 通常より起動時間が長いプログラムやプロトコル (Java アプリケーション、SSL コネクションなど)

このような場合、トラフィックマネージャはクライアントとのセッションの間、以下のようにリクエストごとに同じバックエンドサーバにマッピングすることができます。

トラフィックマネージャは新しいコネクションを受け取ると、負荷分散ロジックを使用してそのコネクション用のノードを選択します。その後、選択したノードをセッション維持マップに記録します。

トラフィックマネージャは、同じセッションで別のコネクションを受け取ると、前に選択したノードを使用します。

このように、同じ「セッション」での全てのコネクションが同じバックエンドノードに固定されます。セッションの識別方法は、使用するセッション維持クラスによって決まり、複数の TrafficScript ルールを使用することでその決定を微調整することが可能です。

セッション維持クラスを使用すると、特定クライアントセッションの全てのリクエストを同じノードに送信することができます。これは、1つのアプリケーションセッションが多数のコネクションにわたるような複雑なアプリケーションで必要になる場合があります。例えば、Web ベースの買い物かごや多くの複雑な UDP ベースのプロトコルがこれに該当します。

セッション維持は、必要な場合のみ使用してください。セッション維持を使用すると、実質的に新しいセッションを除く全てのセッションで負荷分散プロセスが回避されます。同じクライアントからの全てのリクエストを同じバックエンドノードに送信する必要がある場合を除き、セッション維持を使用しない方がレスポンス時間は短くなります。

セッション維持の設定

セッション維持は、カタログを使用して設定します。カタログに追加したセッション維持クラスは、プールに割り当てたり、TrafficScript ルールで選択したりできます。

TrafficScript を使用しない場合、セッション維持はプール単位で有効化します。複数のプールを定義し、必要なプールでのみセッション維持を有効化することができます。同じバックエンドサーバが複数のプールに存在することに注意してください。

複数のプールで同じセッション維持クラスを参照し、セッション維持マッピングを共有することができます。例えば、いくつかの HTTP ノードが含まれるプール(ポート 80)と、HTTPS サービスを実行する同じノードが含まれるプール(ポート 443)は、ユーザとサービス間のセッション維持マッピングを共有できます。

TrafficScript を使用する場合、TrafficScript コマンドでセッション維持をオーバーライドすることができます。このコマンドが使用されていないければ、トラフィックマネージャは、リクエストを処理するプールに指定されたセッション維持クラスを使用します。

セッション維持の有効化

まず、**Catalogs** (カタログ)ボタンをクリックし、次に **Persistence** (セッション維持) タブをクリックすると、トラフィックマネージャが管理しているセッション維持クラスのリストが表示されます。次に、**Create new Session Persistence class** (新規セッション維持クラス作成)ラベルの付いたボックスに名前を入力し、**Create Class** (クラス作成)をクリックします。

ユーザインターフェースにセッション維持クラスの設定の概要が表示されます。

▼ Basic Settings

Each Session Persistence class controls two main issues: How to identify requests from the same session, and what action to take if the required node is unavailable.

Name: ASP_MANAGE_class

The type of session persistence to use.

type:

☒ IP-based persistence
Send all requests from the same source address to the same node.

☐ Universal session persistence
Use session persistence data supplied by a TrafficScript rule.

☐ Named Node session persistence
Use a node specified by a TrafficScript rule.

☐ Transparent session affinity
Insert cookies into the response to track sessions.

☐ Monitor application cookies ...
Monitor a specified application cookie to identify sessions.

☐ J2EE session persistence
Monitor Java's JSESSIONID cookie and URLs

☐ ASP and ASP.NET session persistence
Monitor ASP session cookies and ASP.NET session cookies and cookieless URLs.

☐ X-Zeus-Backend cookies
Inspect an application cookie named 'X-Zeus-Backend' which names the destination node.

☐ SSL Session ID persistence
Use the SSL Session ID to identify sessions (SSL pass-through only).

The action the pool should take if the session data is invalid or it cannot contact the node specified by the session.

failuremode:

☒ Choose a new node to use

☐ Redirect the user to a given URL ...

☐ Close the connection (using error_file on Virtual Servers > Edit > Connection Management)

Whether or not the session should be deleted when a session failure occurs. (Note, a failuremode of choosing a new node implicitly deletes the session.)

delete:

☒ Yes

☐ No

A description of the session persistence class.

note:

Fig 39. セッション維持の基本設定

セッション維持手法の選択

デフォルトでは、新しいセッション維持クラスは、**IP-based Persistence** (IP ベースセッション維持)が選択された状態で設定されます。この他、必要に応じて、以下のセッション維持手法を選択できます。

手法	対応プロトコル	説明
IP-based persistence (IP ベースセッション維持)	全て	同じ送信元アドレスの全てのリクエストを同じノードに送信します。
Universal session persistence (ユニバーサルセッション維持)	全て	このクラスでは、TrafficScript ルールにユニークなセッション維持キーを設定することにより、リクエスト内の任意の情報に基づいてセッションを維持することができます。

Brocade Virtual Traffic Manager: User's Guide

213

手法	対応プロトコル	説明
Named Node persistence (ネームドノードセッション維持)	全て	この手法では、TrafficScript ルールにより接続を特定ノードに送信することができます。
Transparent session affinity (トランスペアレントセッションアフィニティ)	HTTP、HTTPS	セッションをトラッキングするために、レスポンスにクッキーを挿入します。
Monitor application cookies (アプリケーションクッキーのモニタリング)	HTTP、HTTPS	セッションを識別するために、指定されたアプリケーションクッキーをモニタリングします。
J2EE	HTTP HTTPS	Java の JSESSIONID クッキーとその URL 書き換えフォールバックをモニタリングします。
ASP and ASP.net session persistence (ASP および ASP.net セッション維持)	HTTP、HTTPS	この手法では、セッション維持を使用して両方のタイプの ASP セッションを処理することができます。
X-Zeus-Backend cookies (X-Zeus-Backend クッキー)	HTTP、HTTPS	宛先ノードを設定する X-Zeus-Backend というアプリケーションクッキーをチェックします。

手法	対応プロトコル	説明
SSL Session ID persistence (SSL セッション ID による維持)	SSL プロトコル	SSL セッション ID でセッションを識別します。

IP ベースセッション維持

IP-based persistence (IP ベースセッション維持)を選択すると、トラフィックマネージャは、このプールに送信された各リクエストの送信元 IP アドレスをトラッキングします。トラフィックマネージャが同じアドレスからのトラフィックを受信したことがある場合、以前使用したものと同一バックエンドサーバにマッピングします。

セッション中に IP アドレスを変更する特殊なモバイルクライアントや、IP を共有するクライアントがあるため(プロキシサーバを経由する場合など)、環境によってはこの手法が適さない場合もあります。

ただし、IP ベースセッション維持は完全に透過的でプロトコルに依存しないため、複雑なプロトコルやアプリケーションでのセッション維持コネクションの確保に役立ちます。

IP セッションマップは、クラスタ内の全てのトラフィックマネージャマシンで共有されます。そのため、リクエストが別のトラフィックマネージャで受信された場合も適切なノードに送信されます。また、特定のトラフィックマネージャで障害が発生した場合も、そのトラフィックマネージャが管理していた IP セッションマップは他のトラフィックマネージャにより認識されます。

ユニバーサルセッション維持

ユニバーサルセッション維持では、TrafficScript ルールで定義されたセッション維持キーを使用して、同じセッション維持キーが設定された全てのコネクションを同じバックエンドノードに送信しようとします。

セッション維持キーは、ルールで TrafficScript 関数の `connection.setPersistenceKey()` によって指定されます。これにより、リクエスト内の任意の情報に基づくセッション維持が可能になります。例については、CHAPTER 12 の「ユニバーサル PHP セッション維持」セクションを参照してください。

ユニバーサルセッション維持マップは、クラスタ内の全てのトラフィックマネージャマシンで共有されます。そのため、リクエストが別のトラフィックマネージャで受信された場合も適切なノードに送信されます。また、特定のトラフィックマネー

ジャで障害が発生した場合も、そのトラフィックマネージャが管理していたユニバーサルセッションマップは他のトラフィックマネージャにより認識されます。

重要: ユニバーサルセッション維持を SIP over UDP で使用する場合、セッション維持は、クライアントから送信された最初のリクエストにのみ適用されることに注意してください。同じクライアントが送信する以降の全てのリクエストは、別のセッション維持キーが割り当てられていても、最初のリクエストと同じノードに送信されます。そのため、ユニバーサルセッション維持は、同じクライアントからの全てのリクエストを同じサーバに送信する場合のみ SIP で使用してください。

ネームドノードセッション維持

ネームドノードセッション維持を使用すると、コネクションをプール内の特定ノードに送信できます。この手法では、リクエストのルーティング方法を非常にきめ細かくコントロールすることができます。

TrafficScript ルールで `connection.setPersistenceNode()` 関数を使用することにより、セッション維持の決定時にどのノードを使用するかを指定できます。例えば、ルールでは、リモートデスクトップサービスへの新しいコネクションを受け取ると、外部データベースをクエリして、ユーザのデスクトップをホストするノードを決定します。次に、`connection.setPersistenceNode()` 関数を使用して、コネクションをそのノードに送信することができます。

トランスペアレントセッションアフィニティ

トランスペアレントセッションアフィニティでは、セッションをトラッキングするためにクッキーを HTTP レスポンスに挿入します。この手法は、ノードがレスポンスにクッキーを設定する必要がないため、一般に HTTP トラフィックや SSL 復号化 HTTPS トラフィックに最適です。

トランスペアレントセッションマップは、クライアント側の **X-Mapping-xxxxxx** というクッキーに保存されます。ここで、「xxxxxx」はセッション維持クラスを識別する内部文字列で、このクッキーの値は適切なノードを識別するための内部文字列になります。クラスタ内の全てのトラフィックマネージャマシンは、このセッションクッキーの値をチェックし、セッションを同じサーバノードに送信します。

アプリケーションクッキーのモニタリング

アプリケーションクッキーによるセッション維持では、ノードから送信された HTTP レスポンス内の特定のクッキーをモニタリングします。例えば、PHP アプリケーションでは、クラスタ化された PHP アプリケーションがセッションの状態をトラッキングしてインデックス化できるよう「PHPSESSID」というセッションクッキーが生成されます。このセッション維持手法では、特定のアプリケーションクッキーが保持されているリクエストを同じサーバノードに送信します。モニタリングするアプリケーションクッキーの名前を指定する必要があります。

アプリケーションクッキーセッションマップは、クライアント側の **K-CookieName-xxxxxx** クッキーに保存されます。ここで、CookieName はモニタリング対象のクッ

キー、「xxxxxx」はセッション維持クラスを識別する内部文字列で、このクッキーの値は適切なノードを識別するための内部文字列になります。クラスタ内の全てのトラフィックマネージャマシンは、このセッションクッキーの値をチェックし、セッションを同じサーバノードに送信します。

バックエンドサーバがアプリケーションクッキーの値を変更しても、セッションは有効で、クライアントは引き続き同じサーバノードに送信されます。

J2EE JSESSIONID クッキー/URL

J2EE JSESSIONID クッキー/URL セッション維持では、JSESSIONID クッキー(アプリケーションクッキーセッション維持と同様)と jsessionid パスパラメータの両方をモニタリングします。これらは、Java Extension 仕様(v2.4)で定義されており、BEA WebLogic、IBM WebSphere Application Server、JBoss/Tomcat などの Java Extension コンテナで使用されています。

J2EE セッションマップは、クラスタ内の全てのトラフィックマネージャで共有されるため、リクエストが別のトラフィックマネージャで受信された場合も同じセッションが適用されます。

ASP.net セッション維持

ASP (Active Server Pages)は、動的に生成される Web ページを処理するために Microsoft が開発したサーバ側スクリプティングプロトコルです。サーバクラスタで ASP セッション管理を使用するには、セッションの存続期間にわたって特定ユーザから受信した全てのリクエストを *同じ* Web サーバで処理する必要があります。

ASP セッションでは、ブラウザでクッキーがサポートされているかどうか、ユーザが自分の意思でクッキーを無効化しているかなど様々な要因に応じて、クッキーを使用することも使用しないこともできます。

- **クッキーベースのセッション:** トラフィックマネージャの ASP セッション維持クラスは、クッキーを検出し、それに基づいてクライアントのセッションを識別します。
- **クッキーレスセッション:** トラフィックマネージャの ASP セッション維持クラスは、ASP アプリケーションが生成した URL に埋め込まれた ASP 識別子を検出して使用します。

ASP セッションマップは、クラスタ内の全てのトラフィックマネージャで共有されるため、リクエストが別のトラフィックマネージャで受信された場合も同じセッションが適用されます。

単一のホスト名で複数の ASP アプリケーションをホストしている場合、各アプリケーションによって独自のクッキーが生成されることに注意してください。そのため、アプリケーションごとに個別にクッキーとノード間のマッピングを管理するためには、各アプリケーションに別々の ASP.Net セッション維持クラスが必要になります。例えば、リクエスト URL をチェックしてどのアプリケーションにアクセスするかを

決定し、TrafficScript 関数の `connection.setPersistence()` を使用してセッション維持クラスを選択することができます。

X-Zeus-Backend クッキー

X-Zeus-Backend クッキーセッション維持では、各アプリケーションリクエスト内で **X-Zeus-Backend** というクッキーを探します。このクッキーが存在し、現在のプールに含まれるノードの名前が保持されている場合、リクエストはそのノードに送信されます。このクッキーは、バックエンドサーバまたは TrafficScript ルールによって挿入できます。

注: このセッション維持手法は廃止される予定です。Zeus Load Balancer セッション維持クッキーとの後方互換性を維持する目的でのみ提供されているものです。

SSL セッション ID セッション維持(SSL パススルーのみ)

SSL セッション ID セッション維持手法では、同じ SSL セッション ID の全ての SSL トラフィックを同じサーバノードに送信します。SSL パススルートラフィックにのみ使用でき、SSL 復号化トラフィックには使用できません。

SSL セッション ID セッション維持手法を使用すると、ノードが実行する SSL ハンドシェイク処理の回数が減少します。SSL ハンドシェイクには、多くの CPU 時間、ネットワーク帯域、および待ち時間が伴います。

SSL セッション ID セッションマップは、クラスタ内の全てのトラフィックマネージャマシンで共有されます。そのため、リクエストが別のトラフィックマネージャで受信された場合も適切なノードに送信されます。また、特定のトラフィックマネージャで障害が発生した場合も、そのトラフィックマネージャが管理していた SSL セッション ID のセッションマップは他のトラフィックマネージャにより認識されます。

多くの SSL クライアントは定期的に SSL セッション ID についてのネゴシエーションを行うため、SSL セッション ID セッション維持は、アプリケーションレベルのセッション維持には適していません。アプリケーションレベルのセッション維持を行うには、IP ベースセッション維持を使用するか、トラフィックを復号化してユニバーサルセッション維持または適切であれば HTTP 手法を使用する必要があります。

ノードへのセッション維持マップの名前解決

トラフィックマネージャは、新しいコネクションを受け取ったときにセッション維持情報がない場合、負荷分散ロジックを使用してそのコネクションに使用するノードを選択します。選択されたノード(IP アドレスとポート)は、内部クッキー(IP、ユニバーサル、または SSL セッション維持の場合)または外部クッキー(トランスペアレント、アプリケーションクッキー、または X-Zeus-Backend クッキーの場合)のいずれかのセッション維持マッピングに記録されます。

トラフィックマネージャは、同じセッションで別のコネクションを受け取った場合に、そのコネクションをプール内のノードに転送できる状態にあれば、使用中のセッション維持クラスをチェックし、特定のノードを使用するかどうかを決定します。

コネクション用のプールに同じ IP アドレスとポートのノードが存在する場合、トラフィックマネージャはコネクションをそのノードに送信します。

正確に一致するものがない場合、トラフィックマネージャはプール内で同じ IP アドレスの(ただしポートが異なる)ノードを検索します。このようなノードが 1 つのみ存在する場合、トラフィックマネージャはコネクションをそのノードに送信します。

これにより、ノードタイプの異なる複数のプール間でセッション維持情報を共有することができます。

例えば、Web ベースのアプリケーションにおいて、買い物かごの中身の管理には HTTP インターフェースを使用し、支払いの管理にはセキュア HTTPS インターフェースを使用する場合があります。このとき、セッション維持の必要条件によっては、ユーザの HTTP トラフィックと HTTPS トラフィックの両方を同じ物理マシンに送信しなければならない場合があります。

両方のサービス(HTTP および HTTPS)で同じセッション維持クラスを参照することもできます。例えば、ユーザが最初に HTTP サービスに接続したときに、トラフィックマネージャは HTTP プールを使用し、そのプール内の特定のノード(IP アドレス、ポート 80)とのセッションが確立されます。

その後、ユーザが HTTPS サービスにアクセスしたときに、トラフィックマネージャが、HTTPS ノードが含まれる別のプールを使用することがあります。この場合、セッション維持クラスにより、ユーザのリクエストは HTTPS プール内の同じ物理マシン(IP アドレス、ポート 443)に送信されます。

ノード障害発生時のオプション

セッション維持マッピングで必要とされるノードが使用できなくなる場合があります。例えば、モニタによって「障害発生」のマークが付けられたり、トラフィックマネージャが接続できなくなった場合などです。

このような場合に備え、トラフィックマネージャには、そのノードを現在使用しているセッションに対する代替アクションが用意されています。**Session Persistence** (セッション維持)ページの下部に、アクションを選択するための以下のラジオボタンがあります。

Choose a New Node to Use (使用する新しいノードの選択)

プールはセッションマップを破棄し、現在の負荷分散アルゴリズムを使用して新しいノードを選択します。

Close the Connection (コネクション終了)

プールは即座にコネクションを終了します。HTTP トラフィックでは、プールのエラーファイルが送信されます。

セッションマップを破棄するか記憶しておくかを選択することができます。記憶しておくと、クライアントが再び接続し、セッションのノードが復旧したときに、リクエストはそのセッションのノードに送信されます。

HTTP リダイレクトの送信(HTTP のみ)

プールは、リクエストに対するレスポンスとして、設定されているロケーション(URL)に HTTP 302 リダイレクトを送信します。そのリダイレクトロケーションにあるリソースでは、メッセージを表示するか、ユーザに再度ログインして新しいセッションを確立するよう促すことができます。

セッションマップを破棄するか記憶しておくかを選択することができます。記憶しておくと、クライアントが再び接続し、セッションのノードが復旧したときに、リクエストはそのセッションのノードに送信されます。

コネクションのドレイン

例えばソフトウェアのアップグレード、ハードウェアのメンテナンス、撤去や再利用のために、バックエンドサーバをサービスから除外することが必要になる場合があります。

プールのコネクションのドレイン機能は、この作業を容易に行えるよう設計されています。ノードに「ドレイン」のマークを付けると、トラフィックマネージャはそのノードに新しいコネクションを送信しなくなります。ただし、そのノードに対して既に確立されているセッションでのコネクションは引き続きそのノードに送信されます。

これにより、継続中のコネクションや長時間確立されているセッションを中断することなく、ノードをプールから安全に除外することができます。管理サーバの **Activity** (アクティビティ)セクションに表示されるレポートで、ドレイン中のノードがアイドル状態にある時間を確認し、全てのセッションが完了したかどうかを判断することができます。

例えば、e コマースサービスを運営し、セッション維持によって個々のセッションを特定のバックエンドノードに紐付けているとします。ここで、ノードの1つにドレインのマークを付けると、以下ようになります。

そのノードとの間で新しいセッションは確立されません。

確立済みの既存のセッションでは、引き続きそのノードを使用できます。

60 分後、管理サーバの **Activity** (アクティビティ)セクションの **Draining Nodes** (ノードのドレイン)ページを調べます。ノードは 35 分間アイドル状態のままです。こ

のことから、確立されている全てのセッションが完了していると判断し、ノードを安全に除外することができます。

実際には、ノードがドレインを完了したかどうかを判断するための時間は、アプリケーションとユーザの動作によって大きく異なってきます。システムの設定時に、管理者は、目的とする用途を考慮してこれらの値を決定する必要があります。

プールでの接続のドレインの設定方法については、CHAPTER 5 の「ノードのドレインと無効化」セクションで説明します。

セッション維持キャッシュのサイジング

セッション維持手法には、クライアント側クッキーを使用してセッション維持データを保存するものと、クラスタ内で自動的に共有されるトラフィックマネージャのキャッシュを使用してセッション維持マッピングを保存するものがあります。

キャッシュのサイズは固定されます。キャッシュがいっぱいになると、新しいエントリが追加されたときに、最も古い(使用されていない期間が最も長い)エントリが破棄されます。キャッシュサイズは、**System** (システム) > **Global Settings** (グローバル設定) ページの **Cache Settings** (キャッシュ設定) セクションで設定します。

セクション	設定
IP session persistence (IP セッション維持)	ip_cache_size
Universal session persistence (ユニバーサルセッション維持)	universal_cache_size
SSL session-id session persistence (SSL セッション ID セッション維持)	ssl_cache_size
J2EE session persistence (J2EE セッション維持)	j2ee_cache_size
ASP session persistence (ASP セッション維持)	asp_cache_size

アクティビティモニタを使用して、キャッシュの動作をモニタリングすることができます。キャッシュのサイジングに役立つ主な値は以下のとおりです。

設定	説明
Entries (エントリ)	キャッシュ内にあるエントリの数
EntriesMax (エントリ最大値)	設定されているキャッシュの最大サイズ
Oldest (最も古いエントリ)	使用されていない期間が最も長いエントリの最終使用時点からの経過時間

トラフィックマネージャが多数のセッションを処理すると、通常、キャッシュはいっぱいになります。**Oldest** (最も古いエントリ)値は、現在のセッションの有効期限、すなわち使用されていないエントリをどれだけの期間保持してから破棄するかを示します。

キャッシュのサイジングを適切に行うには、新しいエントリがキャッシュに追加される頻度(1秒あたりのエントリ数)と、エントリが最後に使用されてから破棄されるまでの時間(秒単位) (セッションの有効期限)を考慮する必要があります。これらの2つの値を乗じて必要なキャッシュサイズを見積ります。また、**Oldest** (最も古いエントリ)値をモニタリングして、キャッシュがいっぱいになってエントリが破棄されるのが早過ぎないかを確認します。

セッション維持レコードをきめ細かくコントロールする必要がある場合、最も効果的な方法は、特定の有効期限が設定されたクライアント側クッキーを使用し、セッション維持をクッキーの存在に紐付けることです。

セッション維持とマルチホストトラフィック IP アドレスの併用

「consider source port (ソースポート考慮)」設定を有効にしてマルチホストトラフィック IP アドレスを使用する場合、特定の送信元 IP アドレス(すなわち特定のクライアント)からのリクエストは、クラスタ内の全てのトラフィックマネージャで処理されます(CHAPTER 6 の「トラフィック IP グループの作成」セクションを参照)。

この状況では、状態の共有(トラフィックマネージャ間でセッションマップを共有)に依存するセッション維持手法が確実に機能しない場合があります。セッション情報がクラスタ全体に伝搬されるまでに短い待ち時間が生じ、その間にクライアントが複数のトラフィックマネージャにアクセスすると、クライアントのセッション維持マッピングが壊れる可能性があるからです。

状態の共有に依存するセッション維持アルゴリズムには以下のものがあります。

- IP ベースセッション維持
- ユニバーサルセッション維持
- SSL セッション ID セッション維持

- J2EE および ASP.NET セッション維持

上記のセッション維持手法を使用するサービスで使用するマルチホストトラフィック IP では、「consider source port (ソースポート考慮)」設定を使用しないでください。

UDP プロトコルでのセッション維持

UDP プロトコルはコネクション指向ではありませんが、同じセッションの UDP データグラムを同じバックエンドサーバにルーティングするのが望ましい場合があります。

UDP セッションをどのように扱うかをトラフィックマネージャに通知するには、**Virtual Server** (仮想サーバ) > **Connection Management** (コネクション管理) 設定にある 2 つの設定 `udp_timeout` および `udp_response_datagrams_expected` を使用します。

UDP セッションは、設定 `udp_response_datagrams_expected` に等しい数のレスポンスデータグラムが検出されるまで存続します。例えば、サーバが 1 つのデータグラムをクライアントに送信したらセッションを完了する場合は、`udp_response_datagrams_expected` を「1」に設定します。

セッション長時間存続させる場合は、`udp_response_datagrams_expected` を「-1」に設定します。UDP セッションは、`udp_timeout` 秒間にわたって UDP トラフィックが確認されない場合にタイムアウトになります。

長時間にわたってセッションをトラッキングする場合、UDP で様々なタイプのセッション維持(一般には IP ベースまたはユニバーサル)を使用できます。

例

ユニバーサル PHP セッション維持

TrafficScript ルールでは、受信した HTTP リクエストをチェックし、リクエストに基づいてバックエンドノードを選択することができます。

CHAPTER 8 の「コンテンツタイプによるルーティング」セクションには、ファイル拡張子に基づいてプールを選択する TrafficScript ルールの例が示されています。このルールは、選択したセッション維持クラスにセッション維持データを渡すよう変更することができます。

PHP というセッション維持クラスを作成し、ユニバーサルセッション維持 (CHAPTER 12 の「セッション維持の設定」セクションを参照)を使用するよう設定します。次に、カタログで以下のような TrafficScript ルールを作成します。

```
$path = http.getPath();
if( string.endsWith( $path, ".php" )) {

    # Persist on the PHPSESSID cookie, IP address and
    # user agent
    $phpCookie = http.cookie( "PHPSESSID" ) .
                  connection.getRemoteIP() .
                  http.getHeader( "User-Agent" );

    # Set the persistence key to our unique value
    connection.setPersistenceKey( $phpCookie );

    # Select the PHP Session Persistence Class
    connection.setPersistence( "PHP" );
}
```

このルールを、トラフィックを処理する仮想サーバに適用します。pool.use() 関数に渡す文字列(\$phpCookie)は、プールが使用するセッション維持データです。

上記のルールでは、その他の拡張子(.html や.jpg など)のファイルに対するリクエストは無視されます。これらのリクエストは、仮想サーバが使用する他のルール、またはデフォルトプールで設定されているデフォルトのセッション維持クラスに渡されます。

また、必要なプールは1つのみで、セッション維持が2つのセッション維持クラスによって全て管理されることに注意してください。

CHAPTER 13 SSL 暗号化

本章では、トラフィックマネージャでの Secure Sockets Layer (SSL)暗号化の使用方法について説明します。SSL の概要、トラフィックマネージャでの認証および暗号化の管理方法、および証明書の保存について扱います。

SSL プロトコルのバージョン 3 は、トランスポート層セキュリティ(TLS)プロトコルのバージョン 1、1.1、および 1.2 で置き換えられました。特定の SSL バージョンが指定されている場合を除き、トラフィックマネージャでは、SSL と TLS の両方のプロトコルを総称して、SSL という用語を一貫して使用します。

SSL の概要

SSL は、インターネット上でトラフィックをセキュアに送信するためのプロトコルです。トラフィックは、サーバマシンとクライアントマシン間で合意された鍵を使用して暗号化されます。

SSL には、以下のような効果があります。

- サーバ認証
- クライアント認証
- 暗号化データ転送

SSL は、ほぼ全ての TCP/IP プロトコルで使用できますが、一般には、HTTP (Web) トラフィックのセキュリティを強化して HTTPS プロトコルを形成するために使用されています。

サーバ認証

サーバは、SSL 通信において SSL 証明書を使用してそれ自体の身元を明らかにします。この証明書には、組織の名前と場所およびその DNS 名が含まれ、これによってクライアントはアクセス先が正しいサイトであることを確認できます。

SSL 証明書は、その証明書を所有している組織が自己署名できます。ただし、第三者による検証がなければ、自動的にクライアントに信頼されません。信頼されるためには、Verisign や Thawte など広く認められた独立系認証局 (CA) の署名が必要です。組織が証明書署名リクエスト(CSR)を CA に送信すると、CA において証明書の詳細が綿密にチェックされます。認証局によるこのサービスには料金がかかる場合があることに注意してください。

CA は、自ら発行する証明書内の鍵が正しく生成され、十分な暗号強度を持つことに責任を負う場合があります。

クライアントは、オンライン証明書ステータスプロトコル(OCSP)を使用して、サーバ証明書が取り消されているか調べようとする場合があります。TLS

「status_request」拡張を使用することにより、このチェックを TLS ハンドシェイクに含めることができ(OCSP Stapling と呼ばれます)、クライアントから OCSP レスポンダへ別途接続する必要がなくなります。OCSP Stapling を有効化するには、下の「OCSP の設定」を参照してください。

OCSP とは

OCSP は、使用時点における SSL 証明書の有効性を取得するためのインターネットプロトコルです。CRL には、リストを最新の状態に保つために更新版を頻繁にダウンロードしなければならないなどの制限がありますが、OCSP はこのような CRL 固有の欠点を解決するために *証明書失効リスト (CRL)* (227 ページの「クライアント認証における CRL」を参照)に代わる手段として作成されました。

ユーザは、セキュアサービスにアクセスしようとするとき、OCSP サーバ(レスポンダと言う)に証明書のステータス情報をリクエストする HTTP リクエストを送信します。このリクエストは、ASN.1 メッセージの形式にパッケージ化され、オプションで証明書によって署名され、レスポンダに送信されます。このリクエストに対し、レスポンダは「good (良好)」、「revoked (取り消し済み)」、または「unknown (不明)」のレスポンスを返します。

OCSP の詳細については、<http://tools.ietf.org/html/rfc6960> を参照してください。

クライアント認証

企業のイントラネットやエクストラネットなど、承認された特定のユーザにのみサービスへのアクセスを許可したい場合もあります。これを行うために、クライアントに対して、信頼された認証局によって署名された SSL 証明書を提示するようリクエストすることができます。

トラフィックマネージャは、CRL および OCSP を使用してこれらの証明書の有効性を確認します。

クライアント認証における CRL

トラフィックマネージャでは、信頼された認証局はカタログに保持されます。各認証局は、*認証失効リスト(CRL)*を配信することができ、これもこのカタログに保持されます。通常、証明書には 12 か月など一定の有効期間がありますが、この期間が期限切れになる前に取り消される場合もあります。この場合、認証局は、その証明書が以降信頼されないよう証明書失効リストに追加します。

クライアント認証における OCSP

OCSP を使用すると、クライアント証明書の現在のステータスを確認することができます。証明書が良好として報告されない限り、SSL コネクションは切断されます。

セキュアサービスに OCSP を設定するには、CHAPTER 13 の「OCSP の設定」セクションを参照してください。

暗号化データ転送

サーバとクライアントは、相互の身元を確認すると、データ転送に使用する暗号化鍵について合意します。これには、効率上の理由から、識別鍵とは異なる鍵が使用されます。データは、第三者が読み取れないよう転送前に暗号化されます。また、SSL には、データストリームの損傷を検出するための信頼性機能があります。これらの機能により、クライアントとサーバは、相互の通信の秘密が守られ、データに損傷がないことに確信を持つことができます。

トラフィックマネージャの SSL 機能

復号化と暗号化

トラフィックマネージャでは、仮想サーバ内で SSL トラフィックを復号化することができます。この機能は、以下の 2 つの理由から有益です。

- 復号化後、トラフィックマネージャのトラフィック分析機能をリクエスト全体に使用することができます。サービスプロテクション手法によって悪意のあるコンテンツ、ウイルス、Web ワームをフィルタしたり、ルールによってリクエストのヘッダおよびボディをチェックし、その情報に基づいてルーティングを決定したりできます。復号化しない場合、パケットから得られる情報はほんのわずかです。
- 復号化には処理能力が必要です。トラフィックマネージャがノードに渡す前にリクエストを復号化すれば、バックエンドサーバの負荷が軽減されるため、効率の向上を図ることができます。

TrafficScript ルールを使用するために仮想サーバで SSL トラフィックを復号化する場合は、一般に、そのトラフィックをノードに送信する前に暗号化します。この暗号化はトラフィックマネージャの設定で作成するプールによって処理され、システムにおいて完全なエンドツーエンドのセキュリティが実現されます。

SSL 証明書カタログ

トラフィックマネージャでは、SSL サービスで利用できる SSL サーバ証明書、クライアント証明書、認証局、および証明書失効リストが一元管理される中央ストアとして、SSL 証明書カタログが使用されます。

仮想サーバは、デフォルトサービスへのトラフィック、およびオプションとして仮想サーバの設定で定義される「SSL サイト」へのトラフィックを管理します。各 SSL サイトでは、カタログからの最大 2 つの SSL 証明書と、IP アドレスまたはホス

ト名のいずれかが関連付けられます。詳細については、239 ページの「1 台の仮想サーバを使用した複数のサイトへの対応」を参照してください。

SSL 復号化ウィザード

SSL Decrypt a service (サービスの SSL 復号)ウィザードを使用すると、画面の簡単な指示に従うことで、SSL 復号化を適切に設定することができます。このウィザードでは、以下の操作を実行します。

- SSL 仮想サーバで SSL 復号化を有効化する
- カタログの SSL 証明書を仮想サーバに割り当てる
- 仮想サーバが使用するデフォルトプールで SSL 暗号化を有効化する
- 全てのプロトコルタイプを基礎となるプロトコルタイプ(非 SSL)に変更する

注: SSL Decrypt a service (サービスの SSL 復号)ウィザードを使用するには、1 台以上の SSL 仮想サーバ(タイプは問わない)が必要です。必要な SSL 証明書がない場合、トラフィックマネージャでウィザードを使用して証明書を作成することができます。

SSL Decrypt a service (サービスの SSL 復号)ウィザードを使用する

1. ツールバーの **Help** (ヘルプ)リンクの横にある「**Wizards** (ウィザード)」ドロップダウンリストをクリックし、**SSL Decrypt a service** (サービスの SSL 復号化)を選択します。新しいウィンドウが開き、ウィザードにより、受信時に復号化し、プールへの引き渡し前に再度トラフィックを暗号化するようサービスが設定されることを示す説明が表示されます。**Next** (次へ)をクリックします。
2. 復号化するサービスを選択します。リストには、SSL プロトコルを使用している仮想サーバのみが表示されることに注意してください。**Next** (次へ)をクリックします。
3. 受信したリクエストの復号化に使用する証明書を選択(または **Create New** (新規作成)をクリックして証明書を追加)します。**Next** (次へ)をクリックします。
4. プロトコルタイプを選択します。これは、基礎となる復号化後のプロトコルです。例えば、HTTPS リクエストは、内部的に HTTP に復号化されます。この場合は、基礎となるプロトコルとして単純に HTTP を選択します。これが、SSL でラップされていない状態で受け入れるプロトコルタイプだからです。
5. **Finish** (終了)をクリックしてウィザードを終了します。

SSL 証明書の設定

SSL 証明書を設定するには、**Catalogs** (カタログ) > **SSL** をクリックし、要件に応じて **SSL Server Certificates catalog** (SSL サーバ証明書カタログ) または **SSL Client Certificates catalog** (SSL クライアント証明書カタログ) をクリックします。

- **サーバ証明書**は、トラフィックマネージャがホストする、SSL で暗号化されたサービスを識別するために使用されます。
- **クライアント証明書**は、トラフィックマネージャがそれ自体を SSL ノードに対して認証する必要がある場合に使用されます。

注: 以降のセクションでは、クライアントおよびサーバの両方の SSL 証明書に適用できる機能について取り上げます。

自己署名 SSL 証明書の新規作成

自己署名 SSL 証明書を新規作成するには、以下の手順に従います。

1. 管理 UI のツールバーで **Catalogs** (カタログ) をクリックし、**SSL** タブをクリックします。そこから、必要に応じて **SSL Server Certificates Catalog** (SSL サーバ証明書カタログ) (サーバ証明書の場合) または **SSL Client Certificates catalog** (SSL クライアント証明書カタログ) の **Edit** (編集) ボタンをクリックします。
2. トラフィックマネージャに設定済みの既存の証明書のリストが表示され、証明書の作成またはインポートが可能になります。テストや内部利用が目的の場合には、SSL 証明書を **自己署名** することができます。これは、証明書が信頼できる第三者によって署名されていないことを表し、サーバを認証する手段として信頼して利用すべきでないということを意味します。

Enter a short name to identify your certificate. If you leave this blank, the 'Common Name' field will be used.

Name:

The public DNS address of your server, such as 'secure.yourcompany.com':

Common Name (CN):

The name of your organisation, such as 'Your Company':

Organisation (O):

The unit within your organisation, such as 'Sales':

Organisational Unit (OU): (optional)

Your location (town or city), such as 'Anytown':

Location (L):

Your state or province, such as 'Somestate':

State (S): (required for US only)

Your two-letter country code, such as 'US', 'GB' or 'FR':

Country (C):

How long should this certificate be valid for:

Expires in:

Private key size (2048 bits recommended):

Key size:

Fig 40. 自己署名証明書の主な設定項目

3. **Create Self-Signed Certificate (自己署名証明書作成)/Certificate Signing Request** (証明書署名リクエスト)をクリックし、フォームに必要事項を入力します。
4. 証明書の **name** (名前)と **Common Name** (CN、一般名称)を指定します。後者にはこの証明書を使用するサーバの DNS 名、例えば `secure.yourcompany.com` などを指定する必要があります。ほとんどのクライアントは、証明書の CN とサーバの DNS アドレスが一致しない場合に警告メッセージを表示します。
5. **organization** (組織)、**organizational unit** (組織単位)、**address** (住所)に、それぞれ入力します。これらは証明書に表示される情報です。
6. 国コードを入力する際には、**2 文字の ISO 国コード**⁷を使用します。例えば、英国は「GB」、ドイツは「DE」です。
7. 証明書の **expiry date** (有効期限) (デフォルトは 1 年)を設定します。
8. 証明書に対して秘密鍵の **key size** (鍵サイズ)を選択します。P256、P384、または P521 を選択して ECDSA 鍵を含む証明書を作成するか、1024、2048、3072 または 4096 を選択して、RSA 鍵を含む証明書を作成します。

⁷ ISO-3166 標準で定義されています。

9. **Create Certificate** (証明書の作成)をクリックします。エラーがなければ、トラフィックマネージャによって SSL 証明書が作成され、SSL 証明書カタログに追加されます。

Catalogs (カタログ)ページで **SSL Certificates Catalog** (SSL 証明書カタログ)の横にある拡張タブをクリックすると、証明書が追加されたことを確認できます。各証明書とその有効期限がテーブルに表示されます。

証明書データの管理

自己署名証明書に保存されている任意の値を変更することができます。これを行うには、適切なクライアントまたはサーバ SSL 証明書カタログ内の既存の SSL 証明書ををクリックします。

Edit Certificate (証明書編集)ボックスで、証明書の既存の値を変更し、**Update Certificate** (証明書更新)をクリックします。

注: 認証局によって署名された証明書を編集することはできません。編集すると、署名が無効になります。

既存の SSL 証明書をコピーして、新しい証明書名を付けることも可能です。これを行うには、SSL 証明書カタログ内の既存の証明書ををクリックします。最初に展開タブをクリックして、クライアントまたはサーバ SSL 証明書カタログを展開することが必要な場合もあります。

Copy Certificate (証明書コピー)ボックスにコピーの新しい名前を入力し、**Copy Certificate** (証明書コピー)をクリックします。新しい SSL 証明書が関連するカタログに直ちに追加されます。

証明書署名リクエストの作成

証明書署名リクエスト(CSR)は、身元を明らかにするデジタル証明書を取得するために、個人が認証局(CA)に対して行う正式なリクエストです。認証局とは、第三者が使用する証明書の発行を担う機関です。

自己署名証明書の作成後、以下の手順に従って、自己署名証明書に含まれる情報に基づいて証明書署名リクエストを作成することができます。

1. **Catalogs** (カタログ) > **SSL** > **[Server|Client] Certs** ([サーバ|クライアント] 証明書)画面に移動します。
2. 必要な証明書ををクリックして編集します。
3. **Certificate Signing** (証明書署名)セクションで、**Export CSR** (CSR のエクスポート)/**Sign certificate** (証明書署名)をクリックします。

ウィンドウに証明書リクエストのテキストが表示されます。

SSL Certificates Catalog

SSL Certificate: SSL Cert 1

This form helps you to sign your certificate.

Certificate Signing Request (CSR)

Your Certificate Authority will use this Certificate Request text to create and issue a trusted certificate, based on this certificate.

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqDCCARECAQAwDELMAkGA1UEBhMCR0Ix EjAQBgNVBAcTCUNhbWJyaWRnZTER
MA8GA1UEChMIUm12ZXJiZWQxZDAsBgNVBAstC0R1dmVsb3BtZW50MRwwGgYD VQQD
ExNzZWNIcmUucm12ZXJiZWQxY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDfmczksdumadlvx19PLHBpykMfrZOOSY4n+1zuYhvKJqLbmesFQv5+41v+Jw3z
zhTd/s/GgzieXl+3z9PMo9x15HQD3Ey7fEB7rum6iAFnccf7nV55hJWthpMEPMSo
tTCrBODoX5t/i1V90FuFoyPKGyYBcjtqREPBvneN0ukwIDAQABoAAwDQYJKoZI
hvcNAQEFBQADgYEAiuv66PRzBYv1NJ2qY1E20lHxbEyXaqBORg+YsB++UcXgMwS3
x/+Kr9i/eOSPLSO77sB7+Wvx2IIBrzzG2hoagCOOLUKCyLPWdlm03BpTKZCYTbyfK
Y72Pf5BFPVDo6eoFj fMNMMA02RuzA8POEA1uUh2V8BMDmJmVVy3mXBtVH6v4E=

Replace certificate

Once you have received a new certificate, paste it here to replace your current certificate.

Update Certificate

Note: This will completely replace your current certificate. You may wish to copy your current certificate before doing this.

Fig 41. 認証局(CA)に対する証明書リクエスト

このウィンドウの上部に表示されるテキストは、CA による署名に適した、保存されている証明書内の情報を符号化したものです。このテキストをコピーしてリクエストに貼り付けることができます。リクエストにその他のクレデンシャルや追加情報を設定するよう求められる場合があります。

認証局から、認証局がいずれかの公開証明書(または公開証明書チェーン内の証明書)でデジタル署名した代替証明書が返されます。返された証明書のテキストをウィンドウの下部に貼り付けて **Update certificate** (証明書更新)をクリックすると、このプロセスは完了します。

新規 SSL 証明書のインポート

既存の証明書をインポートするには、以下のファイルが必要です。

- 証明書ファイル
- 秘密鍵ファイル

これらのファイルは、アップロード用のテキストファイルにカットアンドペースト可能な標準の PEM 符号化フォーマットです。PEM 符号化証明書のフォーマットは以下のとおりです。

```
-----BEGIN CERTIFICATE-----
MIIBPLMAkGA1UEBhMCWkExFTATBgNVBAgTDfdlc3Rlcm4gQ2Fw
ZTESMwZSBUb3duMRQwEgYDVQQKEwtPcHBvcnR1bml0aTEYMBYG
A1UEC1cnZpY2VzMRowGAYDVQQDExF3d3cuZm9yd2FyZC5jby56
YTBaMBAQUAA0kAMEYCQQDT5oxxeBWu5WLHD/G4BJ+PobiC9d7S
6pDvAtXdm2j190D1kgDoSp5ZyGSgwJh2V7diuuPlHDAgEDoAAw
DQYJKDQQBf8ZH8Iu4H8ik2vZQngXh8v+iGnAXD1AvUjuDPCWzFu
pReiq7UR8Z0wiJBeaqiuvTDnTFMz6oCq6htdH7/tvKhh==
-----END CERTIFICATE-----
```

アップロードするには、関連する **Client/Server SSL Certificate Catalog** (クライアント/サーバ SSL 証明書カタログ) の **Import Certificate** (証明書インポート) をクリックします。インポート後の証明書の名前、*証明書ファイル* および *秘密鍵ファイル* の名前を指定します。次に、**Import Certificate** (証明書インポート) ボタンをクリックすると、このプロセスは完了します。トラフィックマネージャは、証明書を適切なカタログに追加し、自動的に鍵データを全てのトラフィックマネージャにセキュアに配布します。

重要: インポートプロセス中に作成した SSL 秘密鍵の一時コピーは必ず削除してください。秘密鍵は大切なものであるため、セキュリティが確保されていない場所に保存しないでください。

認証局によって署名された証明書が **ルート証明書** で直接署名されていない場合、署名された証明書と認証局の中間証明書の両方をクライアントに送信することが必要になることがあります。

中間証明書の使用方法

Web ブラウザおよびその他の SSL クライアントには、信頼する認証局の一連の **ルート証明書** があらかじめ設定されています。ユーザはこれらのルート証明書により、信頼するいずれかの認証局によって署名された **SSL サーバ証明書** を使用している SSL サービスに接続することができます。同様に、バックエンド SSL ノードは、トラフィックマネージャが提示するクライアント SSL 証明書を認証するようあらかじめ設定されている場合があります。

ただし、管理を容易にし、セキュリティを強化するために、多くの認証局は **証明書チェーン** を使用しています。これらの証明書チェーンから配信される SSL 証明書は、ルート証明書で直接署名されていません。代わりに、ルートによって署名された中間証明書によって署名されており、これによって SSL 証明書から、信頼されたルート証明書までの信頼チェーンが形成されています。場合によっては、SSL 証明書からルートまでの間で複数の中間証明書が使用されることもあります。

一般に、Web クライアントおよび SSL ノードには中間証明書が設定されていません。例えば、Web クライアントにはサーバ証明書が提示されますが、その証明書が信頼されたルート証明書によって(間接的であっても)署名されたものであることを検証す

る手段がありません。この場合、SSL サーバ(トラフィックマネージャなど)は、クライアントが SSL 証明書を検証できるよう SSL 証明書チェーンの全部または一部を提示する必要があります。証明書チェーンには、SSL 証明書および全ての中間証明書が含まれますが、ルート証明書は必ずしも含まれている必要はありません。

証明書チェーンが使用されているかどうかは、認証局から通知されます。関連するサーバまたはクライアント SSL 証明書カタログに全ての中間証明書をアップロードする必要があります。

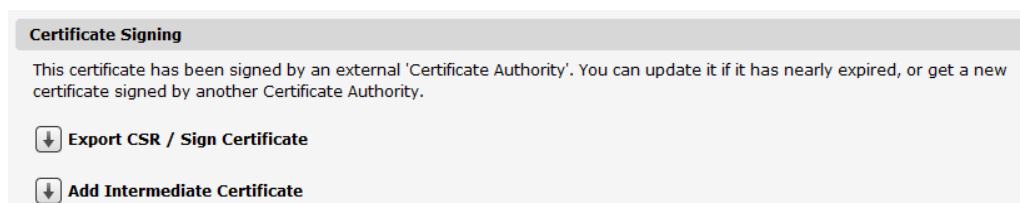


Fig 42. 中間証明書の追加

トラフィックマネージャは、アップロードした中間証明書が現在の SSL 証明書(またはチェーン内の 1 つ前の中間証明書)で署名されていることを検証します。

認証局の証明書と CRL ファイルの管理

トラフィックマネージャでは、SSL コネクションするリモートクライアントに対し、そのクライアントを認証するためにクライアント証明書を提示するようリクエストすることができます。この動作を有効化するには、信頼する認証局(CA)からの公開証明書と、オプションで認証局から配信される証明書失効リスト(CRL)をトラフィックマネージャに設定する必要があります。

1 つ以上の CA 証明書をインストールすると、`ssl_encrypt` と `ssl_strict_verify` を有効化したトラフィックマネージャ設定の各プールでは、コネクションが試行されたときに CA 証明書のオーナーによって発行された証明書の認証が要求されます。

CA 証明書と CRL を管理するには、**Catalogs (カタログ) > SSL > Certificate Authorities and Certificate Revocation Lists Catalog (認証局および証明書失効リストカタログ)**をクリックします。

トラフィックマネージャに新しい CA または CRL ファイルをインポートするには、**Import certificate or CRL (証明書または CRL のインポート)**をクリックし、以下のいずれかの方法を使用します。

- **Choose File (ファイルの選択)**をクリックして、ローカルワークステーションからトラフィックマネージャに証明書または CRL をアップロードします。
- トラフィックマネージャの **File URL (ファイルの URL)**ボックスに HTTP URL または HTTPS URL を入力して、直接ダウンロードします。

- **File Contents** (ファイルコンテンツ)ボックスにファイルのコンテンツを入力またはコピーします(PEM 符号化)。

Import File (ファイルのインポート)をクリックして、プロセスを完了します。トラフィックマネージャが CA または CRL ファイルをインポートし、新しい情報をクラスタ内の全てのトラフィックマネージャに伝搬します。

注: ノードに対して暗号化コネクションが実行されたとき、またはクライアント証明書を認証するために仮想サーバが設定されたときに、暗号化セキュリティを保証するために、CA が自ら発行する証明書の公開鍵有効性チェックを確実に実行するようにします。例えば、FIPS モードが有効化されている場合、NIST SP 800-89

「*Recommendation for Obtaining Assurances for Digital Signature Applications*」(セクション5.2)によって要求されるチェックを実行したことが確認されている CA のみを CA カタログに含める必要があります。

SSL 復号化

トラフィックマネージャは、クライアントとバックエンドプールとの間でリクエストをプロキシする際に、SSL カタログに保存されているサーバ証明書やクライアント証明書を使用して、SSL トラフィックを復号化し、再度動的に暗号化することができます。

例えば、トラフィックマネージャは、TrafficScript ルールを適用するためにトラフィックを復号化した後、データをノードに渡す前に必要に応じてデータを再度暗号化できます。これにより、信頼されていない可能性があるネットワークでも必要なセキュリティを損なうことなく、柔軟にリクエストを管理することができます。このように、トラフィックマネージャは、完全に透過的な SSL ゲートウェイとして機能します。

暗号化に対する攻撃者が利用できるコンピューティングリソースが増加するほど、SSL/TLS サイトを保護するために強度の高い鍵が必要になります。しかし、DSA 鍵や RSA 鍵を使用するために必要な処理時間は、鍵が長くなるほど急速に増大します。この問題に対処するために、トラフィックマネージャには、強度の高い鍵に対する効率の高い楕円曲線暗号化(ECC)オプションがあります。具体的には、トラフィックマネージャは楕円曲線 DSA(ECDSA)に基づく鍵をサポートしています。

SSL 復号化の設定

SSL 復号化を有効にするには、**Services** (サービス) > **Virtual Servers** (仮想サーバ)をクリックし、SSL をサポートしている仮想サーバの **Edit** (編集)ボタンをクリックします。

仮想サーバの設定オプションのリストで、**SSL Decryption** (SSL 復号)の **Edit** (編集)ボタンをクリックします。

この仮想サーバの SSL 復号化に関する設定オプションのリストが表示されます。

SSL Decryption

These settings control how SSL connections are decrypted.

Whether or not the virtual server should decrypt incoming SSL traffic.
ssl_decrypt: ☒ Yes ☐ No

Which SSL certificate(s) should this virtual server use?
 Additional certificates can be supplied to match different sites hosted by this virtual server. You can specify a different certificate for any hostname or IP address. The wildcard character '*' can be used to match multiple hostnames. If none of the addresses or hostnames match the default certificate will be used.
Note: Hostname mappings require support of the TLS 1.0 'Server Name Indication' extension, which is not supported by all browsers.

certificate: Default Certificates: RSA-Cert (www.brocade.com, Expires 24 Feb 2017) ECDSA-Cert (www.brocade.com, Expires 24 Feb 2017)

IP Address / Host Name	Certificate	Remove
example.brocade.com	RSA-Site-Cert (example.brocade.com, Expires 24 Feb 2017)	☐
	ECDSA-Site-Cert (example.brocade.com, Expires 24 Feb 2017)	☐

Add certificate mapping:
 IP Address / Host Name:
 Certificates:

Manage SSL Certificates

Whether or not the virtual server should add HTTP headers to each request to show the SSL connection parameters.
ssl_headers: ☐ Yes ☒ No

If the traffic manager is receiving traffic sent from another traffic manager, then enabling this option will allow it to decode extra information on the true origin of the SSL connection. This information is supplied by the first traffic manager.
ssl_trust_magic: ☐ Yes ☒ No

Whether or not to send an SSL/TLS "close alert" when the traffic manager is initiating an SSL socket disconnection.
ssl_send_close_alerts: ☒ Yes ☐ No

If OCSP URIs are present in certificates used by this virtual server, then enabling this option will allow the traffic manager to provide OCSP responses for these certificates as part of the handshake, if the client sends a TLS status_request extension in the ClientHello.
ssl_ocsp_stapling: ☐ Yes ☒ No

Fig 43. クライアントの SSL 復号化および証明書のマッピングに関する設定

SSL 復号化を有効にするには、**ssl_decrypt** を Yes (はい) に設定し、ページの下部で **Update** (更新) をクリックします。

デフォルトの SSL サイトで使用する証明書を選択するには、**certificate** (証明書) 設定を使用します。この仮想サーバによって処理される追加の SSL サイトに証明書のマッピングを追加するには、「**Add certificate mapping** (証明書のマッピングの追加)」セクションを使用します。

いずれの場合にも、表示される 1 組のドロップダウンリストを使用して、1 つまたは 2 つの証明書を選択します。1 つの証明書を指定する場合は、任意の該当する証明書の鍵タイプを使用できます。しかし、2 つの証明書を指定する場合は、各証明書に対して異なる鍵タイプを選択する必要があります。言い換えれば、RSA 鍵に基づいて 1 つの証明書を選択し、ECDSA 鍵に基づいて別の証明書を選択する必要があります。

異なる鍵タイプで複数の証明書を設定することにより、トラフィックマネージャは、古いクライアントと新しいクライアントに対して同様に最適な鍵タイプを判別できます。

以下の設定を行うこともできます。

- **ssl_headers:** トラフィックマネージャには、実行した復号化の詳細(HTTPS 復号化のみ)が記述された HTTP ヘッダを追加するためのオプションがあります。HTTP ヘッダには、クライアントの暗号、セッション ID、および有効なクライアント証明書が提示されたかどうか記述され、バックエンドアプリケーションは、これらのヘッダをチェックして使用された暗号化パラメータを特定することができます。

ヘッダ名	値の例
SSLClientCipher	SSL_RSA_WITH_AES_128_CBC_SHA256, version=TLSv1.2, bits=128
SSLClientCertStatus	NoClientCert または OK
SSLSessionID	8723367551317ABE5443278C6E... (64 文字の 16 進数文字列)

- **ssl_trust_magic:** トラフィックマネージャの 2 つの独立したクラスタを含む環境がある場合、あるクラスタから他のクラスタに転送される SSL トラフィックでは、通常は最初のクラスタが接続元と見なされます。コネクションの実際の接続元 IP アドレスを相互に渡すようにトラフィックマネージャに指示するには、この設定を使用します。

注: この設定を使用するには、トラフィックマネージャが適切に設定されている必要があります。詳細については、プールの SSL 設定の **ssl_enhance** を参照してください。

- **ssl_send_close_alerts:** トランケーション攻撃を防ぐために、SSL の仕様では、切断アラートの送信によってコネクションが終了しているという情報を、クライアントとサーバが共有することが要求されています。一部のクライアント、特に FTP クライアントは切断アラートに厳密なため、この設定はデフォルトで有効化されており、アラートが省略されると正常に機能しません。Microsoft Internet Explorer の特定のバージョンを含む一部の旧式の SSL 実装では、切断アラートが正しく動作しなくなる可能性があることに注意してください。使用しているサービスがこれらに該当する場合は、この設定を無効化することをお奨めします。TrafficScript ルールを使用することにより、この動作を動的に変更することもできます。

- **ssl_ocsp_stapling:** status_request 拡張がクライアントによって送信される場合、このオプションを有効化することにより、仮想サーバが OCSP 証明書ステータスを TLS ハンドシェイクに含められるようになります。またこれにより、各証明書の AIA 拡張フィールドの OCSP URI で指定されたレスポンドに対し、バックグラウンドで仮想サーバの各証明書に対する OCSP リクエストが実行されます。デフォルトでは、このオプションは無効化されています。

Update (更新)をクリックし、変更内容を仮想サーバに適用します。

1 台の仮想サーバを使用した複数のサイトへの対応

トラフィックマネージャは、以下の 2 通りの方法で複数のサービスを区別することができます。

- **宛先 IP アドレス:** 例えば、「secure.site1.com」と「secure.site2.com」を異なる IP アドレスに名前解決することができます。トラフィックマネージャは、両方の IP アドレスを監視し、コネクションを受け取った IP アドレスに基づいて使用する証明書を選択します。
- **TLS サーバ名の指定⁸:** TLS には、TLS ハンドシェイクの開始時にクライアントがアクセスしようとしているサービスの名前(プレーンテキスト形式)を指定できるオプション機能があります。トラフィックマネージャはこの名前をチェックし、使用する証明書を選択します。

重要: 各 SSL サービスを専用 IP アドレスで実行する必要があるため、TLS サーバ名の指定は拡張性が高く、管理が簡単です。しかし、Internet Explorer 7(Windows XP)などの一部の旧式の Web クライアントではこの機能がサポートされず、TLS サーバ名のサポートが必要なサービスにアクセスしようとする、正しくない証明書を受信することがあります。

仮想サーバが複数のサイトでトラフィックを管理している場合は、クライアントがアクセスしようとしている各サイトと一致するように、Subject Alternative Name (サブジェクトの別名)を含む証明書で仮想サーバを設定する必要があります。クライアントは、サービスの DNS 名と一致する Subject Alternative Name を見つけることができない場合、代わりに Common Name(CN、一般名)フィールドをチェックすることがあります。一致が見つからない場合、クライアントはエンドユーザに警告します。

複数の証明書の設定

単一の証明書には、仮想サーバによって提供される各 SSL サイトと一致するように、複数の Subject Alternative Name が含まれることがあります。この場合は、デフォルトとしてその証明書を使用するように仮想サーバを設定するだけで十分です。

⁸ http://en.wikipedia.org/wiki/Server_Name_Indication

または、1つのSSLサイトと一致する単一の **Subject Alternative Name** を含む複数の証明書で、仮想サーバを設定することもできます。次に、「**Add certificate mapping** (証明書のマッピングの追加)」セクションを使用して、特定の証明書のマッピングを使用するように仮想サーバを設定します。

最初のテキストボックスでIPアドレスまたはホスト名を指定し、ドロップダウンリストから証明書を選択して、新しい証明書のマッピングを作成します。ページの下部で **Update** (更新)をクリックして、このマッピングを保存します。

トラフィックマネージャはリクエストを受信すると、証明書の設定を調べ、送信する証明書を決定します。接続しているクライアントがSSLハンドシェイクで、マッピング内のホスト名と一致するサーバ名を提供するか、クライアントがマッピングで指定されたIPアドレスに接続する場合は、デフォルトの証明書ではなく、指定した証明書が提供されます。

トラフィックマネージャによって、最初に一致する証明書が選択されます。サーバ名の正確な一致、サーバ名のワイルドカードの一致、IPアドレスの正確な一致の順に確認されます。一致するものがない場合は、デフォルト証明書が使用されます。

仮想サーバによって管理される各SSLサービスには、関連付けられている少なくとも1つの証明書が必要です。ただし、トラフィックマネージャでは、異なる鍵タイプに基づいていることを条件として、第2の証明書をサービスに関連付けることもできます。言い換えれば、RSA/DSA 鍵に基づいて最初の証明書を選択した場合、第2の証明書は ECDSA 鍵に基づいて選択する必要があります。このシステムの利点は、より互換性の高いオプションがクライアントに提供されることです。

暗号および TLS バージョンの設定

各SSL復号仮想サーバに対し、**ssl_support_<version>**および**ssl_ciphers** 設定オプションを使用して、セキュア通信に利用できるSSL/TLSバージョンと暗号リストを個々に設定します。

以下の例に示すとおり、スペース、カンマ、またはコロンの区切られたリストで暗号を(望ましい順に)指定します。

```
SSL_DHE_RSA_WITH_AES_128_CBC_SHA,  
SSL_DHE_RSA_WITH_AES_256_CBC_SHA,  
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
```

TLS 1.2 が有効化されている場合、TLS 1.2 コネクションのネゴシエーション時に、仮想サーバまたはプールが許可する署名アルゴリズムのリストを設定することもできます。アルゴリズムのリストの設定は、仮想サーバの「SSL Decryption (SSL 復号化)」ページの **ssl_signature_algorithms**、またはプールの「SSL Encryption (SSL 暗号化)」ページの **ssl_signature_algorithms** で行います。利用可能なアルゴリズムについては、オンラインヘルプを参照してください。

request_client_cert が設定された仮想サーバでは、これらの署名アルゴリズムのうちの 1 つを使用して署名されたクライアント証明書のみが受け入れられます。プールでは、TLS 1.2 コネクションを判別するときに、トラフィックマネージャが署名アルゴリズムのリストをノードに送信します。

これらの項目をデフォルト値の「Use the global setting (グローバル設定を使用)」の設定のままにしたり、**ssl_ciphers** および **ssl_signature_algorithms** の場合にブランクに設定したりすると、トラフィックマネージャは強制的に **System (システム) > Global Settings (グローバル設定) > SSL Configuration (SSL 設定)** で設定されたグローバル値を使用します。

また、TLS 1.0 以降を有効化した場合、トラフィックマネージャは楕円曲線ディフィーヘルマン(ECDHE)鍵共有の使用をサポートします。トラフィックマネージャでは、DHE と同様に、TLS ハンドシェイクで共有秘密情報について合意するために ECDHE が使用されます。その主な利点は、比較的小さい(計算が速い)鍵で同等のセキュリティを実現できることです。

ディフィーヘルマン鍵共有(DHE または ECDHE)を使用する暗号は、使用されている 1 つ以上の署名アルゴリズムがクライアントでサポートされている場合にのみ使用されます。

楕円曲線暗号化の設定

楕円曲線の優先リストを指定するには、**ssl!elliptic_curves** にスペース、カンマ、またはコロンで区切ったリストを入力します。指定されるリストの順序によって、曲線の優先順位が決定されます。トラフィックマネージャでは、クライアントがサポートしている最初の曲線がリストから選択されます。使用できる曲線は以下のとおりです。

- P256
- P384
- P521

ssl!elliptic_curves をブランクのままにした場合、トラフィックマネージャでは「P256, P384, P521」のデフォルトリストが使用されます。

このリストは、個々の仮想サーバまたはプールレベルでオーバーライドできます。仮想サーバの「SSL Decryption (SSL 復号化)」ページの **ssl_elliptic_curves** 設定、またはプールの「SSL Encryption (SSL 暗号化)」ページの **ssl_elliptic_curves** を使用します。

トラフィックマネージャは、内部サーバと管理サーバの通信で使用する、別の楕円曲線優先リストを設定することもできます。このリストを設定するには、**System (システム) > Security (セキュリティ)** で **admin!ssl_elliptic_curves** を使用します。リストが入力されていない場合、トラフィックマネージャでは、代わりに **ssl!elliptic_curves** が使用されます。

クライアント証明書

SSL サーバ証明書は、SSL トランザクションにおいてサーバを識別するために使用されますが、SSL では**クライアント証明書**を使用することもできるため、クライアントは必要に応じてそれ自体の身元を明らかにすることができます。

デフォルトでは、トラフィックマネージャは、仮想サーバに接続するクライアントに対し、クライアント証明書をリクエストしません。必要な場合は、クライアントにクライアント証明書をリクエストするように仮想サーバを設定できます。

SSL トランザクションでクライアントを認証する方法を制御するには、以下の設定を使用します。

- **request_client_cert:** 仮想サーバが各クライアントに身元確認のための証明書をリクエストするかどうか、また証明書の提示をオプションにするか必須(すなわち、全てのクライアントが有効なクライアント証明書を提示する必要がある)にするかを指定することができます。request_client_cert を「Do not request a certificate (証明書をリクエストしない)」に設定した場合(デフォルト設定)、クライアントは、それ自体が証明書を提示しようとしているかどうかに関わらず、証明書を送信することができません。
- **ssl_client_cert_headers:** リクエストの復号化後、トラフィックマネージャは、オプションで HTTP ヘッダを設定して、使用されている SSL 証明書の証明書フィールドを記録することができます。

ヘッダ名	値の例
SSLClientCertVersion	3
SSLClientCertSerialNumber	ED41A8
SSLClientCertIssuer	C=US, ST=CA, L=San Jose, O=MyOrg
SSLClientCertSubject	C=US, ST=CA, L=San Jose, O=MyOrg
SSLClientCertNotValidBefore	1155304575 (unixtime)
SSLClientCertNotValidAfter	1186840575 (unixtime)
SSLClientCertSubjectPublicKey	RSA (2048 bit)
SSLClientCertSignatureAlgorithm	sha256withRSAEncryption
SSLClientCertHash	873FECCB50E0C1D34711ABE65BA70BA7

選択した場合、トラフィックマネージャは、SSLClientCert という名前のヘッダに、PEM 符号化証明書の全体を配置することもできます。

- **client_cas:** クライアント証明書が使用されている場合、トラフィックマネージャは、提示された証明書が正当なものかどうかを検証する必要があります。そのため、トラフィックマネージャは信頼している CA をクライアントに通知し、クライアントはそのいずれかの CA によって署名された証明書を送信します(利用可能な場合)。この設定を使用するには、初めに **Certificate Authorities and Certificate Revocation Lists Catalog** (認証局および証明書失効リストカタログ)で信頼された CA 証明書を設定し、必要に応じてここで CA を選択します。指定した CA によって認識されていない証明書をクライアントが提示した場合、トラフィックマネージャによってコネクションが即座に切断され、クライアントに SSL エラーが送信されます。

OCSP の設定

オンライン証明書ステータスプロトコル(OCSP)を使用すると、OCSP レスポンダと言う中央のサーバで証明書の取り消しステータスを確認することができます。OCSP は、証明書失効リスト(CRL)の代替手段として広く利用されています。詳細については、227 ページの「クライアント認証」を参照してください。

トラフィックマネージャの **Virtual Servers** (仮想サーバ) > **SSL Decryption** (SSL 復号)ページには、OCSP を設定するための専用のセクションがあります。OCSP を設定するには、以下の設定を使用します。

設定	説明
ssl_use_ocsp	yes (はい)に設定すると、この仮想サーバは OCSP を使用して、クライアント証明書が取り消されていないことを確認します。
ssl_ocsp_timeout	トラフィックマネージャは、OCSP レスポンダへの問い合わせ時に、この設定に指定した時間だけレスポンスを待機します。この時間制限内に OCSP レスポンダがレスポンスを返さない場合、クライアントの証明書は拒否されます。

設定	説明
ssl_ocsp_max_response_age	OCSP レスポンスの「thisUpdate」フィールドが、この設定に指定した秒数より古い場合、レスポンスは無効と見なされます。0 (ゼロ) に設定すると、レスポンスが期限切れかどうかを判断するためにレスポンスの「nextUpdate」フィールドが使用されます。 レスポンスの「nextUpdate」フィールドに指定された時間が経過すると、この設定に関わらず、レスポンスは無効と見なされます。
ssl_ocsp_time_tolerance	OCSP レスポンダとトラフィックマネージャのクロックにわずかなずれがある場合、レスポンスの「thisUpdate」および「nextUpdate」フィールドに指定された時間によっては、トラフィックマネージャがそのレスポンスを無効と見なすことがあります。例えばレスポンダのクロックがトラフィックマネージャのクロックより数秒進んでいる場合、トラフィックマネージャがレスポンスの「thisUpdate」フィールドを未来の時間として捉える可能性があります。 この設定では、クロックの差の許容秒数を指定できます。

以下の OCSP レスポンダ設定は、発行者単位で指定します。そのため、認証局カタログに保持されている発行者ごとに、その発行者が署名した証明書に異なる設定を指定することができます。

特定の発行者の設定を追加するには、**Add Issuer Specific Settings** (発行者固有設定の追加) セクションで認証局を選択します。トラフィックマネージャが、明示的に設定されていない発行者によるクライアント証明書を受信した場合は、**Default Settings** (デフォルト設定) が使用されます。

設定	説明
OCSP check (OCSP チェック)	この設定は、発行者の OCSP の確認を有効化/無効化するときに使用します。 「Always (常時)」を選択すると、適切な OCSP レスポンド URL を特定できない場合にリモートコネクションは切断されます。
Use AIA for URL (URL に AIA を使用)	発行者は、X.509 の AIA 拡張フィールドを使用してクライアント証明書に OCSP レスポンド URL を埋め込むことができます。この設定を有効化すると、この拡張が利用可能な場合、トラフィックマネージャは AIA を使用します。それ以外の場合、Fallback URL (フォールバック URL) 設定の URL を使用します。
Fallback URL (フォールバック URL)	レスポンド URL を他の方法で特定できない場合、この URL が使用されます。
Request signing certificate (証明書署名要求)	レスポンドがリクエストの送信元を識別し、認証できるように、OCSP リクエストに署名することができます。SSL 証明書カタログにアップロードされた証明書からリクエストへの署名に使用する証明書を選択するか、Default Settings (デフォルト設定) に設定されている証明書を使用するか、またはリクエストへの署名を無効化することができます。

設定	説明
Use nonce extension (nonce 拡張を使用)	クライアント証明書の取り消し後に攻撃者が「良好な」OCSP レスポンスを返せないようにするために、ユニークなデータ配列を OCSP リクエストおよびレスポンスに追加することができます。nonce 拡張を使用すると、常に「strict」モードを使用して nonce (ランダムな文字列) を返すようレスポンスにリクエストできます。それ以外の場合、トラフィックマネージャは、nonce が含まれていないレスポンスを受け入れません。 重要: この拡張を使用すると、レスポンスは、性能を向上させるために実行するレスポンスの事前生成を行わなくなります。また、この拡張は、全ての OCSP レスポンスがサポートしているわけではありません。サポートされていない場合、レスポンスは「unauthorized」レスポンスを返す可能性があります。
Responder certificate (レスポンス証明書)	OCSP レスポンスは、その正当性を証明するためにレスポンスによって署名されます。レスポンスへの署名に使用する特定の証明書を、SSL Certificate Authorities (SSL 認証局カタログ) にリストされている証明書から選択することができます。 または、「Issuing Certificate (証明書の発行)」を選択して、レスポンスが発行者の証明書によって署名されていることを想定するようシステムを設定することもできます。 あるいは、「Signed by issuing certificate (証明書の発行による署名)」を選択する方法もあります。この場合、証明書が発行者の証明書そのものであるか、発行者によって署名されている必要があります。また、extendedKeyUsage 拡張に id-kp-OCSPSigning マーカおよび id-pkix-ocsp-nocheck 拡張がなければなりません。

SSL セッション ID キャッシュ

SSL ハンドシェイクが実行された後、SSL クライアント(Web ブラウザなど)と SSL サーバ(トラフィックマネージャなど)は、SSL セッション ID を使用して暗号化パラメータを記録します。

SSL TCP コネクションの終了後、クライアントが再接続するときに SSL セッション ID を提示することがあります。サーバがその SSL セッション ID を認識した場合、演算負荷の大きい SSL ハンドシェイクを省略し、キャッシュされている暗号化パラメータを使用して SSL セッションを再開することができます。

SSL セッション ID のキャッシュは、トラフィックマネージャによって管理されます。キャッシュのサイズ(`ssl!cache!size`)と有効期限(`ssl!cache!expiry`)を設定するには、**System (システム) > Global Settings (グローバル設定) > SSL Configuration (SSL 設定)**を参照してください。

SSL セッション ID キャッシュの動作をモニタリングするには、**Activity (アクティビティ) > Current Activity (現在のアクティビティ)**をクリックします。**Cache values (キャッシュ値) > SSL Session ID Cache (SSL セッション ID キャッシュ)**値を選択して現在のアクティビティチャートにプロットするか、SNMP を使用してこれらの値をモニタリングします。**Oldest (最も古いエントリ)**値は、キャッシュ内で使用されていない期間が最も長いエントリの最終使用時点からの経過時間を示します。キャッシュがいっぱいになったときに(**Entries (エントリ)**と **EntriesMax (エントリ最大値)**を比較)、設定されている有効期限より **Oldest (最も古いエントリ)**値が大幅に低い場合、キャッシュが小さすぎるために、早すぎる時期にエントリが期限切れになる可能性があります。

OCSP Stapling キャッシュ

トラフィックマネージャは、`ssl_ocsp_stapling` が有効化されている仮想サーバによって使用されるサーバ証明書のために、OCSP レスポンスのキャッシュを保持します。

管理 UI の **System (システム) > Global Settings (グローバル設定) > SSL Configuration (SSL 設定)** セクションで、トラフィックマネージャの OCSP レスポンスのキャッシング動作を設定することができます。

- `ssl!ocsp_stapling!mem_size` でキャッシュのサイズを設定します。
- `ssl!ocsp_stapling!default_refresh_interval` は、その先の「nextUpdate」時間を伴うレスポンスがキャッシュ内で利用できない場合に、新たな OCSP リクエストが行われる前にトラフィックマネージャにどれだけ待機させるかをコントロールするために使用します。

- `ssl!ocsp_stapling!maximum_refresh_interval` は、OCSP リクエスト間の最大経過間隔を定義するために使用します。有効な OCSP レスポンスを受信すると、OCSP レスポンスの「nextUpdate」時間の方に早く到達しない限り、ここで定義された秒数後に次の更新が行われます。
- トラフィックマネージャには、`ssl!ocsp_stapling!verify_response` の署名検証オプションがあります。このオプションを **Yes (はい)** に設定すると、トラフィックマネージャは、OCSP レスポンスを使用して **Stapling** を行う前に、強制的に OCSP レスポンスの署名および有効期間の認証を行います。認証に失敗すると、トラフィックマネージャはそのレスポンスを OCSP Stapling に決して使用しません。
- OCSP レスポンスには、OCSP レスポンス内の 2 つのフィールド「thisUpdate」および「nextUpdate」で定義された有効期間があります。
`ssl!ocsp_stapling!verify_response` が有効化されているとき、この期間外のレスポンスはキャッシュされないため、**Stapling** されません。環境によっては、クロックスキューが関係していることがあります。そのためにローカルの現在時刻が OCSP レスポンスの「thisUpdate」フィールドよりも早く表示されることがあり、誤って有効性の認証に失敗する原因になります。トラフィックマネージャに、レスポンスが有効であると見なし続けさせるためには、`ssl!ocsp_stapling!time_tolerance` 設定を使用して、この間隔外の許容時間マージン(秒単位)を定義します。

SSL 暗号化

プールは、バックエンド SSL サーバにトラフィックを送信する前に、SSL 暗号化を実行することができます。SSL 暗号化が有効化されている場合、暗号化には、それを実行するトラフィックマネージャの CPU リソースが追加で使用されます。

プールに対して SSL 暗号化を有効化するには、**Pools (プール) > Edit (編集) > SSL Settings (SSL 設定)** に移動します。

以下のオプションを設定することができます。

- **ssl_encrypt:** バックエンドノードへの SSL 暗号化を有効化するには、**Yes (はい)** に設定します。
- **ssl_send_close_alerts:** ソケット切断を開始するときに SSL/TLS 「切断アラート」を送信するには、**Yes (はい)** に設定します。
- **ssl_support_<version>:** このプールがバックエンドノードへの接続に SSL または TLS の専用バージョンを使用できるようにするには、**Enabled (有効)** に設定します。トラフィックマネージャに、**System (システム) > Global Settings (グローバル設定) > SSL Configuration (SSL 設定)** で設定されている同等

のグローバル値を強制的に使用させるには、**Use the global setting** (グローバル設定を使用)を選択します。

- **ssl_ciphers:** バックエンドノードとのセキュア通信に利用できる暗号のリストを指定します。スペース、カンマ、またはコロンで区切られたリストを(望ましい順に)使用します。トラフィックマネージャに、**System** (システム) > **Global Settings** (グローバル設定) > **SSL Configuration** (SSL 設定)で設定されている同等のグローバル値を強制的に使用させるには、リストをブランクのままにします。
- **ssl_client_auth:** バックエンドサーバがクライアント証明書の認証をリクエストする場合、トラフィックマネージャがローカルのクライアント証明書カタログから適切な証明書を選択するよう設定するには、この設定を有効化します。サーバは、サーバが信頼する認証局のリストをクライアントに送信します。トラフィックマネージャは、サーバが信頼する認証局の 1 つによって署名されたと判別した最初のクライアント証明書を選択します。ただし、使用される証明書の明示的なコントロールはありません。
- **ssl_server_name:** トラフィックマネージャに TLS 1.0 「server_name」拡張を試行させるには、**Yes** (はい)に設定します。これにより、バックエンドノードが適切な証明書を提示しやすくなります。この設定を有効化すると、トラフィックマネージャは強制的に TLS 1.0 以上を使用します。
- **ssl_strict_verify:** 中間者攻撃やサーバスプーフィングから保護するために、トラフィックマネージャは、バックエンドサーバが使用する SSL 証明書をカタログ内の CA と照合して検証することができます。証明書の有効期限が切れている場合、証明書の一般名称がノードの IP アドレス、ホスト名、または「ssl_common_name_match」にリストされているいずれかの名前と一致しない場合、カタログ内の CA がその証明書に署名していない場合に、サーバへのコネクションを拒否するようにトラフィックマネージャに指示するには、**Yes** (はい)に設定します。
- **ssl_common_name_match:** サーバによって提示される証明書で、トラフィックマネージャが一般名称として受け入れるホスト名または IP アドレスのリストです。

Update (更新)をクリックして変更を保存します。

SSL 転送における IP アドレスの維持

トラフィックマネージャは、HTTP リクエストを転送するときに、ダウンストリームのサーバがリクエストの正しい送信元 IP アドレスを特定できるように、オプションで X-Cluster-Client-Ip というヘッダを挿入することができます。トラフィックマネージャが HTTPS リクエストを復号化する場合は、後でリクエストを再度暗号化するときでも、X-Cluster-Client-Ip ヘッダをリクエストに挿入できます。

ただし、復号化と再暗号化を行わずに SSL リクエストを転送する場合、トラフィックマネージャは、リクエスト内部のデータを変更することができません。この設定は、復号化のためにコネクションをトラフィックマネージャシステムのクラスタに負荷分散するループバック仮想サーバで使用されます。この場合、プールで **ssl_enhance** 設定を使用することにより、独自のヘッダを SSL コネクションに追加し、ここに送信元 IP およびポート、宛先 IP およびポートなど維持されない重要なコネクションデータを保持することができます。

ssl_enhance が設定された「拡張」SSL コネクションを受信するトラフィックマネージャシステムには、仮想サーバの SSL 復号化設定の **ssl_trust_magic** を設定する必要があります。この設定により、トラフィックマネージャは、独自のヘッダを取り出し、送信元 IP およびポート、宛先 IP およびポートなどの正しいコネクションデータを認識できるようになります。

SSL 暗号処理専用デバイスの使用

トラフィックマネージャソフトウェア製品は、Thales e-Security 社の nShield Connect など、RSA Security Inc. の PKCS #11 Cryptographic Token Interface (Cryptoki) に準拠した SSL ハードウェアをサポートしています。

トラフィックマネージャ仮想アプライアンスは、Thales e-Security 社の nShield Connect や nCipher netHSM ネットワーク型 HSM をサポートしています。

全てのトラフィックマネージャは、Microsoft Azure Key Vault サービスをサポートしています。Azure Key Vault は、従来のハードウェアセキュリティモジュールに代わるクラウドベースサービスとして提供されています。Azure Key Vaults では、以下の条件に従ってください。

- トラフィックマネージャは、それ自体によって作成された鍵、または Azure Key Vault に移行された有効な既存の RSA 鍵の使用をサポートしています。
- トラフィックマネージャは、「RSA-HSM」タイプの鍵をサポートしています。このタイプの鍵は、Azure Key Vault サービスでセキュアなハードウェアに保存されます。
- RSA-HSM 鍵を作成し、使用するには、Azure Key Vault の「Premium」サービスレベルを使用する必要があります。利用可能なサービスレベルの詳細と価格については、<http://azure.microsoft.com/> を参照してください。
- 他のセキュアハードウェアのサポートと同様に、トラフィックマネージャは、SSL 復号化(仮想サーバ)または SSL 暗号化(プール)用として Azure Key Vault に保存された鍵を使用できます。トラフィックマネージャは、SSH または DNSSEC などの他の目的のために、保存された鍵を使用することはできません。

- トラフィックマネージャが FIPS モードを使用するように設定されている場合 (CHAPTER 31 の「トラフィックマネージャ」を参照)、Azure Key Vault REST API との通信も FIPS モードになります。
- Azure Key Vault を使用するには、トラフィックマネージャの DNS で、Azure Key Vault サーバと Microsoft のログインサーバの両方のホスト名を解決できる必要があります。トラフィックマネージャは、ポート 443 で送信 TCP コネクションを確立できる必要もあります。トラフィックマネージャシステムがファイアウォールによる制限を受けている場合は、この通信チャネルが許可されるようにファイアウォールを設定する必要があります。

注: Azure Key Vault の設定と管理については、<http://azure.microsoft.com/en-us/documentation/services/key-vault/>を参照してください。

HSM デバイスまたは Azure Key Vault サービスを使用すると、トラフィックマネージャシステムの CPU から接続先の SSL 暗号処理専用デバイスに、SSL 演算(RSA 秘密鍵の復号化)がオフロードされます。ハードウェアで鍵管理を行う PKCS#11 デバイスもあり、この場合、秘密鍵はハードウェアデバイスにセキュアに保存され、適切な認証がなければ直接アクセスすることはできません。

SSL 暗号処理専用デバイスまたはサービスで実行されている RSA 暗号動作はトラフィックマネージャの FIPS 140-2 Cryptographic Boundary の対象ではないため、暗号処理専用デバイスまたはサービスが FIPS 140-2 に十分に適合していることを、必要に応じて個別に確認する必要があります。

注: 仮想アプライアンスなどトラフィックマネージャ製品によっては、サポートされるハードウェアとサービスが異なるため、以下のオプションの一部のみしか表示されない場合があります。

注: SSL デバイスを使用しても、トラフィックマネージャシステムの全体的な性能が向上しない場合があります。RSA 演算はメイン CPU コアからオフロードされますが、デバイスとの通信オーバーヘッドも無視できません。トラフィックマネージャは、数千または数万の SSL 演算を汎用 CPU で実行することができます。多くの SSL デバイスを使用する最大のメリットは、秘密鍵をセキュアに保存できることです。

SSL デバイスを使用するためのトラフィックマネージャの設定

SSL デバイスまたはサービスを使用するようにトラフィックマネージャを設定するには、**System (システム) > Global settings (グローバル設定)**をクリックし、**SSL Hardware Support (SSL ハードウェアサポート)**をクリックします。**ssld!library** で必要な SSL デバイスを選択します。

The type of SSL hardware to use. The drivers for the SSL hardware should be installed and accessible to the traffic manager software.

☒ None
☐ Microsoft Azure Key Vault
☐ PKCS#11 (e.g. nCipher NetHSM) ...

ssid!library: Location of PKCS#11 Library:
The User PIN for the PKCS#11 token:
Security token type:
Security token label (required when multiple tokens available):

Appliances can use an nCipher NetHSM as SSL hardware. Where this is used, leave ssid!accel set to No.

Connect to NetHSM

The Microsoft Azure Key Vault can be used to provide cryptographic services.

Connect to Microsoft Azure Key Vault

Whether or not the SSL hardware is an "accelerator" (faster than software). By default the traffic manager will only use the SSL hardware if a key requires it (i.e. the key is stored on secure hardware and the traffic manager only has a placeholder/identifier key). With this option enabled, your traffic manager will instead try to use hardware for all SSL decrypts.

ssid!accel: ☐ Yes ☒ No Default: No

The number of consecutive failures from the SSL hardware that will be tolerated before the traffic manager assumes its session with the device is invalid and tries to log in again. This is necessary when the device reboots following a power failure.

ssid!failure_count: Default: 5

Fig 44. SSL ハードウェアアクセラレータの使用に関する設定

以下の表に、利用可能な設定を示します。

設定	説明
ssld!library	Azure Key Vault サービスを使用することをトラフィックマネージャに指示するには、「Microsoft Azure Key Vault」に設定します。Azure Key Vault に接続するには、「Connect to Microsoft Azure Key Vault (Microsoft Azure Key Vault に接続)」をクリックします。 Thales e-Security nShield Connect などの汎用 PKCS 準拠ハードウェアの使用をトラフィックマネージャに指示するには、「PKCS#11」に設定します。NetHSM デバイスに接続するには、「Connect to NetHSM」をクリックします。 PKCS#11 の追加設定は以下のとおりです。 ▪ Location: トラフィックマネージャで PKCS#11 インターフェースライブラリを検索する必要があるシステムロケーション。トラフィックマネージャで標準のシステムロケーションを検索するには、このフィールドを空白のままにします。 ▪ User PIN: SSL ハードウェアでユーザセッションを作成するために必要な PIN です。 ▪ Token Type: トラフィックマネージャで SSL 秘密鍵の保護に使用されるセキュリティトークンの形態です。 ○ Operator Card Set: トラフィックマネージャに接続される適切なカードリーダー付きの物理的なスマートカードです。このスマートカードには、オプションのユーザ PIN が必要になる場合があります。 ○ Soft Card: 物理的なスマートカードと同様に動作するディスク上のファイルです。このファイルにはユーザ PIN が必要です。 ○ Module Protected: スマートカードやユーザ PIN を使用せずに単独で鍵を保護する暗号モジュールです。 ▪ Token Label: (「Operator Card Set」と「Soft Card」のトークンタイプのみ)使用するカードの識別ラベルまたは識別名を指定するには、この設定を使用します。

設定	説明
ssld!accel	全ての復号化動作で、このページで定義した SSL デバイスを使用するかどうかを指定します。全ての動作で SSL デバイスを使用するようにトラフィックマネージャに指定するには、「Yes (はい)」に設定します。「No (いいえ)」に設定すると、トラフィックマネージャは、秘密鍵が SSL ハードウェアにセキュアに保存されている場合のみそのハードウェアを使用します。
ssld!failure_count	(リブート、停電などの場合に)トラフィックマネージャがセッションを無効と見なして再度ログインを試行するまでに実行できる連続試行回数です。

SSL デバイスの正常な動作の検証

Diagnose (診断) ページには、トラフィックマネージャが SSL デバイスと通信可能かどうかについてのチェック結果が表示されます。全てのエラーまたは警告が示されます。

ハードウェアによる鍵管理をサポートしている SSL デバイスを使用するようにトラフィックマネージャを設定した場合は、以下の操作を実行できます。

- トラフィックマネージャを使用して、そのデバイスに新しい秘密鍵(およびそれに対応する公開証明書)を作成できます。新しい自己署名サーバまたはクライアント SSL 証明書を作成するには、230 ページの「自己署名 SSL 証明書の新規作成」の手順に従います。この手順では、鍵管理デバイスが設定されているときに表示される「Create private key on hardware (ハードウェアで秘密鍵作成)」オプションを選択してください。
- ハードウェアからトラフィックマネージャに手動で秘密鍵と公開証明書をインストールできます。ハードウェアデバイスの手順に従って、符号化された秘密鍵と公開証明書を抽出します。次に、関連する **SSL Server Certificates** (SSL サーバ証明書) カタログと **SSL Client Certificates** (SSL クライアント証明書) カタログの **Import Certificate** (証明書インポート) ツールを使用して、抽出した秘密鍵と公開証明書をトラフィックマネージャにインストールします。
- トラフィックマネージャから新たに接続した SSL デバイスに、既存の秘密鍵を移行します。鍵を移行するには、関連する証明書編集ページの「Migrate certificate to Hardware Security Module (ハードウェアセキュリティモジュールへの証明書の移行)」セクションを使用します。

重要: 攻撃者がすでに秘密鍵のコピーを入手している場合、トラフィックマネージャから SSL デバイスに秘密鍵を移行しても、その保護は保証されません。セキュリティを最大限にするために、SSL デバイス自体で生成した鍵を使用することをお奨めします。

トラフィックマネージャによって、ハードウェアで管理される鍵にアクセスして使用できることが検証されます。問題が発生した場合は、エラーが **System** (システム) > **Diagnose** (診断) ページと **Event log** (イベントログ) に表示されます。

注: **Diagnose** (診断) ページと **Event log** (イベントログ) で、クライアント証明書とそれに対応する秘密鍵に **使用不可** のマークが付けられることがあります。これらの証明書および鍵を使用するプールに問題があることを示すマークが付くことはありません。これは、トラフィックマネージャが、クライアント証明書をリクエストするバックエンド SSL ノードに接続するまで、どのクライアント証明書を求められるかが分からないためです。

Connect to Microsoft Azure Key Vault (Microsoft Azure Key Vault に接続) ウィザードの使用

Azure Key Vault に接続するには、**System** (システム) > **Global Settings** (グローバル設定) > **SSL Hardware Support** (SSL ハードウェアサポート) で、「Connect to Microsoft Azure Key Vault (Microsoft Azure Key Vault に接続)」をクリックします。

ただし、このウィザードをアクティブにする前に、以下の情報を入手してください。

- **Azure Key Vault の URL:** 指定された Azure Key Vault の URL(「<https://dev.vault.azure.net>」など)。
- **クライアント ID:** Azure Key Vault でのユーザアカウントの識別子(ID)。
- **クライアントのパスワード:** ユーザアカウントのパスワードを表す英数字の文字列。
- **信頼された認証局:** (オプション) 「中間者」攻撃から Azure Key Vault への接続を保護するために、Azure Key Vault REST API によって使用される SSL 証明書の ID を検証するようにトラフィックマネージャに指示することができます。Microsoft の SSL 証明書に署名した信頼された認証局の情報を入手し、ウィザードを実行する前にトラフィックマネージャの CA カタログに保存します。

この情報の入手については、サポート窓口に問い合わせるか、以下の場所にある Microsoft Azure Key Vault のドキュメントを参照してください。

<http://azure.microsoft.com/en-us/documentation/services/key-vault/>

他のトラフィックマネージャウィザードと同様に、「Connect to Microsoft Azure Key Vault (Microsoft Azure Key Vault に接続)」ウィザードでは、一連のステップが

実行されます。各ステップ間を移動するための、**Back** (戻る) ボタンと **Next** (次へ) ボタンが表示されます。

1. はじめに
2. Azure Key Vault の URL を入力します。

Connect to a Microsoft Azure Key Vault, step 2 of 5

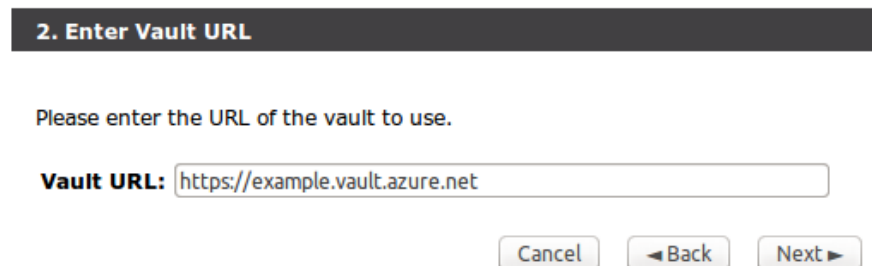


Fig 45. Azure Key Vault の URL の入力

3. クライアントのクレデンシャルを入力します。

Connect to a Microsoft Azure Key Vault, step 3 of 5

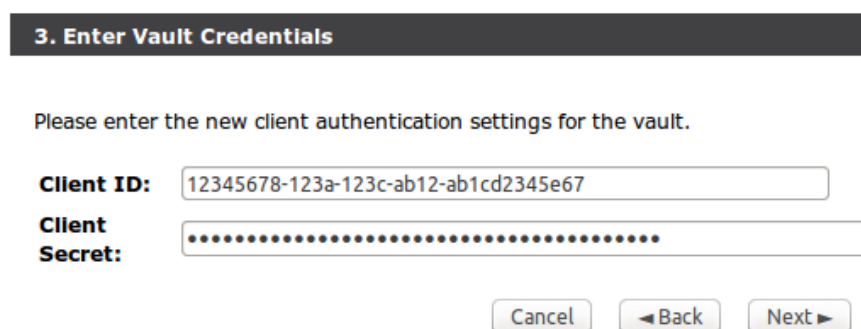


Fig 46. クライアントのクレデンシャルの入力

4. **Yes** (はい)を選択して、Azure Key Vault REST API の SSL 証明書を検証するようにトラフィックマネージャに指示するか、**No** (いいえ)を選択して検証を無効にします。

Connect to a Microsoft Azure Key Vault, step 4 of 5

4. Enable SSL Certificate Validation

The connections between the traffic manager and the REST API for the Microsoft Azure Key Vault use TLS encryption to ensure data confidentiality.

To protect these connections against 'man in the middle' attacks the SSL certificates of the REST API should be validated. This requires the trusted certificate authority that signed Microsoft's SSL certificates to be present in SSL Certificate Authority Catalog of the traffic manager. Please ensure this is the case before using this wizard.

☒ Yes
☐ No

Cancel

◀ Back

Next ▶

Fig 47. Azure Key Vault REST API の SSL 証明書の検証

5. **Finish** (完了)をクリックし、表示されている設定を使用して Azure Key Vault に接続します。

接続に成功した場合、SSL Hardware Support (SSL ハードウェアサポート)セクションが更新され、アクティブな Azure Key Vault コネクションが表示されます。

The type of SSL hardware to use. The drivers for the SSL hardware should be installed and accessible to the traffic manager software.

☐ None
☒ Microsoft Azure Key Vault
☐ PKCS#11 (e.g. nCipher NethSM) ...

ssld!library: Location of PKCS#11 Library:
The User PIN for the PKCS#11 token:
Security token type:
Security token label (required when multiple tokens available):

Appliances can use an nCipher NethSM as SSL hardware. Where this is used, leave ssld!accel set to No.

Connect to NethSM

The Microsoft Azure Key Vault can be used to provide cryptographic services.

Connected to Key Vault: <https://example.vault.azure.net>

Fig 48. 接続された Azure Key Vault

ウィザードに戻って別の設定を行うには、**Edit** (編集)をクリックします。アクティブな Azure Key Vault からトラフィックマネージャを切断するには、**Remove** (削除)をクリックします。

Azure Key Vault に関連する接続失敗、警告、およびエラーは、**Diagnose** (診断ページ)とイベントログでレポートされます。

Connect to NetHSM ウィザードの使用

注: このセクションは、トラフィックマネージャアプライアンス製品のみを対象としています。

トラフィックマネージャには、Thales 社の nShield Connect や nCipher netHSM ネットワーク型 HSM で安全に通信を有効化するウィザードが用意されています。このウィザードで、プロセスが複数の基本ステップにまとめられ、単純化されます。

注: nShield Connect は nCipher netHSM デバイスの後継です。ウィザード全体をとおして、netHSM と呼ぶ場合は両者を意味すると理解してください。現在、Brocade はこれらのデバイスのみをサポートしています。詳細については、サポート窓口にお問い合わせください。

ウィザードを実行するには、前提条件として特定の情報が必要です。以下の情報の取得方法については、nShield Connect のドキュメントを参照してください。

- Security World が適切に設定されている nShield Connect デバイスの IP アドレス。
- nShield Connect が使用するリモートファイルシステムの IP アドレス。
- Operator Card Set (nShield Connect に挿入されたカードと、そのパスフレーズ)。
- nShield Connect で作成された 1 つ以上の「PKCS#11」タイプの鍵。このカードセットで保護され(モジュール保護ではない)、トラフィックマネージャで使用する鍵であり、外部アプリケーションで使用するために nShield Connect ソフトウェアで生成されるファイルも含まれます。

ウィザードを起動する前に、nShield Connect で使用するスマートカードに対応する正しい PKCS#11 トークン PIN を入力したことを確認する必要があります。管理 UI の **System** (システム) > **Global Settings** (グローバル設定) ページで、**SSL Hardware Support** (SSL ハードウェアサポート) セクションを見つけます。**ssld!library** 設定キーの PKCS#11 オプションの隣にあるラジオボタンをクリックし、該当するボックスに PIN を入力します。要件は個別に異なるかもしれませんが、他の設定は通常ここでは関係ありません。

ウィザードを起動するには、**Connect to NetHSM** をクリックします。

ウィザードはいくつかのステップで実行され、前後のステップに移動するための **Back** (戻る) ボタンと **Next** (次へ) ボタンが各ステップにあります。

以下の 4 つのステップがあります。

1. ウィザードの導入部が表示されます。

2. アプライアンスで Thales/nCipher Support Software を使用する際に必要な使用許諾書が表示されます。
3. netHSM とリモートファイルシステムの IP アドレスを入力します。
4. netHSM で表示されている ESN と HKNETI をフロントパネルと比較して、netHSM の ID を確認します。これで、トラフィックマネージャが悪意のある第三者ではなく、間違いなく目的の netHSM に接続していることが確認されます。フロントパネルを確認する際には、許可されるクライアントとしてトラフィックマネージャアプライアンスの適切な IP アドレスが netHSM に追加されていることも確認してください。



Fig 49. Connect to NetHSM ウィザードのステップ4

5. **Confirm** (確認)チェックボックスを選択し、**Finish** (完了)をクリックして設定を適用します。設定が全て正しいければ、成功のページが表示されます。設定に何か問題があれば、以下の図のように接続失敗などのエラーが表示されます。



Fig 50. 接続失敗のレポート

一般的には、以下のようなエラーがあります。

- **PKCS#11 PIN incorrect: System (システム) > Global Settings (グローバル設定)** ページに設定されている PKCS#11 の User PIN が正しくありません。
- **PKCS#11 could not connect to hardware:** nCipher PKCS#11 ライブラリが netHSM/nShield Connect への接続に失敗し、それ以上の情報はトラフィックマネージャに返されませんでした。ESN と HKNETI が正常に取得された後でウィザードにこれが表示された場合は、netHSM がこのアプライアンスのコネクションを許可するように設定されていない可能性が高いと考えられます。このメッセージを後からログで見ると、使用しているデバイスが通信不可になったと考えられます。トラフィックマネージャをリスタート/リブートさせることによって強制的に再接続が試行され、状況によってはこの問題が解決することがあります。
- **Failed to synchronize nCipher Remote File System:** nCipher ツールが、入力された IP アドレスからリモートファイルシステムのデータのコピーを取得できませんでした。これはリモートファイルシステムが通信不可であるか、アプライアンスと同じ nShield Connection に関連付けられていないか、アプリケーションのコネクションを許可するように設定されていない(またはトラフィックマネージャアプライアンスが使用できない nToken を要求するように設定されている)場合に発生します。

ウィザードが正常に完了すると、**Global Settings (グローバル設定)** ページの「Connect to NetHSM」リンクが変化し、コネクションの完了が示されます。

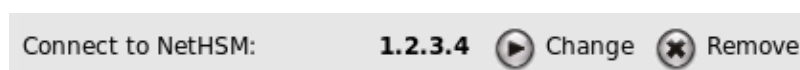


Fig 51. netHSM デバイスに正常に接続した状態

セキュアなデバイスに保存された鍵と証明書の識別

セキュアなハードウェアまたは Azure Key Vault に正常に接続した後、他の鍵ペアと同様に、デバイスの鍵および証明書ファイルを使用できます(証明書の編集を含みます)。デバイスが正常に動作している場合、トラフィックマネージャは秘密鍵が必要な全ての暗号化演算をデバイスに委任します。管理 UI で、鍵がセキュアなハードウェアまたは Azure Key Vault に保存されたことが以下のように示されます。

Which SSL certificate should this virtual server use?

certificate:	Name	Site (CN)	Signed by (Issuer)	Expires
 	example	example.cam.zeus.com	(self-signed)	02 May 2007
	(no certificate specified)			

 = private key stored on secure hardware

Fig 52. 鍵が安全なハードウェア上に保存されていることを示す「錠前型」のアイコン

CHAPTER 14 ヘルスマニタリング

本章では、トラフィックマネージャのヘルスマニタリング機能について説明します。この機能を使用すると、サーバノードが正常に動作しているかどうかをモニタリングし、エラーが検出された場合は、アラートを報告し、障害が発生したノードを迂回してトラフィックをルーティングすることができます。

どのノードをモニタリングするか

トラフィックマネージャは、パッシブモニタリング(使用されているノードのステータスを確認)とヘルスマニタリング(定期的にノードに対して実行される追加テスト)の2通りの方法でノードが正常に動作しているかどうかをモニタリングします。

パッシブモニタリングテストは、アクティブに使用されているノードに対してのみ実行されます。

ヘルスマニタは、プール内の全てのノードに対して実行されます。実行対象は、以下の使用されているプールのみです。

- 仮想サーバのデフォルトプールとして設定されているプール
- TrafficScript または RuleBuilder ルールで `pool.select()` または `pool.use()` 関数によって明示的に名前参照されているプール
- 使用中のプールの障害プールとして設定されているプール

トラフィックマネージャの設定によりこの方法で参照されていないプールは、「使用されていない」と見なされ、ヘルスマニタでモニタリングされず、ノードの障害は検出されません。

注: プールを TrafficScript ルールで使用するときは、明示的に名前参照する必要があります。デフォルトでは、トラフィックマネージャで変数(`pool.use($name);`)またはその他の方法によって参照することはできません。

trafficscript!variable_pool_use 設定(Global Settings (グローバル設定)ページ)を有効化すると、プール名を変数で指定することができます。この設定が有効化されている場合、トラフィックマネージャは、明らかに使用中のプールだけでなく、設定されている全てのプールに対してヘルスマニタを実行します。

複数プールでのノードの使用

複数のプールで同じノード(IP アドレスおよびポート)を参照することができます。1つのプールで障害が発生したノードは、そのプールでは負荷分散またはセッション維持の決定に使用されません。ただし、他のプールに割り当てられたパッシブモニ

タおよびヘルスモニタが障害を報告するまで、これらのプールでは使用される場合があります。

例

Web アプリケーションが 4 台のバックエンドサーバ(ノード)で構成されているとします。各ノードでは、同じ動的コンテンツと静的コンテンツがホストされています。あるノードの動的コンテンツで障害が発生したときに(**Java Servlet** がクラッシュするなど)、そのノードの他の静的コンテンツは引き続き使用したい場合があります。

- それぞれに 4 台の全てのノードが含まれる「**Dynamic**」および「**Static**」という 2 つのプールを作成します。
- 動的コンテンツ(**Java Servlet**、**PHP**、**ASP** ファイルなど)には「**Dynamic**」プールを使用し、その他全てのコンテンツには「**Static**」プールを使用するルールを作成します。
- 「**Dynamic**」プール内のあるノードで障害が発生し、有効な **HTTP** レスポンスを送信できなくなった場合、そのプールで使用されているパッシブモニタリングにより、そのノードで障害が発生したと判断されます。以降、そのノードには「**Dynamic**」プールからトラフィックが送信されなくなります。ただし、そのノードが「**Static**」プールからのリクエストに対して有効な **HTTP** レスポンスを返している限り、「**Static**」プールからは、そのノードに引き続きトラフィックが送信されます。
- さらにきめ細かくエラーを検出するために、各プールに異なるヘルスモニタを割り当てることができます。例えば、「**Dynamic**」プールのヘルスモニタから擬似的な **PHP** または **ASP** リクエストを送信できます。これらのモニタが有効なレスポンスを受信できなかった場合、そのノードは「**Dynamic**」プールで障害が発生したと見なされますが、他のプールでは障害はないと見なされます。

パッシブヘルスモニタリング

プールは、リクエストをノードに送信するたびに一連のチェックを実行します。このプロセスを「パッシブモニタリング」と言います。

- トラフィックマネージャがノードに接続しようとするときに、コネクションが拒否されるか、**max_connect_time** 設定の時間内(デフォルトでは 4 秒)にコネクションが確立されなかった場合、そのリクエストは失敗したと見なされます。
- トラフィックマネージャはコネクションによってデータを書き込むときに、処理の完了前にコネクションが切断されるか、**max_reply_time** 設定の時間内(デフォルトでは 30 秒)にレスポンスの最初のデータが受信されない場合、リクエストは失敗したと見なされます。

- **SSL のみ:** ノードとの SSL ハンドシェイクに失敗した場合、リクエストは失敗したと見なされます。

注: `max_connect_time` および `max_reply_time` 設定は、プールの **Connection Management** (コネクション管理) 設定 (CHAPTER 5 の「コネクション管理」セクションを参照) のプロパティです。

失敗したリクエストの再試行

これらのチェックが失敗した場合、プールは、稼働中の別のノードに対してリクエストを試行し、リクエストを破棄する前にプール内の全てのノードを試すことがあります。この動作は、リクエストのべき等性(idempotent)の状況によって決まります。

デフォルトでは、リクエストはべき等であると見なされます。すなわち、悪影響が生じることなく複数回安全に再試行することができます。例外は、*generic* タイプのプロトコル(「Generic Client First」、「Generic Server First」、「Generic Streaming」)のいずれかを使用して仮想サーバを通じて受信されるリクエストです。デフォルトでべき等にするためには、障害を測定して再試行を実行できるように、初めにリクエストへのエンドポイントを定義⁹する必要があります。

RFC 2616 では、一部の HTTP リクエストが非べき等¹⁰として定義されています(トラフィックマネージャが発生する場合やサーバのステータスが変更される場合など)。トラフィックマネージャはこれらの勧告に従い、HTTP GET、HEAD、PUT、DELETE、OPTIONS、および TRACE メソッドをべき等として扱い、その他全てのリクエストを非べき等と見なします。

- **べき等(悪影響なし)** - トラフィックマネージャは、動作していると考えられる他のノードに対してこれらのリクエストを再試行し、リクエストを破棄する前に、プール内の全てのノードに対してリクエストを試行することができます。
- **非べき等(悪影響あり)** - トラフィックマネージャは、障害が発生したノードに対する TCP コネクションの確立に失敗した場合のみ、非べき等リクエストを再試行します。

注: トラフィックマネージャは、TCP コネクションを確立すると、直ちにそのコネクションによってリクエストデータを書き込みます。送信先のノードがリクエストデータを受信し、その処理を開始したかどうかを特定することはできません。そのため、非べき等リクエストは、最初の段階でコネクションを確立できなかった場合のみ再試行されます。

トラフィックマネージャによるべき等/非べき等の判断を変更するには、TrafficScript 関数の `request.setIdempotent()` および `http.setIdempotent()` を使用

⁹ これは、`request.endsWith()` または `request.endsAt()` のような関数を使用して、TrafficScript で実現されます。

¹⁰ <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

して、特定のリクエストを安全に再試行できると見なすようトラフィックマネージャに指示を出します。

503 サーバエラー

503 サーバエラーレスポンスは扱いが異なります。これは、通常、サーバエラーの原因となったリクエストをクラスタ内の全てのノードに対して試行することは望ましくないからです。

503 サーバエラーの原因となったリクエストは、以下の場合のみ再試行されます。

- パッシブモニタリングが有効化されている
- セッション維持が設定されていない
- リクエストがベキ等である(悪影響なし)

リクエストは、503 エラーがクライアントに返されるまでに最大 3 回試行されます。

ノード障害

トラフィックマネージャは、ノードへのコネクションが、中間トランザクションが成功することなく **node_connection_attempts** 設定の回数(デフォルトでは 3 回)だけ連続して失敗すると、そのノードで障害が発生したと推測します。

障害が発生したと見なされたノードは、少なくとも **node_fail_time** 設定の時間(デフォルトでは 60 秒)は使用されません。その後、復旧したかどうかを判断するために暫定的に使用されます。ただし、プールの全てのノードで障害が発生している場合、**node_fail_time** 設定に関わらず、トラフィックマネージャは復旧したノードに直ちにトラフィックを送信します。

注: **node_connection_attempts** および **node_fail_time** 設定は、**Pool (プール) > Connection Management (コネクション管理)** ページにあります。

Virtual Server (仮想サーバ) > Connection Management (コネクション管理) ページの **Connection Error Settings (コネクションエラー設定)** セクションで **log!server_connection_failures** 設定を有効化することができます。これにより、実際にどのようなノード障害が発生したかに関する役立つログ情報を利用できるようになります。詳細については、CHAPTER 4 の「エラーの処理」セクションを参照してください。

パッシブモニタリングの有効化と無効化

パッシブモニタリングはデフォルトで使用されますが、プール設定の **Monitors (モニタ)** セクションの **passive_monitoring** 設定を使用することによりプール単位で無効化することができます。この設定が無効化されている場合、適切なヘルスマニタが設定されていることを確認してください。設定されていないと、リクエストの失敗が検出されず、再試行されません。

ヘルスマニタの概要

パッシブモニタリングによる推測に加え、定期的にテストを実行し、各ノードが正常に動作しているかどうかを検証する明示的なヘルスマニタをプールに設定することができます。トラフィックマネージャはヘルスマニタを使用することで、トラフィックが処理されていないときでも障害を検出できます。

テストは、設定された間隔でプール内の各ノードに対して実行されます。頻繁にテストで不合格になったノードには使用不可のマークが付けられ、トラフィックマネージャからトラフィックが送信されなくなります。プール全体のモニタでは単一のマシンのみがテストされ、その結果からプール全体が使用不可であることを示すことができます。

モニタはモニタカタログに保持され、任意のプールに適用することができます。各モニタでは、各ノードに対する Ping の実行などの単純なテストから、該当するポートが開いていてノードがホームページなど指定されたデータを処理できるかどうかを確認する高度なテストまで、特定のテストが実行されます。

モニタカタログ

モニタの設定

モニタカタログにアクセスするには、管理サービインターフェースの任意のページにある **Catalogs** (カタログ) ボタンをクリックします。**Monitors Catalog** (モニタカタログ) セクションを展開すると、既に設定されているモニタが表示されます。

Monitors Catalog (モニタカタログ) をクリックすると、カタログに移動します。

トラフィックマネージャには、多数のモニタがプリセットされています。これには、一般的な TCP の Client First または Server First プロトコルを処理するモニタと、プロトコル固有のモニタがあります。例えば、内蔵の **Full HTTP** モニタは、指定されたページを Web サーバにリクエストし、レスポンス内で適切なステータスコードを検索します。

全てのモニタについて、以下の設定を行うことができます。

- **delay** - モニタは、各ノードに対して繰り返し実行されます。モニタリングが完了するかタイムアウトになると、トラフィックマネージャは、ここに設定された時間だけ一時停止してから再度モニタリングを試行します。
- **timeout** - モニタがこのタイムアウト時間内に完了できなかった場合、中断され、失敗したと判断されます。
- **failures** - 一時的なネットワーク異常など繰り返されない予測外の原因により、モニタが失敗することがあります。この設定に指定された回数だけ連続してモニタリングに失敗すると、モニタリング対象ノードで障害が発生したと判断されます。

- **verbose** - この設定により、エラーログへの詳細モニタロギングが有効化されます。モニタのテストには役立ちますが、エラーログファイルがいっぱいになる可能性があるため、商用システムでの使用や長時間の使用は避けてください。

内蔵モニタを編集するには、モニタの名前をクリックし、タイミング、失敗条件、およびロギングと、その他のモニタ固有の設定(リクエストする Web ページなど)を変更します。**Update** (更新)をクリックして変更内容を適用します。

Copy Monitor (モニタのコピー)をクリックして内蔵モニタを新しい名前でコピーし、コピーに変更を加えることもできます。

内蔵ヘルスモニタ

トラフィックマネージャの**モニタカタログ**には、多数のモニタがあらかじめ定義されています。これらの内蔵モニタは、様々なプロトコルについて標準テストを実行します。

基本モニタ

基本モニタは、ノードに対して単純なテストを実行します。以下に例を挙げます。

- **Ping:** Ping モニタは、各ノードに対して Ping リクエストを実行します。レスポンスが受信されない場合、失敗と判断されます。
- **Connect:** ノードへの TCP コネクションを実行します。サーバがポートで監視してトラフィックを受け入れていることの確認を目的としています。このモニタでは、一部のプログラム(exim など)でエラーが出力されることがあります。この場合は、よりプロトコル固有のモニタを使用してください。

高機能モニタ

高機能モニタは、ノードに対してより複雑なリクエストとレスポンスのテストを実行します。他のモニタのツールとして使用され、SSL を使用して接続してリクエストおよびレスポンスを転送することができます。以下に例を挙げます。

- **Client First:** これは、TCP トランザクションモニタに基づくモニタです。ノードにデータを送信し、レスポンスを指定の正規表現と照合します。このモニタを使用するには、送信データ(**write_string**)を設定する必要があります。
- **Server First:** このモニタでは、最初にサーバがデータを書き込むサービスをチェックすることができます。ノードに接続し、サーバが何らかのデータを返すことを確認します。サーバが特定データを返していることを確認するよう簡単に変更できます。
- **Full HTTP / HTTPS:** このモニタでは、HTTP または HTTPS リクエストを実行できます。ホストヘッダ、URL、および認可パラメータを全て設定できます。

レスポンスの戻りコードは、**status_regex** の値と照合されます。例えば、200 の戻りコードが予期される場合は、**status_regex** を「`^200$`」に設定します。または、「`^[234][0-9][0-9]$`」を使用して、200～499 のエラーを受け入れます。

レスポンスボディはオプション設定の **body_regex** と照合されます。

SSL モニタが割り当てられたプールで **ssl_server_name** が有効化されている場合、モニタは、ノード名とともに TLS 拡張のサーバ名の指定(SNI)をバックエンドに送信します。

プロトコル固有のモニタ

プロトコル固有のモニタは、ノードに対して特定のテストを実行します。これには以下のようなアプリケーションがあります。

- **DNS:** これはプログラムモニタの例です。ノードに対して DNS リクエストを行い、返されたアドレスを検証するプログラム (ZEUSHOME/zxtm/conf/scripts/dns.pl)を実行します。
- **FTP:** このモニタは、TCP トランザクションモニタに基づきます。単純な FTP コネクションを確立し、サーバが正常にレスポンスを返していることを確認します。このとき、サーバからバナーが返されることのみを確認し、認証は行いません。
- **POP:** このモニタは、TCP トランザクションモニタに基づきます。ノードに対して初期コネクションを確立し、正しい POP バナーが返されることを確認します。
- **SMTP:** これは POP モニタと同じですが、SMTP サーババナーが返されることを確認します。
- **シンプル HTTP および HTTPS:** これらのモニタは、各ノードにホームページをリクエストします。レスポンスコードが 2xx、3xx、または 4xx であることを確認し、これらのレスポンスコードが返されない場合は、失敗を報告します。
- **RTSP:** このモニタは、サーバに RTSP DESCRIBE リクエストを送信します。レスポンスコードが 2xx、3xx、または 4xx であることを確認し、これらのコードが返されない場合は、失敗を報告します。一部の RTSP サーバでは、このような DESCRIBE リクエストにパスが存在していることが必要になります。この場合は、**rtsp_path** 設定にパスを入力する必要があります。RTSP のプロトコル固有の追加設定は以下のとおりです。
 - **rtsp_path:** DESCRIBE リクエストで使われるパスです。これは、ストリーミングされるマルチメディアファイルのパスです。
 - **rtsp_status_regex:** RTSP レスポンスが一致する必要があるステータスコードの正規表現です。
 - **rtsp_body_regex:** RTSP レスポンスボディが一致する必要がある正規表現です。

- **SIP (UDP):** このモニタは、Max-Forwards を 0 に設定してサーバに SIP OPTIONS リクエストを送信します。レスポンスコードが 2xx、3xx、または 4xx であることを確認し、これらのコードが返されない場合は、失敗を報告します。
 - **sip_status_regex:** レスポンスが一致する必要があるステータスコードの正規表現です。
 - **sip_body_regex:** SIP レスポンスボディが一致する必要がある正規表現です。
 - **udp_accept_all:** このモニタが任意の IP およびポートからレスポンスを受け入れるかどうかを指定します。
- **SIP (TCP):** このモニタは、SIP-UDP モニタと同じリクエストを使用しますが、そのリクエストを TCP コネクションで送信します。

カスタムヘルスモニタ

Monitors Catalog (モニタカタログ)メインページには、全く新たなモニタを作成することのできる **Create New Monitor** (新モニタ作成)セクションがあります。基礎となるモニタタイプとして、以下のオプションがあります。

モニタタイプ	説明
TCP Transaction	ターゲットマシンとの間で TCP トランザクションを実行し、オプションのデータ文字列をコネクションに書き込みます。指定した正規表現をレスポンス内で検索することができます。
Ping	指定した間隔でターゲットマシンに対して Png を実行します。
External Program	指定したトラフィックマネージャマシン上のファイルパスにある外部プログラムを実行します。2つの内部引数 port および ipaddr と、その他指定したデータを送信します。終了コード 0 は成功に分類され、0 以外のコードまたはタイムアウトは失敗と見なされます。 <code>ZEUSHOME/zxtm/conf/scripts</code> で、カスタムモニタの実行可能ファイルの例を参照することができます。

モニタタイプ	説明
HTTP	オプションで SSL を使用して、ホストヘッダや使用 URL パスなど指定されたパラメータとともに、HTTP リクエストをターゲットサーバに送信します。レスポンスコード内でステータスコードの正規表現を検索し、正規表現をレスポンスボディデータと照合します。
TCP Connect	ターゲットマシンとの TCP コネクションを確立し、適切なポートが開いていることを確認します。
RTSP	指定された rtsp_path パラメータを使用して、ターゲットサーバに RTSP DESCRIBE リクエストを送信します。レスポンス内でステータスの正規表現を検索し、正規表現をレスポンスボディデータと照合します。
SIP	指定された転送プロトコルを使用して、ターゲットサーバに SIP OPTIONS リクエストを送信します。受信したレスポンスコード内でステータスの正規表現を、またボディデータ内でボディの正規表現を検索します。これらのいずれかが一致しない場合、失敗が報告されます。

モニタの基本設定を選択したら、**Add Monitor** (モニタ追加)をクリックします。モニタの名前をクリックすると、そのモニタを編集できます。

TCP トランザクションまたは **HTTP** モニタテンプレートを使用して、高度なカスタムモニタを作成することができます。例えば、2 つの **HTTP** ベースのモニタを組み合わせ、Web サイトのセキュアな部分をテストし、以下のことを検証できます。

- ユーザ名とパスワードが有効なリクエストに対して、有効な 200 レスポンスを受信すること
- ユーザ名とパスワードが不正なリクエストに対して、401 または 403 エラーレスポンスを受信すること

TCP トランザクションモニタは、他のプロトコルを使用するノードでも構成要素として役立ちます。

より高度なテストを実行するために、モニタで外部プログラムやスクリプトを実行することができます。**外部プログラムモニタ**については、CHAPTER 14 の「**外部プログラムモニタ**」セクションで説明します。

ノード単位のモニタとプール全体のモニタ

モニタは、**ノード単位**と**プール単位**の2つのカテゴリに分類されます。ノード単位のモニタは、プール内の各ノードのヘルスをテストします。プール全体のモニタは、プール全体のヘルスに影響を与える特定のマシンに対してテストを実行します。例えば、メールサーバプールがデータを NFS サーバに保存し、各バックエンドサーバがその NFS サーバにアクセスしている場合、プール全体のモニタではこの NFS サーバをテストできます。このサーバで障害が発生すると、どのバックエンドサーバもデータを読み込むことができないため、プール全体で障害が発生したものと見なされます。

モニタをノード単位にするかプール全体にするかを選択できます。プール全体にする場合、プールが使用する NFS サーバなど、モニタでテストするマシン(および場合によってはポート)を指定する必要があります。

ヘルスモニタの使用

プールへのモニタの適用

モニタをプールに適用するには、**Services** (サービス) ボタンをクリックし、次に **Pools** (プール) タブをクリックします。プールの名前をクリックして編集します。

Health Monitoring (ヘルスモニタリング) をクリックして、プールで使用するモニタを選択します。ドロップダウンからモニタを選択して **Add Monitor** (モニタ追加) をクリックし、新しいモニタを追加することができます。プールに適用されているモニタの設定を変更するには、そのモニタの **Edit In Catalog** (カタログ内編集) リンクをクリックします。

注: テストリクエストを送信するモニタを使用すると、多少ですがプールへの負荷が増加します。DoS 障害を防ぐために、バックエンドサーバがモニタで生成されたトラフィックに対応できることを確認してください。

外部プログラムモニタ

外部プログラムモニタは、任意の言語で作成できます。トラフィックマネージャは、実行可能ファイルに以下のコマンドライン引数を渡します。

```
domonitor --ipaddr=<machine_to_monitor> \  
          --node=<nodename> \  
          --port=<port_to_monitor>
```

注: `ipaddr`、`node`、および `port` 引数は常に外部プログラムモニタに渡されます。`node` 引数はプールに設定されているノードのホスト名部分、`ipaddr` はそれに対応する IP アドレスです。モニタにその他の引数を指定することもできます。

これは、同じ実行可能ファイルで複数のモニタリングタスクを実行する場合に役立つことがあります。該当するモニタの **Edit (編集)** ページの **Program Arguments (プログラム引数)** セクションを使用します。

例えば、2 つの追加の引数(「**regex**」および「**interface**」)を設定し、**Edit(編集)** ページでこれらの引数に適切な値を指定すると、モニタの実行可能ファイルは以下のよう呼び出されます。

```
domonitor --ipaddr=<machine_to_monitor> \  
          --node=<nodename> \  
          --port=<port_to_monitor> \  
          --interface=<interface_value> \  
          --regex=<regex_value>
```

モニタが正常に終了すると、終了コード 0 が返されます。タイムアウトまたは 0 以外の終了コードは、失敗と見なされます。この場合、**STDERR** の内容がイベントログに書き込まれます。

モニタの実行可能ファイルまたはスクリプトを作成してテストする場合、上記のフォーマットで引数を指定してコマンドラインから簡単に実行することができます。モニタにより、デバッグ情報を **STDOUT** に出力することができます。

トラフィックマネージャでモニタを使用する場合、詳細モードを有効化することができます。モニタが詳細モードで呼び出された場合、実行可能ファイルによって **STDOUT** に出力された全ての情報がイベントログに書き込まれます。詳細モードの有効化/無効化は、モニタの **Edit (編集)** ページで切り替えることができます。

トラフィックマネージャへのモニタのアップロード

トラフィックマネージャにモニタをアップロードしたり、トラフィックマネージャでモニタを管理および削除するには、トラフィックマネージャ管理インターフェースの **Catalog (カタログ) > Extra Files (追加ファイル) > Monitor Programs (モニタプログラム)** ページを使用します。

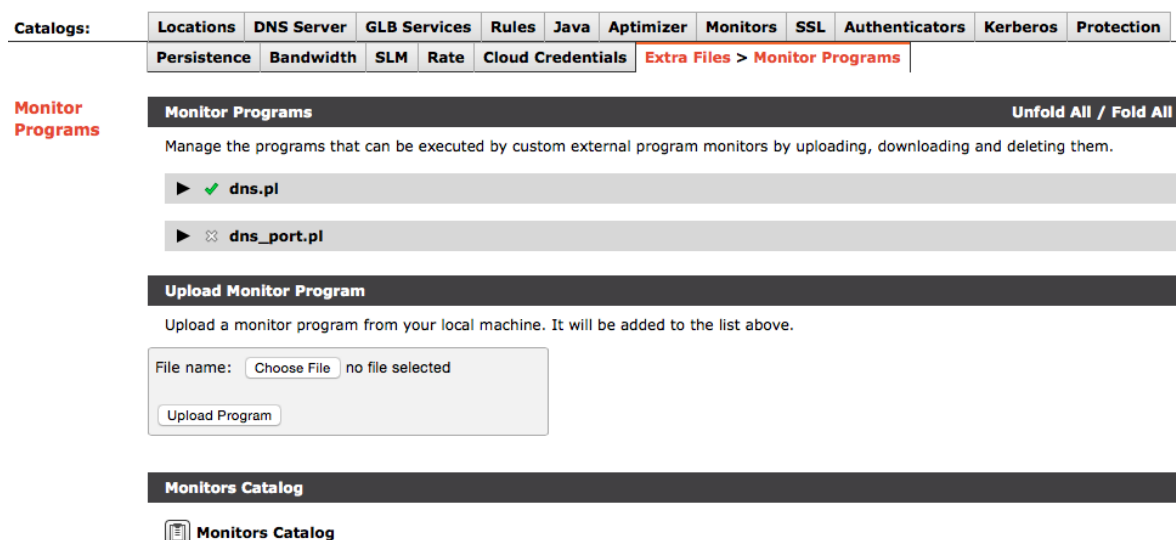


Fig 53. 外部モニタプログラムの管理

Perl でのモニタの作成

Perl でモニタを作成する場合、トラフィックマネージャに含まれているヘルパモジュール `Monitor.pm` を利用できます。このモジュールには、`ParseArguments()` 関数と、成功または失敗で終了する関数およびログを書き出す関数が含まれています。

```
use Zeus::ZXTM::Monitor qw( ParseArguments
                             MonitorWorked
                             MonitorFailed
                             Log );

%ARGS = ParseArguments();

Log( "Running test" );
if( do_test( $ARGS{ipaddr}, $ARGS{port} ) ) {
    MonitorWorked();
} else {
    MonitorFailed( $error_message );
}
```

`Monitor.pm` の詳細を表示するには、以下のコマンドを使用します。

```
$ perldoc ZEUSHOME/zxtm/lib/perl/Zeus/ZXTM/Monitor.pm
```

CHAPTER 15 サービスプロテクション

本章では、インターネットホスティングに伴ういくつかのリスクと、トラフィックマネージャを使用してサービスに対するこれらのリスクを減らす方法について説明します。

リスクの分類

DoS(サービス妨害)

DoS (サービス妨害)攻撃は、サービスの正当な利用を妨げる悪意のある試みを特徴とし、ネットワークへの過剰なデータの送信やマシン間の接続の妨害といった形を取ります。サーバまたはサーバクラスタの使用可能な CPU、帯域、またはストレージを枯渇させるなど、特定のユーザやアプリケーションが通常利用できるリソースを消費することを狙ったものもあります。

DoS 攻撃の根本にあるのは、設計上の弱点や OS の欠陥、CPU 時間、ディスク、メモリなどのリソースへのアクセスが制限されていないサービスの悪用です。例えば、悪意のある攻撃者がデータベース駆動型の Web サイトに対し、完了までに極度に集中的な SQL 処理を必要とするリクエストを作り上げる方法を見つける可能性もあります。このようなリクエストが行われると、そのクエリが実行されている間はデータベースへの正当なアクセスが実質的に拒否されかねません。

Web サイトなど公開されたサービスを通して仕掛けられる DoS 攻撃は、最も一般的なリスクに分類されます。トラフィックマネージャは、このような攻撃の影響を軽減、または全て阻止できるよう設計されています。

Web ワームとウイルス

OS の脆弱性は、弱点から侵入し、自己インストールして増殖する多くのインターネットワームが生まれるきっかけになりました。例えば、ワームには、それ自体をさらに増殖させるために十分なペイロードをインストールできるものもあります。ワームの影響はデスクトップ PC にも及んでいます。このリスク分類の攻撃頻度は増加しつつあり、その多くは、感染したマシンが攻撃の一端を担う計画的な DDoS 攻撃へと発展します。

ワームに加え、デスクトップ PC のユーザは、電子メールやセキュリティの甘い Web サイトから侵入するウイルスのリスクにもさらされています。これらのウイルスにより、PC にリモートコントロール可能なサービスがインストールされる可能性もあります。

DDoS(分散型サービス妨害)攻撃

DDoS 攻撃では、通常はワームやウイルスに感染した、「戦力として集められた」多数のクライアントマシンが利用されます。悪意のある攻撃者は、これらのクライアントをリモートコントロールできるため、特定のサービスプロバイダや企業に対して集中攻撃を行い、従来の DoS 攻撃よりはるかに大規模な影響を与えることが可能です。この問題を完全に解決するためには、他の ISP との協力(外部に送信されるパケットのフィルタなど)が必要になるため、分散型攻撃を減らすことは DoS 攻撃に対処するより大幅に困難です。

トラフィックマネージャは、ネットワーク帯域がデータであふれた状態ではなく、受信するコネクションを管理する十分な処理能力があれば、基本的な DDoS 攻撃および DoS 攻撃を阻止することができます。

不正形式の HTTP 攻撃

適切にファイアウォールで保護された Web サーバであっても、公開サービスをポート 80 でインターネットに提供する必要があります。HTTP のクライアントサーバーアーキテクチャでは、クライアントがポート 80 を通してこれらの Web サーバに自由にデータを送信することができます。言い換えれば、悪意をもってサーバを過負荷状態または正常に機能しない状態へと追い込むようこの情報を作り上げることも可能です。バグを悪用して Web サーバを脆弱にし、コントロールをリモート攻撃者に引き渡すような不正形式のリクエストが作成される状況で攻撃は発生します。

ファイアウォールおよびその他のセキュリティ対策

トラフィックマネージャはファイアウォールではなく、専用のファイアウォールと組み合わせて使用するための製品です。セキュアな運用の詳細については、CHAPTER 23 を参照してください。

保護機能

トラフィックマネージャは、セキュリティ上の弱点を突いた既知の攻撃の影響を受けない堅牢なソリューションです。例えば、既知のあらゆるワームおよび不正形式の HTTP 攻撃に屈することはありません。また、トラフィックマネージャはクラスタに送信されるトラフィックの入口に配置され、ルーティングをインテリジェントに決定することができるため、プラットフォームを明らかなネットワーク攻撃から防御したり、悪意のあるコンテンツを透過的にフィルタすることが可能です。これは、バックエンドサーバがインターネット IP アドレスを持たず、プライベートネットワークを使用している場合に特に有効です。

トラフィックマネージャは、各仮想サーバに個別に設定可能なサービスプロテクション機能も備えています。これらの機能では、様々な方法で動的にリクエストを管理し、フィルタすることができます。

ネットワークアクセス制限

クライアントの送信元 IP アドレスを基に、リクエストを受け入れるかどうかを決定することができます。特定の IP アドレスまたはネットワークブロックで悪意のあるリクエストが生成されている場合、これらのアドレスからの接続を全て効率的に破棄するようトラフィックマネージャを設定することで、バックエンドプールを攻撃から保護することができます。

コネクション制限

システムを乱用しようとするクライアントが異常に多くのコネクションを生成することがよくあります。トラフィックマネージャは、アクセス頻度の高い上位 10 の IP アドレスや、各 IP アドレスから行われた 1 分あたりの接続数に基づいて、過剰なアクティビティを検出してフィルタすることができます。これにより、トラフィックマネージャは、システムに対して過度に集中的なリクエストを行うクライアントのアクセスを拒否することができます。

不正形式の HTTP のフィルタ

トラフィックマネージャは、特定の不正形式の HTTP リクエストを検出して拒否したり、規格に準拠したリクエストを送信するようクライアントにリクエストしたりすることができます。例えば、リクエストからバイナリデータをフィルタし、非常に大きいヘッダが DoS 攻撃の誘導手段として使用されないようにすることができます。

ルールベースの保護

トラフィックマネージャのサービスプロテクションシステムは、受信する全てのリクエストをチェックし、指定条件に一致するものを全て破棄するよう設定できます。この機能を利用すると、Web サイトで実行されているサードパーティアプリケーションの既知の脆弱性からシステムを保護することができます。例えば、買い物かごなどの一部のビジネスクリティカルなアプリケーションには、不正形式のパラメータを含む URL のリクエストを契機として生じる脆弱性が存在することが分かっています。

一般に、これらのリクエストの形式は、セキュリティメーリングリストで公開されています。トラフィックマネージャのサービスプロテクションシステムでは、この情報を使用して、不正形式のパラメータを含む全てのリクエストをフィルタして除外できるため、全ての Web サイトにわたってこのような脆弱性に素早く簡単に対処することができます。

受信する全てのリクエストをフィルタするためのルールのリストを作成することにより、リクエストをブロックするようトラフィックマネージャを設定することができます。また、リクエストのサイズやリクエストに含めることのできる情報のフォーマットを制限することも可能です。

サービスプロテクションの有効化

サービスプロテクションの設定は2段階で行います。まず、**カタログ**にサービスプロテクションクラスを作成します。サービスプロテクションクラスは必要な数だけ設定でき、それぞれに独自の設定およびルールを指定できます。

次に、サービスプロテクションクラスを1台または複数の仮想サーバに割り当てます。これにより、保護の設定がその仮想サーバのリクエストに関連付けられます。同じサービスプロテクションクラスを複数の仮想サーバに割り当てることも可能です。

サービスプロテクションクラスの追加

サービスプロテクションクラスを追加するには、まず **Catalogs** (カタログ) ボタンをクリックします。 **Service Protection catalog** (サービスプロテクションカタログ) を選択します。

トラフィックマネージャに、既存のサービスプロテクションクラスのリストが表示されます。名前を入力して **Create Class** (クラス作成) をクリックすることにより、このリストに新しいサービスプロテクションクラスを追加できます。

新しいサービスプロテクションクラスを追加するときに、5つの主要設定グループで設定を行います。

基本設定

これらの設定では、サービスプロテクションの有効化/無効化を切り替えたり、どのモードでサービスプロテクションクラスを機能させるかを決定したりします。

このサービスプロテクションクラスが一時的に適用されないようにするには、そのクラスのサービスプロテクションの有効化/無効化を切り替えます。これは、特にこのサービスプロテクションクラスを使用するよう多数の仮想サーバが設定されている場合に便利です。

ログバッファ時間、すなわちディスクへの書き込みを減らすために(また DoS 攻撃中に大量のログが生じる場合の波及的な影響を抑えるために)トラフィックマネージャがロギングをバッファする時間を設定することができます。トラフィックマネージャには、詳細な出力を記録するデバッグオプションもあります。

重要: デバッグオプションを使用すると、トラフィックマネージャにかかる負荷が大きくなる可能性があるため、商用システムでは使用しないでください。

トラフィックマネージャにはテストモードもあります。このモードでは、このサービスプロテクションクラスに従ってトラフィックマネージャが行った決定が記録されますが、それに対するアクションは実行されません。このオプションは、サービ

スプロテクションクラスのパラメータをサービスに適した値に調整することを目的としています。商用サービスへの移行前のテスト期間に使用してください。

同時接続

これらの設定では、トラフィックマネージャが個々の IP アドレスに許容する接続数の制限を指定します。

- **max_1_connections:** トラフィックマネージャが各接続 IP アドレスに許容する最大同時接続数です。この制限を無効化するには、0 に設定します。
- **per_process_connection_count:** トラフィックマネージャがプロセス単位の同時接続カウントモデルを使用するかどうかを決定します(通常、トラフィックマネージャには、有効な CPU コアに 1 プロセスずつ、複数のプロセスがあります)。

「Yes (はい)」をクリックすると、その接続 IP アドレスはトラフィックマネージャ内の各プロセスに最大 **max_1_connections** のコネクションを行えるようになります。

「No (いいえ)」をクリックすると、その接続 IP アドレスはトラフィックマネージャ全体で最大 **max_1_connections** のコネクションを行えるようになります。

per_process_connection_count が「Yes (はい)」に設定されている場合、以下の制限が追加で適用されます。

- **max_10_connections:** アクセス頻度の高い上位 10 を合わせた接続 IP アドレスからの最大同時接続数です。通常、トラフィックマネージャに対して何らかの DoS 攻撃が進行中か、その他の不正なコネクションが行われている場合、特定仮想サーバの負荷のほとんどは上位 10 のクライアントによって占められます。そのため、この特定グループのユーザを制限しながら、その他のリクエストは通常どおり処理するのが適切です。通常の利用においては、**max_1_connections** 制限の 1~10 倍の値をお奨めします。この制限の効果をなくすには、**max_1_connections** 制限の 10 倍以上の値に設定してください。**per_process_connection_count** が「No (いいえ)」に設定されている場合、または **max_1_connections** が 0、または **min_connections** が 0 の場合、トラフィックマネージャはこの設定を無視します。
- **min_connections:** **max_10_connections** 制限に対するエントリのしきい値です。この値以下の同時接続を行う接続 IP アドレスには、**max_10_connections** 制限は適用されません。**max_1_connections** および **max_10_connections** の両方の制限を無効化するには、0 に設定します。**per_process_connection_count** が「No (いいえ)」に設定されている場合、トラフィックマネージャはこの設定を無視します。

接続レート

重要: 接続レートの制限はプロセス単位で行われ、そのレートは、プロセスが処理しているコネクションに対して独立して適用されます。

これらの設定では、各接続 IP アドレスが新たなコネクションまたはリクエストを行うレートに対して制限をかけることができます。

- **max_connection_rate:** 個々の IP アドレスが、**rate_timer** で指定された間隔の間にトラフィックマネージャと行うことができる最大接続数です。制限を無効化するには、0 に設定します。

注: HTTP の場合、単一の **keepalive** コネクション内でトンネルされている場合でも、各リクエストが 1 つのコネクションとしてカウントされます。

- **rate_timer:** トラフィックマネージャが、**max_connection_rate** が評価される頻度を判定するために使用する間隔(秒単位)です。例えば、この値を 60 に設定すると、毎分 **max_connection_rate** コネクションの制限が課されます。この制限を上回って試行されたコネクションは、**rate_timer** 間隔の残りの間、全て拒否されます。

CHAPTER 17 の「リクエストレートシェイピング」で説明するリクエストレートシェイピング機能を利用すると、接続レートをさらにきめ細かくコントロールすることができます。

アクセス制限

Allowed (許可)リストでは、許可する IP アドレスまたはネットワークを指定できます。これらの IP アドレスまたはネットワークは、このサービスプロテクションクラスでの通常のサービスプロテクションチェックが免除されます。つまり、このリストに含まれる IP アドレスは、同時接続や接続レートセクションで設定されている制限の対象になりません。

Banned (禁止)リストは、トラフィックマネージャに対して常にコネクションを破棄させる IP アドレスまたはネットワークを宣言するために使用します。

送信元 IP をチェックするとき、この両方のリストに含まれるサブネットがロングストマッチ優先のアルゴリズムに従って評価されます。すなわち、より具体的なサブネットが優先されます。そのため、許可された具体性の低いサブネットに含まれる具体性の高いサブネットを禁止することも、その逆のこともできます。例えば、192.168.0.0/16 を許可し、192.168.128.0/24 を禁止すると、192.168.0.137 の IP からのリクエストは許可しますが、192.168.128.137 の IP からのリクエストは許可しません。

Allowed (許可)および **Banned** (禁止)の両方のリストに同じサブネットが存在する場合、そのサブネットからのリクエストは許可されます。

いずれかのリストに追加するには、適切なボックスに IP アドレスまたはネットワークアドレスを入力します。以下のいずれかのフォーマットを使用します。

- 192.0.2.0/255.255.255.0
- 192.0.2.0/24
- 192.0.2.1
- 192.0.2.

HTTP 固有の設定

これらの設定では、HTTP トラフィックに対して実行できる追加のチェックを指定します。

トラフィックマネージャでは、一般に認められているガイドラインに従って URL の形式が正しいかどうかをチェックすることにより、RFC2396 への適合を確保できます。標準的な Web ブラウザは RFC2396 に適合しているため、このテストでは、異常なデータや悪意のあるデータまたはフォーマットを含むリクエストは拒否され、正当なリクエストは通過することができます。

HTTP ボディと HTTP ヘッダの長さを制限することができます。これにより、悪意のある攻撃者からボディやヘッダが大量または際限なく送信されるのを防ぎます。通常、Web サーバはリクエストを受信しながらメモリにバッファするため、メモリ枯渇攻撃に対して脆弱になる可能性があります。クライアントが大量のリクエストデータを送信できなくすることで、このような攻撃から保護できます。

全ての HTTP ヘッダの合計サイズ制限を指定することもできます。これにより、攻撃者が多数の HTTP ヘッダを使用して非常に大きいリクエストを作成するのを阻止できます。

バッファオーバーフロー攻撃や大量のデータやワームの誘導手段として URL を使用するその他の攻撃を防ぐために、リクエストされる URL のサイズを制限できます。

復号後、トラフィックマネージャはバイナリデータを含む HTTP ヘッダを拒否できます。これは、一般にバッファオーバーラン攻撃が試行されたことを示しますが、一部の低品質のアプリケーションがバイナリデータを使用する場合もあります。ご使用のアプリケーションが影響を受ける可能性が疑われる場合は、まずテストモードでこの機能を使用して確認してください。

最後に、トラフィックマネージャでは、上記のいずれかのテストで不合格になったリクエストをそのまま拒否するか、リクエストを行ったクライアントに HTTP エラーメッセージを送信するかを指定できます。エラーメッセージを送信すると、オーバーヘッドがわずかに増加し、DoS 攻撃が進行中の場合は非常に大きいオーバーヘッドが生じる可能性があります。

注: HTTP トラフィックを処理しない仮想サーバにこのサービスプロテクションクラスを関連付けた場合、これらの設定は無視されます。

サービスプロテクションルール

このセクションには、サービスプロテクションクラスが使用している現在のルールが表示されます。ルールを変更するには、ドロップダウンメニューから使用するルールを選択します。TrafficScript の詳細については、CHAPTER 8 を参照してください。

設定内容を適用してサービスプロテクションクラスを作成するには、**Update** (更新) をクリックします。

仮想サーバへのサービスプロテクションクラスの適用

サービスプロテクションクラスを有効にするには、まず仮想サーバに割り当てる必要があります。これを行うには、**Services** (サービス) ボタンをクリックします。次に **Virtual Servers** (仮想サーバ) タブをクリックし、既存の仮想サーバを選択します。

Service Protection (サービスプロテクション) を選択します。このユーザインターフェースには、カタログ内の使用可能なサービスプロテクションクラスのドロップダウンボックスがあります。

サービスプロテクションクラスを選択したら、**Update** (更新) をクリックします。

トラフィックマネージャにより、新しいサービスプロテクションクラスが即座に全てのトラフィックマネージャに適用されます。サービスプロテクションクラスがテストモードかデバッグモードかを示す概要テーブルが表示されます。サービスプロテクションクラスが有効化されていない場合は(CHAPTER 15 の「基本設定」セクションを参照)、そのことも示されます。

サービスプロテクション性能

トラフィックマネージャのサービスプロテクションシステムでは、リクエストごとに追加のメモリと CPU 時間が必要になります。多数のリクエストを管理し、サービスプロテクションクラスで高度なルールを使用する場合は、このサービスプロテクションクラスを使用する仮想サーバに対する全てのリクエストについて、処理に要する時間がわずかに長くなります。大規模なシステムでは、処理能力を強化するために、フロントエンドネットワークにトラフィックマネージャを追加することが必要になる場合があります。

CHAPTER 16 帯域制御

本章では、帯域制御の概要と、ネットワークで帯域をどのように使用するかをコントロールするようトラフィックマネージャを設定する方法について説明します。

注: トラフィックマネージャ製品によっては、帯域制御を利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

帯域制御とは

帯域制御により、トラフィックマネージャは、外部から受信するトラフィックまたは外部へと送信されるトラフィックで使用される 1 秒あたりのバイト数を仮想サーバまたはリクエストのタイプごとに制限することができます。

通常、ネットワーク帯域は、全てのコネクションに対して最高の転送速度で提供されます。そのため、二次的なサービスで多くの帯域が使用され、最もクリティカルなサービスが犠牲になるなど、ネットワークの使用が均等でなくなることがあります。トラフィックマネージャでは、帯域制御を使用して、この不均衡を明示的にコントロールすることができます。

例えば、20Mbps/s のネットワークコネクションがあり、このコネクションが FTP ダウンロードに過剰に使用されているために、主要な HTTP サービスのレスポンス速度に悪影響が及んでいるとします。この場合、FTP 仮想サーバの帯域を 2Mbps/s に制限できます。

帯域制限は、クラスタ内の全てのトラフィックマネージャマシンで自動的に共有され、適用されます。個々のトラフィックマネージャは、それぞれの負荷に応じて、合計制限から異なる比率の帯域を受け取ります。未使用の帯域は、各トラフィックマネージャのニーズに応じて、クラスタ全体に均等に割り当てられます。

注: 帯域制御は、トラフィックマネージャから送信されるトラフィックにのみ適用されます。すなわち、トラフィックマネージャは、リクエストをサーバノードに書き込むときやレスポンスをクライアントに書き込むときに使用される帯域をコントロールできますが、クライアントがリクエストをトラフィックマネージャに書き込む速度やサーバがレスポンスをトラフィックマネージャに書き込む速度をコントロールすることはできません。

帯域制御は、外部アドレスに送信されるか、トラフィックマネージャの内部のみで転送されるかにかかわらず、全てのデータに適用されます。内部トラフィックには、例えば、ループバックインターフェースを使用してデータがプールから仮想サーバに戻される場合が含まれます。

帯域制御の設定

トラフィックマネージャの帯域制御機能は、カタログを通して提供されます。すなわち、それぞれ制限の異なる複数の帯域制御クラスを追加することができます。また、TrafficScript ルールによって帯域制御クラスを選択できるため、同じ仮想サーバに対するリクエストを区別すると有益な状況では、さらに柔軟性が高まります。例えば、トラフィックマネージャの帯域制御システムを使用して、CGI リクエストに対して、画像リクエストとは異なる制限を適用することも可能です(以下の例を参照)。

複数のトラフィックマネージャから成るクラスタがある場合、帯域制御システムではこの構成が考慮されます。サーバを増設した場合も調整が自動的に行われるため、ユーザが調整を行う必要はありません。

カタログへの帯域クラスの追加

帯域制御クラスをカタログに追加するには、**Catalogs** (カタログ)をクリックし、次にメニューバーの **Bandwidth** (帯域)をクリックします。

トラフィックマネージャに既存の帯域制御クラスのリストが示され、新しい帯域制御クラスを作成するためのボックスが表示されます。

このボックスに名前を入力し、**Create Class** (クラス作成)をクリックします。この新しいクラスの設定ページが表示されます。最大帯域および制限共有オプションを選択できます。

最大帯域制限を選択するには、以下のいずれかの操作を行います。

- ドロップダウンリストからあらかじめ定義されている値を選択する
- ドロップダウンリストから **Custom** (カスタム)を選択し、表示されるボックスに値を kbits/秒単位で入力する

帯域クラスにより帯域をコネクションに分配する方法として、以下の3つのモードがあります。

- **各コネクションを最大転送速度に制限** - 帯域の共有は適用されず、全てのトラフィックマネージャにおいて、このクラスが適用された全てのユーザが、設定されている最大帯域制限に固定されます。これは、個々のクライアントにビデオ/オーディオストリームを配信する場合に便利です。
- **トラフィックマネージャ単位で帯域を共有** - 各トラフィックマネージャにおいて、この帯域クラスが適用された全てのユーザで制限が共有されます。1Mbps の制限を適用し、3 台のトラフィックマネージャがある場合、各トラフィックマネージャが使用できる帯域は最大 1Mbps で、合計帯域は 3Mbps になります。

- 全てのトラフィックマネージャで帯域を共有 - 全てのトラフィックマネージャにわたり、この帯域クラスが適用されている全てのユーザで制限が共有されます。1Mbps の制限を適用し、3 台のトラフィックマネージャがある場合、クラス全体で利用できる帯域は合計 1Mbps のみです。この制限は動的に割り当てられます。各トラフィックマネージャでは、他のトラフィックマネージャがアイドル状態のときは、1Mbps 全てを使用できます。

Update (更新)をクリックして変更内容を適用します。

注: ストレージまたはメモリ容量とは異なり、ネットワークスループットは慣習的に (1024 ではなく)1000 の倍数で測定されます。

仮想サーバへの帯域クラスの割り当て

仮想サーバに割り当てられた帯域制御クラスにより、その仮想サーバがレスポンスデータをクライアントに送信するときに利用できる帯域が制限されます。

設定

帯域制御クラスを仮想サーバに割り当てるには、**Services** (サービス)をクリックし、次に **Virtual Servers** (仮想サーバ)をクリックします。使用可能なサーバのリストから仮想サーバを選択し、**Edit** (編集)ボタンをクリックします。

Assigned Classes (割り当てクラス)の横にある **Edit** (編集)をクリックします。次に、**Bandwidth Management** (帯域制御)をクリックし、いずれかの帯域制御クラスをラジオボタンで選択します。

ページの下部にある **Update** (更新)ボタンをクリックして変更内容を適用します。仮想サーバで即座にこの帯域制御クラスの使用が開始されます。

注: 帯域制御クラスがない場合は、**none(なし)**オプションのみが表示されます。帯域制御クラスを追加するには、**Create New Bandwidth Management Class** (新しい帯域制御クラス作成)リンクをクリックします。

プールへの帯域クラスの割り当て

プールに割り当てられた帯域制御クラスにより、そのプールがリクエストデータをノードに送信するときに利用できる帯域が制限されます。

設定

帯域制御クラスをプールに割り当てるには、**Services** (サービス)をクリックし、次に **Pools** (プール)をクリックします。使用可能なサーバのリストからプールを選択し、**Edit** (編集)ボタンをクリックします。

Bandwidth Management (帯域制御)セクションで、**Edit** (編集)をクリックします。いずれかの帯域制御クラスをラジオボタンで選択します。

ページの下部にある **Update** (更新) ボタンをクリックして変更内容を適用します。ルールで即座にこの帯域制御クラスの使用が開始されます。

注: 帯域制御クラスがない場合は、*none(なし)* オプションのみが表示されます。帯域制御クラスを追加するには、**Create New Bandwidth Management Class** (新しい帯域制御クラス作成) リンクをクリックします。

TrafficScript を使用した帯域クラスの選択

帯域制御クラスは、TrafficScript 関数の `request.setBandwidthClass()` および `response.setBandwidthClass()` のいずれかを使用してリクエストごとに選択することもできます。これらの関数は、それぞれ「ノード方向」の帯域と「クライアント方向」の帯域に影響を与えます。

帯域制御クラスは、TrafficScript ルールの任意の時点で適用でき、これによって仮想サーバ自体の設定がオーバーライドされます(設定がある場合)。

例えば、以下の TrafficScript レスポンスルールでは、大きいレスポンスまたはサイトの「/downloads/」部分からのレスポンスに対して「downloads」帯域制御クラスが選択されます。

```
# This must be used as a response rule, because we
# want to determine the response content length...
$url = http.getPath();
$clen = http.getResponseHeader( "Content-Length" );

if( string.startsWith( $url, "/downloads" ) ||
    $clen >= 100*1024 ) {
    response.setBandwidthClass( "downloads" );
}
```

CHAPTER 17 リクエストレートシェイピング

本章では、リクエストレートシェイピングの概要と、アプリケーションに対するリクエストをレート制限するようトラフィックマネージャを設定する方法について説明します。

注: トラフィックマネージャ製品によっては、リクエストレートシェイピングを利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

リクエストレートシェイピングとは

特定のユーザによってサービスが独占され、他のユーザによるそのサービスの利用が妨害されることがあります。バックエンドアプリケーションインフラストラクチャの拡張性が限られている場合もあり、アプリケーションに送信されるリクエストが多すぎると、容易に処理が追いつかなくなります。より広範なセキュリティポリシーの一環として、電子メールの送信やサービスへのログインなどの特定のアクティビティの許容レートを制限することが望ましい場合があります。

リクエストレートシェイピングでは、イベントの識別方法を非常にきめ細かくコントロールしながら、幅広いイベントに対して制限を指定することができます。これらのイベントに1秒あたりのレートおよび1分あたりのレートを適用できます。

以下に例を挙げます。

- 個々の Web スパイダをレートシェイピングして、Web サイトの処理が追いつかなくなる状況を阻止できます。それぞれ異なるリモート IP アドレスからアクセスする各 Web スパイダに最大リクエストレートを適用することができます。
- 個々の SMTP コネクションまたは同じクライアントによるコネクショングループを抑制し、各コネクションの1分あたりの最大送信電子メール数を制限することができます。
- 新しい SMTP コネクションをレートシェイピングし、リモートクライアントが特定レートでのみ新しいコネクションを確立できるようにすることもできます。
- 特定のアプリケーションに転送される1分あたりの接続数にグローバルレートシェイピングを適用できます。
- 個々のユーザによるサービスへのログイン試行を識別し、各ユーザの1分あたりの試行回数を制限することにより、ログインの辞書攻撃を阻止することができます。

TrafficScript 関数の `rate.use()` を使用してリクエストレート制限を適用すると、以下のように処理が行われます。

- 仮想サーバが受信トラフィックを受け入れます。
- 仮想サーバによりリクエストルールが実行されます。
- リクエストルールにより、TrafficScript 関数の `rate.use()` を使用してレートシェイピングが適用されます。

注: 一般に、リクエストレート制限は、リクエストの処理レートをシェイピングするために TrafficScript リクエストルールで使われます。必要に応じて、レスポンスルールで使用することも可能です。この場合、その他のイベントを制限できますが、通常は必要ありません。

リクエストレートシェイピングクラス(レートクラス)の設定

リクエストレートシェイピングを使用するには、少なくとも 1 つのリクエストレートシェイピングクラス(レートクラスとも言う)を追加する必要があります。各レートクラスでは、1 秒または 1 分あたりのイベントのレート制限を定義します。

レートクラスはカタログに保存され、TrafficScript 関数の `rate.use()` を使用して呼び出します。

カタログへのレートクラスの追加

新しいレートクラスを追加するには、**Catalogs** (カタログ)をクリックし、次にメニューバーの **Rate** (レート)をクリックします。**Create new Rate class** (新しいレートクラスの作成)ボックスに新しいクラスの名前を入力し、**Create Class** (クラス作成)をクリックします。

以下の 2 つの設定を行えます。

設定	説明
<code>max_rate_per_minute</code>	1 分あたりに許容されるリクエストの最大数です。「0」は「制限なし」を意味します。
<code>max_rate_per_second</code>	1 秒あたりに許容されるリクエストの最大数です。「0」は制限なしを意味します。

必要に応じて、1 秒あたりの制限と 1 分あたりの制限を両方設定することができます。この場合、両方の制限が適用されます(1 分あたりの制限が 1 秒あたりの制限の 60 倍を超えている場合、1 分あたりの制限に効果がないことに注意)。

レートクラスを設定したら、**Update** (更新)をクリックして変更内容を適用します。

レートクラスの使用

レートクラスはキューとして機能します。TrafficScript 関数の `rate.use()` が呼び出されると、コネクションが一時中断し、レートクラスが管理するキューに追加されます。その後、1 秒および 1 分あたりの制限に従って、コネクションがキューから解放されます。

「shape requests」というレートクラスを作成し、1 秒あたりのイベント数(リクエスト数)を 10 個に制限するよう設定したとします。

以下の TrafficScript リクエストルールでは、新しいリクエストがシステムに着信する頻度に関わらず、1 秒あたり処理されるリクエストは 10 個のみです。

```
rate.use( "shape requests" );
```

リクエストが着信すると、リクエストルールが実行されます。1 秒あたりに着信するリクエストが 10 個未満の場合、`rate.use()` 関数は即座に戻り、リクエストは遅延なく処理されます。

1 秒あたりに着信するリクエストが 10 個を超えると、`rate.use()` 関数によって一部のリクエストがキューイングされます。TrafficScript からは、この関数がブロックされているように見えます。その後、リクエストがキューから抜けた時点で、`rate.use()` 関数が戻ります。

レートキュー

キューイングされるコネクションのバックログのサイズに制限はありません。例えば、1000 個のリクエストが短時間に連続して着信すると、そのうち 990 個のリクエストは即座にキューイングされます。その後、1 秒あたり 10 個のリクエストがキューの前方から解放されていきます。

キューイングされている間に、コネクションがタイムアウトになったり、リモートクライアントにより切断される場合があります。この場合、コネクションは即座に破棄されます。

`rate.getBacklog()` 関数を使用して、現在キューイングされているリクエスト数を調べることができます。バックログが大きすぎる場合、コネクションがタイムアウトになるまで待たずにユーザにエラーページを返すことができます。

```
if( rate.getBacklog( "shape requests" ) > 100 ) {  
    http.redirect( "http://some.other.site/" );  
} else {  
    rate.use( "shape requests" );  
}
```



```
}
```

このルールにより、このレートクラスによってキューイングされるリクエストが100個を超えることがなくなります。

キューイングの回避

TrafficScript 関数の `rate.use.noqueue()` を使用しても、`rate.use()` と同様にリクエストをレートシェイピングできますが、この場合、リクエストがキューイングされないという大きな違いがあります。

- レート制限を超えていない場合、`rate.use.noQueue()` は即座に値 1 を返し、接続レートのカウンタが増加します。これは `rate.use()` と同じです。
- レート制限を超えている場合、`rate.use.noQueue()` は値 0 を返します(これに対して `rate.use()` はリクエストをキューイングします)。この場合、TrafficScript を使用して、503 Too Busy エラーを返すなどリクエストに対する処理を決定することができます。

選択的レートシェイピング

特定タイプのリクエストのみをレート制限したい場合があります。例えば、Web サイトで Java Extension ページとその他のコンテンツ(画像、静的コンテンツなど)が混在している場合、以下のように Java Extension ページのみをレートシェイピングすることができます。

```
$url = http.getPath();
if( string.endsWith( $url, ".jsp" ) ) {
    rate.use( "shape requests" );
}
```

さらにきめ細かいレートシェイピング

これまでに説明した例は、全てグローバル制限を適用するものです。全てのリクエストが同じレートキューに入れられます。

多くの場合、さらにきめ細かいレートシェイピング制限を適用することが必要になります。例えば、ログインページについて、各ユーザがログイン試行できる頻度を1分あたり2回に制限するとします。

グローバル制限では、ログインページへのアクセス回数を合計で1分あたり2回に制限することになりますが、代わりにユーザごとの制限を適用して、各ユーザのログイン回数を1分あたり2回に制限します。

ユーザごとに固有のレートシェイピングクラスを作成し、TrafficScript を使用して適切なクラスを選択することもできますが、これよりはるかに良い方法があります。

`rate.use()` 関数は、レートクラスの新しいインスタンスを識別するオプションの「キー」を使用できます。このキーを使用して、同じ制限を共有する複数の独立したレートクラスを作成し、これらのクラスをキーごとに個別に適用することができます。

```
$url = http.getPath();
if( string.endsWith( $url, "login.cgi" ) ) {
    $user = http.getFormParam( "username" );
    rate.use( "login limit", $user );
}
```

レートシェイピング制限は、`$user` の各値に個別に適用されます。システムに新たにアクセスした各ユーザは、「login limit」レートシェイピングクラスを共有するその他全てのユーザとは関係なく、1 分あたりのリクエストが 2 個に制限されます。

レートシェイピング Web スパイダ

低品質の Web スパイダから Web サイトに大量のリクエストが送信され、過剰な帯域が消費されて、正当なサイトユーザが受けるサービスレベルが低下する場合があります。

以下の TrafficScript ルールは、「spiders」というレートクラスを使用して各スパイダのリクエストを 1 秒あたり 1 個に制限します。

User-Agent ヘッダに「Mozilla/」または「Opera/」のプレフィックスのないユーザをスパイダと見なします。

ユーザエージェントおよび送信元 IP アドレスでスパイダを識別し、各スパイダインスタンスに制限を適用します。

```
$ua = http.getHeader( "User-Agent" );

if( ! string.startsWith( $ua, "Mozilla/" ) &&
    ! string.startsWith( $ua, "Opera/" ) ) {

    $ip = connection.getRemoteAddress();
    rate.use( "spiders", $ua.$ip );

}
```

さらに高度なルールでは、スパイダを動的に検出するために、5秒間に渡って1秒あたりのリクエストが10個を超えるなどレートを超えたことのあるユーザのみに対して、1秒あたりのリクエストを1個に制限することもできます。このルールは以下のように機能します。

TrafficScript 関数の `data.set()` を使用して、既知の違反者(ユーザエージェントとIPアドレス)のリストを維持します。このリストは、最初は空の状態です。

リクエストの受信時に、既知の違反者のリストを確認し、リクエストを抑制する場合は1秒あたり1個のリクエストのレートを使用します。

それ以外の場合は、1秒あたり10個のリクエストのレートを使用しますが、レート制限の適用前にバックログを確認します。バックログが大きすぎる場合(50を超える場合など)、このユーザが送信しているリクエストが多すぎるため、違反者のリストに挿入します。

リクエストレートシェイピングのグラフ化

アクティビティモニタを使用して、リクエストレートシェイピングクラスのアクティビティのチャートを作成したり、SNMPを通して値をモニタリングしたりできます。

この方法でグラフ化やモニタリングを行えるのは、グローバルレート制限のみです。特定のキーに適用されたレート制限をグラフ化したりモニタリングしたりすることはできません。

CHAPTER 18 サービスレベルモニタリング

本章では、サービスレベルモニタリングの概要と、ネットワーク上のレスポンス時間をモニタリングするようトラフィックマネージャを設定する方法について説明します。

注: トラフィックマネージャ製品によっては、サービスレベルモニタリングを利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

サービスレベルモニタリングについて

サービスのレスポンス時間は、Web サービスの重要なメトリクスであり、レスポンス時間が短いほど、反応のよい使いやすいサービスであるという印象を与えます。サービスレベルモニタリングでは、トラフィックマネージャがリクエストを受信してからレスポンスデータの最初のバイトをクライアントに送信するまでにかかった時間(ミリ秒単位)をレスポンス時間として使用します。これには、コンテンツがインターネットを経由してクライアントまで返される時間は含まれず、一般にはリモートネットワークの速度の影響を受けません。

レスポンス時間には、トラフィックマネージャでの内部処理(TrafficScript ルールや Java Extensions の実行など)にかかった時間、バックエンドノードへのリクエストの送信にかかった時間、ノードがレスポンスを生成して最初のデータを返すまでにかかった時間、レスポンスルールの実行にかかった時間が含まれます。一般には、クライアントコネクションの速度や待ち時間とは無関係のため、内部システムの性能を表す正確な指標となり、アプリケーション管理者は、これを基に調整や最適化を行うことができます。

注: ノードがレスポンスを返す前に、クライアントから HTTP POST ボディデータなどの追加データを読み取る必要がある場合、このデータの読み取り時間もレスポンス時間の測定対象になります。

サービスレベルモニタリングを使用すると、トラフィックマネージャは、実際のレスポンス時間と適合値を比較して、ノードのレスポンス時間の変化を測定し、それに対処することができます。また、レスポンス時間やその他の関連情報のグラフを取得したり、レスポンス時間が適合値を下回った場合にアラートを報告するかログに記録することができます。トラフィックマネージャでは、TrafficScript ルールを使用して、レスポンス時間に基づいて利用可能なリソースを動的に調整することもできます。

サービスレベルモニタリングクラス(SLM クラス)の設定

サービスレベルモニタリングを使用するには、少なくとも 1 つのサービスレベルモニタリングクラス(SLM クラスとも言う)を追加する必要があります。各 SLM クラスでは、目標とするレスポンス時間とアラートおよびロギングのしきい値を定義します。

SLM クラスはカタログに保存され、各クラスは 1 台以上の仮想サーバに適用することができます。

カタログへの SLM クラスの追加

新しい SLM クラスを追加するには、**Catalogs** (カタログ)をクリックし、次にメニューバーの **SLM** をクリックします。**Create new service level monitoring class** (新しいサービスレベルモニタリングクラスの作成)ボックスに新しいクラスの名前を入力し、**Create Class** (クラス作成)をクリックします。

以下の 3 つの設定を行えます。

設定	説明
response_time	この時間制限内に着信したレスポンスは、適合していると見なされます。この時間より長くかかったレスポンスは、不適合と見なされます(時間はミリ秒単位)。
serious_threshold	適合レスポンスのパーセンテージがこのレベルを下回ると、SERIOUS (重大)レベルのイベントが有効になり、SERIOUS (重大)レベルのメッセージがイベントログに書き込まれます。
warning_threshold	適合レスポンスのパーセンテージがこのレベルを下回ると、WARNING (警告)レベルのイベントが有効になり、WARNING (警告)レベルのメッセージがイベントログに記録されます。

例えば、レスポンス時間を 1000ms、アラートしきい値を 40%、ロギングしきい値を 70%に設定したとします。この場合、以下ようになります。

- サーバから戻る時間が 1000ms 未満のリクエストが 70%未満の場合、Warning (警告)レベルのエントリがトラフィックマネージャログに書き込まれます。
- 所要時間が 1000ms 未満のリクエストが 40%未満の場合、Serious (重大)レベルのエントリがトラフィックマネージャのイベントログに書き込まれます。

適合リクエストのパーセンテージは、過去 10 秒間のリクエストについて、その SLM クラスでモニタリングしているリクエストのうち **response_time** しきい値を満たしたリクエスト数とリクエストの合計数を比較した平均値から求められます。

SLM クラスにより Warning (警告)または Serious (重大)レベルのイベントが有効になったときにアクションを呼び出すイベントハンドラを定義することができます。

SLM クラスを設定したら、**Update** (更新)をクリックして変更内容を適用します。

仮想サーバへの SLM クラスの適用

SLM クラスを仮想サーバに適用するには、**Services** (サービス)をクリックし、次に該当する仮想サーバの横にある **Edit** (編集)ボタンをクリックします。

Assigned Classes (割り当てクラス)までスクロールし、**Edit** (編集)をクリックします。次に、**Service Level Monitoring** (サービスレベルモニタリング)をクリックし、適用する SLM クラスを選択します。

Update (更新)をクリックして変更内容を適用します。

SLM クラスを仮想サーバに関連付けると、そのクラスが全てのリクエストに対して使用されます。また、次のセクションで説明するように、TrafficScript を使用してリクエストの性質に応じて SLM クラスを変更することも可能です。

TrafficScript による SLM クラスの適用

TrafficScript には、SLM クラスから最新情報を読み取ったり、コネクションの SLM クラスを変更したりするための関数が多数用意されています。

これらの関数については、『**TrafficScript Reference**』(オンラインヘルプページから利用可能)に記載されています。

特に、`connection.setServiceLevelClass()`を使用すると、仮想サーバのデフォルト設定をオーバーライドしてリクエストに対して特定の SLM クラスを選択することができ、`slm.conforming()`を使用すると、現在の適合パーセンテージ値が返されます。

SLM クラスの TrafficScript の例

以降に示す例では、特定タイプのリクエストについてデフォルトの SLM クラスをオーバーライドする方法と、特定の SLM クラスの性能を基にトラフィックの管理方法をコントロールする方法を示します。

「FrontPage スクリプトのみ」のサービスレベルモニタリング

特にクライアントが低速な接続でコンテンツをアップロードする場合、Microsoft FrontPage™ アップロードスクリプトの実行に時間がかかることがあります。これにより、正常に動作しているにも関わらずプールに障害のマークが付けられる可能性もあります。

通常のコンテンツのレスポンス時間と FrontPage™ バイナリのレスポンス時間を区別するために、FrontPage™ リクエスト(_vti_bin 内)を検出する TrafficScript ルールを使用できます。

```
$path = http.getPath();

if( string.StartsWith( $path, "/vti_bin" ) ) {
    connection.setServiceLevelClass( "FrontPage" );
} else {
    connection.setServiceLevelClass( "Regular" );
}
```

FrontPage SLM クラスでは、仮想サーバが管理するその他全てのリクエストで 사용되는 Regular クラスよりも大幅に長いレスポンス時間を許容する必要があります。

サービスレベルモニタリングによるリソースのプライオリティ設定

この例では、プレミアムカスタマに「premium」サービスレベルモニタリングクラスのタグを付け、そのトラフィックを「premium」プールに送信します。

プレミアムカスタマ以外のカスタマは、premium SLM クラスが許容値内で機能しているときは premium プールを共有できますが、premium SLM クラスの実行速度が遅い場合は「standard」プールに送信されます。

```
if( $IsPremium ) {
    connection.setServiceLevelClass( "premium" );
    pool.use( "premium" );
} else {
    if( slm.conforming( "premium" ) == 100 ) {
        pool.use( "premium" );
    } else {
        pool.use( "standard" );
    }
}
```

```
}  
}
```

注: `pool.use()` は TrafficScript ルールを即座に終了し、リクエストを指定プールに送信します。

SLM クラス適合レートのグラフ化

特定 SLM クラスのデータのグラフを取得するには、**Activity** (アクティビティ) ボタンをクリックします。現在のアクティビティグラフの下にある **Change Data** (データ変更) をクリックします。

トラフィックマネージャに、プロット可能な様々な値のツリーが表示されます。

Service Level Monitoring Classes (サービスレベルモニタリングクラス) と示されたノードを展開します。

注: ここで、特定の SLM クラスのデータをプロットするか、全ての SLM クラスのデータをプロットするかを選択できます。また、特定の SLM クラスの平均レスポンス時間など、SLM クラスが使用する測定データの値をプロットすることもできます。全ての SLM クラスの合計接続数と合計不適合接続数を同じグラフにプロットすると、システムの全体的な性能を視覚的に比較しやすくなります。

全てのコネクションデータとレスポンス時間データをノード別にプロットすることができます。これは、十分に機能していないノードを特定するときに役立ちます。この目的には、SLM クラスが使用する各ノードの平均レスポンス時間をプロットしたグラフが適しています。SLM クラスが使用するノードのデータを選択するには、その SLM クラス名の付いたセクションを展開し、対象のノードを選択します。

CHAPTER 19 コンテンツキャッシング

本章では、トラフィックマネージャのコンテンツキャッシング機能の概要と、この機能を設定してバックエンド Web サービスノードの負荷を減らす方法について説明します。

注: トラフィックマネージャ製品によっては、コンテンツキャッシングを利用できない場合があります。この機能が必要な場合は、ソフトウェアまたはライセンスキーをアップグレードすることにより入手できます。

重要: 仮想サーバで Brocade Web Accelerator 機能を有効にしているときのキャッシュに対する影響については、CHAPTER 20 の「コンテンツキャッシング」セクションを参照してください。

はじめに

多くの Web サーバは、幅広いタイプのコンテンツをホストし、これらのコンテンツによって Web サイトと提供サービスが構成されます。例えば、一般的な Web サイトには、標準的な Web ページ(HTML ファイル)、画像(GIF、JPEG、PNG ファイル)、CSS スタイルシート、JavaScript ソースファイルが含まれます。Web サイトで提供するサービスによっては、Web サーバでホストされているアプリケーションによって生成される、パーソナライズされたカスタムコンテンツがさらに含まれる場合もあります。

トラフィックマネージャのコンテンツキャッシング機能により、トラフィックマネージャは、リクエストごとにカスタマイズされないコンテンツを特定し、そのコンテンツを記憶(キャッシング)することができます。コンテンツは、Web サーバのディスク上のファイルなど「静的」な場合と、Web サーバで実行されているアプリケーションによって生成される場合があります。

トラフィックマネージャは、内部 Web キャッシュにキャッシングしたコンテンツが別のクライアントからリクエストされた場合、リクエストをバックエンド Web サーバに転送せずに、そのコンテンツをクライアントに直接返すことができます。

これには、特に、Web サーバマシン上のリソースを消費する複雑なアプリケーションによって生成されたコンテンツをキャッシングできることをトラフィックマネージャが検出した場合に、バックエンド Web サーバの負荷を減らす効果があります。

コンテンツキャッシングの設定

コンテンツキャッシングは、HTTP 仮想サーバが実行します。仮想サーバは、頻度の高いリクエストに対するレスポンスを Web キャッシュに保存し、このようなリクエストに対するレスポンスを Web キャッシュから直接返します。

仮想サーバへのコンテンツキャッシングの適用

コンテンツキャッシングを HTTP 仮想サーバに設定するには、**Services** (サービス) をクリックし、次に該当する仮想サーバの横にある **Edit** (編集) ボタンをクリックします。

「Content Caching (コンテンツキャッシング)」までスクロールし、**Edit** (編集) をクリックします。「**webcache!enabled**」を使用して、その仮想サーバのコンテンツキャッシングを有効化または無効化します。

変更を適用するには、**Update** (更新) をクリックします。

重要: この仮想サーバに対して Brocade Web Accelerator が有効化されている場合、**webcache!enabled** の設定に関わらず、トラフィックマネージャは常に結合された画像などの最適化されたリソースをキャッシュに保存します。ただし、この設定が有効化されない限り、トラフィックマネージャは、Web Accelerator により生成された、最適化された HTML ページのキャッシングを試みません。キャッシングと Web Accelerator の両方が有効化されている場合、トラフィックマネージャは、Web Accelerator が取得する最適化されていないリソースもキャッシングします。

ライフタイムの設定

トラフィックマネージャは、頻繁にリクエストされるコンテンツを自動的にキャッシングして、Web キャッシュを最大限に活用します。

Web キャッシュを使用する各仮想サーバについて、「古くなった」リソースを破棄するまでのキャッシング時間を設定することができます。

設定	説明
webcache!time	リソースをキャッシングしておく時間(秒単位)
webcache!errorpage_time	エラーページをキャッシングしておく時間(秒単位)
webcache!refresh_time	リソースが期限切れになるどのくらい前にトラフィックマネージャがリソースの更新を開始するか

トラフィックマネージャは、通常の HTTP レスポンスを、**webcache!time** に設定された時間だけキャッシングしておきます。リソースは、リモートクライアントから

初めてリクエストされたときにキャッシュにロードされ、その後、期限切れになるまで同じリソースに対する以降のリクエストはキャッシュから直接提供されます。

注: バックエンドサーバでリソースが変更された場合、トラフィックマネージャがその変更を検出してキャッシングされたコピーを更新するまでに **webcache!time** 秒かかることがあります。コンテンツが更新された場合にそのコンテンツをキャッシュから破棄する方法については、CHAPTER 19 の「キャッシュ内の古いコンテンツの削除」セクションで説明します。

トラフィックマネージャは、エラーページ(404 Not Found、410 Gone、501 Not Implemented など)もキャッシングしますが、バックエンドサーバに対してこれらのページを再度リクエストするまでのキャッシング時間は、**webcache!errorpage_time** パラメータで決まります。一般に、エラーページは長時間キャッシングせず、より頻繁にコンテンツを再確認して、エラーメッセージを消去して正しいコンテンツに置き換えるようにします。

注: トラフィックマネージャは、サーバエラーの結果として返される「503 Too Busy」などのページはキャッシングしません。

コンテンツの期限切れ

キャッシングされたリソースが期限切れになると、キャッシュから削除され、空いたメモリが他のリソースをキャッシングするために再利用されます。削除したリソースが再度リクエストされた場合、そのリソースがバックエンドサーバから読み込まれ、再度キャッシングされます。

リソースが期限切れになると、完全なリソースが再度キャッシュにロードされるまで、そのリソースに対する全てのリクエストはバックエンドサーバに送信されます。同じリソースに対して多数のリクエストが同時に発生するか、リソースの生成に時間がかかる場合、バックエンドサーバへのトラフィックが急増する可能性があります。

この問題は、**webcache!refresh_time** 設定により解決できます。この値は、リソースの期限が近づいたときにトラフィックマネージャが投機的にキャッシュを更新する方法をコントロールします。

例えば、**refresh_time** が 10 秒に設定されているとします。この場合、リソースが期限切れになる最後の 10 秒間にトラフィックマネージャがそのリソースに対するリクエストを受信すると、トラフィックマネージャはキャッシングされているレスポンスを使用せずに、リクエストをバックエンドサーバに転送します。バックエンドサーバから有効なレスポンスが返された場合、このレスポンスによってキャッシュが更新され、期限切れカウンタがリセットされます。

この **refresh_time** 期間にトラフィックマネージャが各リソースについてバックエンドサーバに送信するリクエストは、1 秒あたり最大 1 回のみです。同じリソースに対する他のリクエストは、キャッシングされたレスポンスを使用して満たされます。

このように、キャッシュを期限切れ前に更新できるため、バックエンドサーバに大量のリクエストが送信されることはありません。

Web キャッシュのメモリ使用量の設定

Web キャッシュのメモリ使用量を設定するときは、以下の要件が満たされるよう注意深く行うことが重要です。

- 最も頻繁にリクエストされるページの全てのバリエーションを保持できる十分な容量のキャッシュにすること
- ホストマシンが実行中のトラフィックマネージャプロセスの一部をディスクにスワッピングするほどキャッシュが大きすぎないこと
- 全ての仮想サーバが同じグローバルキャッシュメモリを共有し、サイズがトラフィックマネージャシステムの **Global Settings** (グローバル設定) で設定されていること

Web キャッシュのメモリサイズを設定するには、**System** (システム) をクリックし、次に **Global Settings** (グローバル設定) タブをクリックします。**Global Settings** (グローバル設定) ページの **Cache Settings** (キャッシュ設定) セクションを開きます。

設定	説明
webcache!size	Web キャッシュが使用できるメモリ容量の上限 (Mb 単位) です。合計システムメモリのパーセンテージで指定することもできます。
webcache!max_file_num	キャッシュに保存できる最大エントリ数を制限します。
webcache!max_file_size	キャッシュに挿入される最大ファイルサイズを指定します。キャッシュの合計サイズのパーセンテージ、または絶対的なファイルサイズで指定することができます。
webcache!avg_path_length	キャッシングされるリソースに対して推定されるパスの平均的な長さ(クエリ文字列を含む)です。この数字に相当するメモリ容量を <code>webcache!max_file_num</code> 倍した容量が、キャッシュエントリ用のパスを保存するために割り当てられます。

設定	説明
webcache!max_path_length	パスをキャッシュに保存するときの、パスの最大長です。1つで必要以上に保存スペースを使い尽してしまうような、非常に長い URL を阻止するための設定です。
webcache!verbose	この設定を有効化すると、リクエスト/レスポンスをキャッシング可能かどうかを示す「X-Cache-Info」というヘッダが HTTP レスポンスに追加されます。
webcache!normalize_query	クエリ文字列の正規化を有効化します。
webcache!disk	ディスクベースのキャッシュの使用を有効化します。下の「ディスクベースのキャッシングの設定」セクションを参照してください。
webcache!disk_dir	ディスクキャッシングが有効化されている場合に、ディスクキャッシュファイルを保存するディレクトリを設定します。

キャッシュアクティビティのモニタリング

Activity (アクティビティ)をクリックし、次に **Content Cache** (コンテンツキャッシュ)タブをクリックします。このページには、Web キャッシュのコンテンツと、保存されている各レスポンスのバリエーション数が表示されます。**Content Cache** (コンテンツキャッシュ)ページには、Web キャッシュで使用されているメモリ容量も示されます。

キャッシュがいっぱいになるか、ほぼいっぱいになると、**Entries** (エントリ)値が **MaxEntries** 値に近い値になるか、または **MemUsed** 値が **MemMaximum** 値に近い値になります。この状態で新しいエントリがキャッシュに挿入されると、最も使用頻度の低いエントリが期限切れ前に破棄されます。**Oldest** (最も古いエントリ)値をモニタリングして、キャッシュ内で最も使用頻度の低いエントリの最終使用時点からの経過時間を判断することができます。

Virtual Server (仮想サーバ) > **Content Cache** (コンテンツキャッシュ)ページの「**View cache contents** (キャッシュコンテンツのビュー)」リンクをクリックすることもできます。これにより、キャッシュの使用状況の概要(割り当てられたエントリのパーセンテージおよび使用中のメモリなど)とセレクトが表示されるため、キャッシュのコンテンツを調べ、必要に応じて手動で期限切れにすることができます。

アクティビティモニタを使用して、キャッシュの使用状況をリアルタイムにグラフ化することもできます。**Activity** (アクティビティ)をクリックして **Activity Monitor** (アクティビティモニタ)タブにアクセスします。スクロールして「**HTTP Content Cache (HTTP コンテンツキャッシュ)**」レポートを選択します。このアクティビティ情報は、SNMPからも利用できます。

ディスクベースのキャッシングの設定

一般に、トラフィックマネージャのキャッシュはメインメモリ内に存在し、選択するサイズは、OS、アプリケーションメモリ、ネットワークバッファなどへの割り当て後に使用可能な RAM の容量で決まります。

または、より大容量のローカルディスク上のキャッシュを使用するようトラフィックマネージャを設定することもできます。これにより、OS は、システムのカーネルディスクバッファを使用して、ディスクベースのキャッシュの最もアクティブな部分をメモリに保存するため、効果的な 2 段階キャッシュが構成されます。

キャッシュの設定

ディスクベースのキャッシュを使用するには、**System (システム) > Global Settings (グローバル設定)** ページに移動し、このページの **Cache Settings (キャッシュ設定)** セクションを見つけます。

1. キャッシュのサイズを **Mb** または **Gb** 単位で設定します。このサイズは、サーバのメモリ容量の数倍になる場合があります。
2. **webcache!disk** 設定を使用して、ディスクベースのキャッシングを有効化します。
3. ディスクキャッシュの場所を選択します。この場所は、十分な空き容量のあるローカルディレクトリでなければなりません。また、より低速なローカル HDD (ハードディスクドライブ)ではなく、高速 SSD (ソリッドステートドライブ)ストレージを選択することを強くお勧めします。

Update (更新)をクリックして設定を保存します。ディスクベースのキャッシュを完全に適用するには、**System (システム) > Traffic Managers (トラフィックマネージャ)** ページからトラフィックマネージャソフトウェアをリスタートする必要があります。

ディスクキャッシュは永続的ではなく、トラフィックマネージャソフトウェアをリスタートするとクリアされ、リセットされます。ディスクキャッシュを複数のトラフィックマネージャシステムで共有することはできません。

SSD およびHDD ディスクキャッシュ

ディスクキャッシュは、選択した場所に作成される単一のフラットファイルに保存されます。ホスト OS は、ファイルの最も頻繁にアクセスされる部分をキャッシングし、書き込みをバックグラウンドで行いますが、キャッシュに対する全ての読み取りおよび書き込みがディスクアクセスにつながる可能性があります。

ローカル SSD ドライブの読み取り、書き込み、およびシークは非常に高速なため、キャッシュファイルにはこのドライブを使用することを強くお奨めします。HDD の読み取り、書き込み、およびシークはこれよりはるかに低速です。一般的な機械式 HDD のディスク操作は 1 秒あたり数百回に制限されるため、トラフィックマネージャがキャッシュによって処理できる 1 秒あたりのリクエスト数が制限されます。

キャッシングポリシー

全ての Web コンテンツをキャッシングできるわけではありません。トラフィックマネージャは、Web キャッシュを使用してリクエストを処理するかどうか、レスポンスをキャッシングするかどうかを、HTTP リクエストおよび HTTP レスポンス内の情報を基に決定します。

リクエスト

キャッシングできるのは、HTTP GET および HEAD リクエストのみです。その他全てのメソッドはキャッシングできません。

HTTP リクエストの `Cache-Control` ヘッダにより、トラフィックマネージャに強制的に Web キャッシュを無視してバックエンドノードと通信させることができます。

HTTP basic-auth (基本認証)を使用するリクエストはキャッシングできません。

レスポンス

HTTP レスポンスの `Cache-Control` ヘッダにより、HTTP レスポンスを Web キャッシュに保存しないよう指定することができます。

このヘッダで `max-age` 値を使用して、キャッシングされたオブジェクトのキャッシング時間を指定することもできます。この場合、レスポンスのキャッシング時間が **`webcache!time`** パラメータの設定値より短くなる場合があります。

HTTP レスポンスで `Expires` ヘッダを使用して、レスポンスのキャッシング時間をコントロールすることができます。`Expires` ヘッダは、`Cache-Control` レスポンスヘッダで `max-age` 値を使用する場合より効率が劣ります。

`Vary` HTTP レスポンスヘッダは、リソースのバリエーションのキャッシング方法と、新しいリクエストに対するレスポンスでキャッシュからどのバリエーションを提供するかをコントロールします。

Web アプリケーションでトラフィックマネージャがレスポンスをキャッシングしないようにするには、レスポンスに「`Cache-Control: no-cache`」ヘッダを追加します。

キャッシュの動作をデバッグするには、**`webcache!verbose`** グローバル設定を使用します。この設定は、**System (システム) > Global Settings (グローバル設定)** ページの **Cache Settings (キャッシュ設定)** セクションにあります。この設定を有効化すると、

トラフィックマネージャは、キャッシュポリシーがどのように有効になっているかを示す「X-Cache-Info」というヘッダを HTTP レスポンスに追加します。

HTTP キャッシュポリシーの詳細については、RFC 2616 を参照してください。

バリエント

以下の場合、1つの URL に対して複数のバリエントが存在することがあります。

- レスポンスが、Accept-Encoding ヘッダや Accept-Language ヘッダなど追加のリクエストパラメータによって異なる場合。コンテンツ提供元の Web サーバは、レスポンスに「Vary」ヘッダを含めることによってこのようなケースに該当することを示します。

```
Vary:Accept-Language, Accept-Encoding
```

- TrafficScript 関数の「http.cache.setkey()」により、レスポンスのコンテンツが他の演算パラメータに依存することが示されている場合。

バイトレンジ

バイトレンジは、ドキュメントの一部をリクエストするために一部のクライアントソフトウェアで使用されます。例えば、PDF リーダは、PDF ファイルの最初の部分(ページの索引を含む)をダウンロードしてから、バイトレンジリクエストを送信し、ユーザが読もうとしているページを読み込みます。

トラフィックマネージャでは、バイトレンジは URL のバリエントとして扱われます(詳細については、304 ページの「バリエント」を参照)。バイトレンジレスポンスはキャッシングされ、同じバイトレンジに対する別のリクエストを受信した場合、レスポンスはキャッシュから直接提供されます。

トラフィックマネージャは特定の URL のレスポンス全体をキャッシングした場合、そのレスポンスによって全てのバイトレンジリクエストを処理します。

ETag

ETag (エンティティタグ)は、Web サーバが特定レスポンスの各バージョンをユニークに識別するために使用します。トラフィックマネージャは、ETag をキャッシングし、これを考慮します。現在キャッシングされているリソースのバージョンの ETag がリモートクライアントでキャッシングされているバージョンと同じ場合、304 Not Modified レスポンスを返します。

TrafficScript を使用したコンテンツキャッシングのコントロール

HTTP リクエストの処理

トラフィックマネージャは、新しい HTTP リクエストを受信したときに、レスポンスを Web キャッシュから直接提供できるか、リクエストをバックエンドサーバに転送する必要があるかを判断する必要があります。トラフィックマネージャは、Web キャッシュをチェックする前に、リクエストに対して全ての TrafficScript リクエストルールを実行します。そのため、TrafficScript リクエストルールによってリクエストを変更し、トラフィックマネージャによる決定に影響を与えることができます。

例えば、TrafficScript リクエストルールにより、リクエストに「Cache-Control: no-cache」ヘッダを追加し、トラフィックマネージャがレスポンスをローカル Web キャッシュから提供しないようにすることができます。

HTTP レスポンスの処理

同様に、トラフィックマネージャは、HTTP レスポンスをバックエンドサーバノードから受信したときに、今後のためにレスポンスを Web キャッシュに挿入するかどうかを決定する必要があります。トラフィックマネージャは、この決定を行う前に全ての TrafficScript レスポンスルールを実行するため、TrafficScript レスポンスルールによってレスポンスを変更し、この決定に影響を与えることができます。

TrafficScript レスポンスルールにより、レスポンスに「Cache-Control: no-cache」ヘッダを追加してキャッシングされないようにしたり、「Cache-Control: max-age=60」ヘッダを追加してレスポンスを 60 秒間キャッシングするよう指示したりできます。

レスポンスが Web キャッシュから提供される場合、TrafficScript レスポンスルールは実行されません。

TrafficScript キャッシュコントロール関数

トラフィックマネージャのキャッシング動作をコントロールするためのキャッシュコントロール関数がいくつか用意されています。多くの場合、これらの関数を使用すると、HTTP リクエストおよびレスポンスのヘッダを操作する必要がなくなります。

関数	説明
<code>http.cache.disable()</code>	レスポンスルールで <code>http.cache.disable()</code> を呼び出すと、トラフィックマネージャはレスポンスをキャッシングしません。
<code>http.cache.enable()</code>	レスポンスルールで <code>http.cache.enable()</code> を呼び出すと、<code>http.cache.disable()</code> の呼び出し前の状態に戻り、トラフィックマネージャのデフォルトのキャッシングロジックが有効になります。 トラフィックマネージャに、キャッシュ不可と見なされているレスポンスを強制的にキャッシングさせることはできません。
<code>http.cache.setkey()</code>	<code>http.cache.setkey()</code> 関数は、Vary レスポンスヘッダ関数とほぼ同様に、同じリクエストの複数のバージョンを区別するために使用します。この関数はリクエストルールで使用しますが、レスポンスルールで使用することも可能です。 送信元アドレスが内部か外部か、クライアントの User-Agent ヘッダが示すブラウザが IE か Gecko ベースかなどの計算値に基づいてリクエストを区別できるため、RFC2616 の Vary のサポートより柔軟性に優れています。
<code>http.cache.exists()</code>	<code>http.cache.exists()</code> 関数は、呼び出し時点でキャッシュ内に一致するレスポンスが存在しているかどうかを確認するために、リクエストルールで使用できます。ルールがリクエストパラメータを変更せずに終了した場合、レスポンスはキャッシュから提供されます(途中で期限切れにならなかった場合)。

関数	説明
<code>http.cache.respondIfCached()</code>	この関数は、リクエストルールから呼び出すことができます。Web キャッシュ内に一致するレスポンスが存在する場合、ルールは即座に終了し、レスポンスがリモートクライアントに返され、これ以上リクエスト(またはレスポンス)ルールは実行されません。 一致するレスポンスが存在しない場合、関数が戻り、ルール処理が続行されます。 この関数は、以下のようにバックエンドノードに送信するリクエストについてはレートシェイピングし、レスポンスをトラフィックマネージャのキャッシュから直接提供できる場合はレート制限を適用する必要がない場合のレートシェイピングなどのロジックをサポートするために使用できます。 <pre>http.cache.respondIfCached(); rate.use("my rate limit");</pre>

詳細およびコードサンプルについては、『*Brocade Virtual Traffic Manager: TrafficScript Guide*』(Brocade の Web サイト: <http://www.brocade.com> から入手可能)を参照してください。

キャッシュ内の古いコンテンツの削除

負荷分散されている Web サイトのコンテンツの一部を更新した場合、トラフィックマネージャは、キャッシングされている既存のバージョンが期限切れになったときに、新しいコンテンツのみを取り出します。

手動でトラフィックマネージャに強制的にキャッシュのコンテンツを期限切れにさせ、次回クライアントからそのコンテンツがリクエストされたときに更新後のバージョンを読み込むようにすることができます。または、この処理をプログラムで行うこともできます。

キャッシングされているコンテンツの手動削除

キャッシングされているコンテンツを手動で削除するには、以下の手順に従います。

1. トラフィックマネージャ管理インターフェースで **Activity** (アクティビティ) > **Content Cache** (コンテンツキャッシュ) ページに移動します。このページには、

トラフィックマネージャクラスタにより現在キャッシングされているファイルが表示されます。

2. 必要に応じて、「**Display Cache Items** (キャッシュ項目表示)」セレクトアを使用し、一致させるホストヘッダやパスを指定して選択を絞り込み、期限切れにする項目を見つけます。
3. 期限切れにするエントリを選択し、「**Clear Selected** (選択項目をクリア)」ボタンをクリックします。

これにより、これらのリソースについてキャッシングされているコピーが即座に無効になり、トラフィックマネージャは、次回クライアントからリクエストされたときに新しいバージョンを読み込みます。

キャッシングされているコンテンツのプログラムによる削除

トラフィックマネージャコントロール API には、キャッシュ内のエントリをチェックして無効にする SOAP メソッドが含まれています。これらのメソッドは **System.Cache** インターフェースにあります。

新しいコンテンツを Web サーバにアップロードするコンテンツプロビジョニングシステムは、トラフィックマネージャコントロール API を使用したスクリプトを呼び出し、更新された全ての項目をコンテンツキャッシュで無効にする可能性があります。

詳細については、『*Brocade Virtual Traffic Manager: Control API Guide*』を参照してください。

CHAPTER 20 Web コンテンツを最適化するための Brocade Web Accelerator の使用

はじめに

Web ページのロードを高速化することにより、Web ページが表示され、使えるようになるまでの待機時間が短くなるため、満足度の高いユーザエクスペリエンスを実現できます。研究でも明らかになっているように、平均的な Web ページのロード時間が 6 秒の場合でもアプリケーションと Web サーバの内部で要している時間は 1 秒ほどに過ぎず、ユーザが待たされる残りの 4~5 秒はクライアントブラウザが Web ページをレンダリングしている時間です。Web ページのロード時間を短縮することは、ユーザエクスペリエンス全体の改善に不可欠です。

Brocade Web Accelerator(Web Accelerator)は、ロードを高速化するために Web ページのマークアップと要素を自動的に最適化します。そのための技術はトラフィックマネージャインフラストラクチャに既に装備されており、バックエンドのアプリケーションに対する変更は特に必要ありません。

動作モード

Web Accelerator はトラフィックマネージャのオプション機能であり、特別に設定されたライセンスキーで有効になります。ご使用のライセンスキーに応じて、Express と Enterprise の 2 つの機能レベルを利用できます。

Web Accelerator Express は、幅広い用途で最適に機能するように設計されており、あらかじめ定義された最適化レベルを Web コンテンツに適用します。このモードでは、Web Accelerator が有効化された仮想サーバは、処理対象の Web ページに含まれている HTML を圧縮します。また、クライアントがページリソースを取得する際に、そのページリソースに対して以下の最適化を実行します。

- JavaScript ファイルの最小化および圧縮
- スタイルシートの最小化および圧縮
- バックグラウンド画像のインライン化またはバージョニング
- Web フォントのバージョニング
- 画像コンテンツの再サンプリング
- 全てのリソースの圧縮

各項目の詳細については、「最適化手法について」を参照してください。

重要: Express モードでは、認証を必要とするサイトを最適化するために Web Accelerator を使用することはできません。詳細については、「認証」を参照してください。

注: Web Accelerator 固有の TrafficScript 関数は、Express モードのデフォルトパラメータで機能するように制限されており、オーバーライドすることはできません。詳細については、『*Brocade Virtual Traffic Manager: TrafficScript Guide*』を参照してください。

Enterprise ライセンスを使用すると、Web Accelerator の全ての最適化手法を実行できるようになり、Web Accelerator を使用して HTML ページのコンテンツをチェックしたり、変更したりできるようになります。また、Web Accelerator Enterprise では、ご使用の Web インフラストラクチャに適用される個々の最適化パラメータを完全にコントロールできます。ご使用の仮想サーバが処理する個々のアプリケーションスコープに対して、固有の最適化プロファイルを作成して適用できます。

本章では、Web Accelerator を Enterprise モードで実行する場合の特徴と機能について詳細に説明します。一部の要素は、Express モードではご利用いただけません。

サービス用の Web Accelerator の設定

管理 UI の **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページで **Web Accelerator** リンクをクリックすると、Web Accelerator を仮想サーバのプロパティとして有効化できます。

Basic Settings (基本設定) セクションで **optimizer!enabled** 設定キーを使用して、仮想サーバの Web Accelerator 機能を完全に有効化または無効化できます。

注: Web Accelerator Express を実行している場合、このセクションで以下に説明する UI コントロールを利用することはできません。参考情報としてお読みください。詳細については、サポート窓口にお問い合わせください。

仮想サーバは、指定されたポートとプロトコルのトラフィックを管理します。仮想サーバが処理するリクエストは、リクエストされた URL (<http://www.mywebsite.com/supportsystem> など) に基づいて特定のアプリケーションに向けることができます。このホスト名とパスの組み合わせを利用すると、(仮想サーバを介した) リクエスト先のアプリケーションと、その識別に使用すべき最適化プロファイルとの間にコネクションを作成できます。

Web Accelerator Profiles セクションでは、使用する各種のアクセラレーションプロファイルと、適用されるスコープのマッピングを定義します。このどちらのコンセプトも、**カタログオブジェクト**であり、このページで使用する前に作成する必要があります。これらのコンセプトについては、下のパラグラフで説明しています。

必要に応じて、複数のスコープを定義して 1 つのプロファイルにマップすることができますが、1 つの仮想サーバ内で一度に使用できるスコープは 1 つのみです(多対 1 の関係)。

Web Accelerator ウィザード

他のウィザードと同様に、トラフィックマネージャには、Web サービスで Web コンテンツの最適化を簡単に設定できる **Optimize a Web application (Web アプリケーション最適化)** ウィザードという自動プロセスが用意されています。

このウィザードを実行するには、あらかじめ少なくとも 1 つの仮想サーバが作成されている必要があります。仮想サーバがない状態でウィザードを実行することはできません。ウィザードを実行できる適切な仮想サーバを作成するには、まず **Manage a new service (新規サービスを管理)** ウィザードに従うことをお勧めします。

ウィザードを実行するには、ナビゲーションバーの「Wizards (ウィザード)」ドロップダウンメニューから **Optimize a Web application (Web アプリケーション最適化)** を選択します。

このウィザードには、以下の 3 つの主なステップがあります。

- 1) 最適化を適用する仮想サーバを選択します。
- 2) アプリケーションのスコープを選択します。
- 3) そのスコープに適用する Web Accelerator プロファイルを選択します。

アプリケーションスコープ

アプリケーションスコープは、仮想サーバでホストされている特定の Web サイトまたはアプリケーションに一致する URL の集合を定義するためのオブジェクトです。新しいアプリケーションスコープの表示、変更、および追加を行うには、**Catalogs (カタログ) > Web Accelerator Catalogs (Web Accelerator カタログ) > Application Scopes catalog (アプリケーションスコープカタログ)** をクリックします。.

仮想サーバを介してリクエストが受信されると、**Virtual Server (仮想サーバ) > Edit (編集) > Web Accelerator** で定義されているスコープのマッピング内で最も限定的な URL との一致が検索されます。トラフィックマネージャで一致が見つかり、マッピングに定義されている適切な最適化プロファイルが適用されます。

1 つのスコープレコードには、複数のホスト名を含めることができます。ただし、これらの名前は全て同じホストのエイリアスであるため、それを使用する同じリソースが取得される可能性があります。別のホスト名を使用してコンテンツ配信ネットワークからのコンテンツを提供している場合は、コンテンツがトラフィックマネ

ージャ内で同じアプリケーションスコープを通過するように、その CDN が元ホストのプロキシとして機能する必要があります。

「任意」のホスト名と一致させるには、新しいスコープレコードを作成し、hostnames リストをブランクのままにします。あるいは、特定の仮想サーバで複数のアプリケーションがホストされ、リクエストの URL で区別されることがあります。この場合、関連するホスト名とルートパスの組み合わせごとに新しいスコープレコードを設定します。

仮想サーバを介して提供されるリソースは、設定されたスコープで宣言されているホスト名と、その仮想サーバが実行しているポートの両方がそのリソースと一致する場合にのみ、最適化の対象であると見なされます。ホスト名の一致は、スコープで指定されているホスト名を使用して、またはスコープのホスト名設定がブランクの場合はページリクエストのホスト名を使用して実行されます。依存リソースのリクエストには、明示的に設定されている場合、必ずプライマリホスト名(リストの先頭)が使用されます。

外部メディアホストのように異なるホスト上にあるリソースを最適化するには、**Applies to externally hosted resources** (外部ホストリソースに適用)オプションを有効にして URL を含む Web Accelerator プロファイルを使用します。詳細については、323 ページの「アクセラレーション設定」を参照してください。

ホスト名が指定されているアプリケーションスコープを、その名前でアプリケーションをホストしていない仮想サーバに適用すると、リクエストが 502 ステータスで失敗することがあります。この HTTP ステータスが表示された場合は、アプリケーションスコープのホスト名が、その名前の表すアプリケーションによって論理的にグループ化されていることを確認してください。

注: 複数のスコープによって、一致する URL が競合しないようにしてください。

新しいスコープエントリを作成するときに、さらにスコープのアプリケーションへのルートパスを指定するには、root 設定キーを使用します。

内蔵のスコープは、「**Any hostname or path**」という名前で自動的に指定され、Web アプリケーション内で特定のホスト名やパスの指定が必要ない場合に、これを使用できます。ホスト名のフィールドはブランク、ルートのフィールドは「/」にそれぞれ設定します。

Web Accelerator プロファイル

Web Accelerator プロファイルカタログには、コンテンツの自動最適化を有効化するために Web サイトと Web アプリケーションに適用できるプロファイルのリストが含まれています。新しい Web Accelerator プロファイルの表示、変更、および追加を行うには、**Catalogs (カテゴリ) > Web Accelerator Catalogs (Web Accelerator カタログ) > Web Accelerator Profiles catalog (Web Accelerator プロファイルカタログ)** をクリックします。.

内蔵プロファイルとカスタムプロファイル

トラフィックマネージャには、あらかじめ定義された最適化設定を Web アプリケーションに適用する多数の**内蔵**プロファイルが用意されています。これらのプロファイルは変更も削除も不可能ですが、設計されている最適化設定は、ほとんどの状況に適用可能です。以下のプロファイルを選択できます。

名称	説明
Express	幅広い Web サイト用に設計された、最適化の基本的な設定および手法を提供するプロファイルです。CSS の圧縮、CSS バックグラウンド画像のインライン化、JavaScript の最小化、画像の圧縮、高度なキャッシングなどの最適化手法があります。
SharePoint 2007	Microsoft SharePoint 2007 Internet およびイントラネットのサイトのアクセラレーションと互換性のために調整された最適化設定を提供するプロファイルです。リソース URL のバージョンニング、持続性のあるキャッシング、CSS スタイルシートのインライン化、CSS の最小化、バックグラウンド画像のインライン化、JavaScript の最小化、圧縮、画像の圧縮およびメタデータ削除、高度な JavaScript 互換性などの最適化設定があります。
SharePoint 2010	Microsoft SharePoint 2010 Internet およびイントラネットのサイトのアクセラレーションと互換性のために調整された最適化設定を提供するプロファイルです。リソース URL のバージョンニング、持続性のあるキャッシング、CSS スタイルシートのインライン化、CSS の最小化、バックグラウンド画像のインライン化、JavaScript の最小化、圧縮、画像の圧縮およびメタデータ削除、高度な JavaScript 互換性、SharePoint リボンの最適化などの最適化設定があります。

名称	説明
SharePoint 2013	Microsoft SharePoint 2013 Internet およびイントラネットのサイトのアクセラレーションと互換性のために調整された最適化設定を提供するプロファイルです。ダウンロード最小化戦略(MDS、Minimal Download Strategy)という SharePoint の機能をサポートしています。ページ間を移動する際のホワイトスクリーンの影響が低減されたり、転送されるデータが低減されたりするように設計されています。 ブラウザでページ間を移動するとき、前のページとは異なる HTML のフラグメントのみが HTTP ページのレスポンスに含まれます。MDS が有効化されていない場合、各ページのレスポンスにはページ全体のコンテンツが含まれます。デフォルトでは、MDS 機能は有効化されていますが、サイトのルック&フィールがカスタマイズされると無効化されます。 SharePoint 2010 のプロファイルとは以下の違いがあります。 Web Accelerator が MDS のフラグメントをサポートし、最適化します。 SharePoint 2013 は多くの独自スクリプトをロードします。Web Accelerator は、あらかじめ設定されたスクリプトセットを使用してスクリプトコンテンツを提供することにより、この動作を最適化します。例えば、SharePoint が「/_layouts/15/init.js」を要求している場合、Web Accelerator は、それを含むスクリプトセットと、通常ロードされるその他のスクリプトを同時に提供します。 SharePoint がそれらのスクリプトのうちの 1 つを必要とする場合、それは既に Web Accelerator のスクリプトセットの一部としてブラウザキャッシュに存在しているため、即座に使用することができます。

名称	説明
SharePoint 2013 Custom Website	SharePoint 2013 のユーザインターフェースは大幅にカスタマイズすることが可能であり、デフォルトの SharePoint 2013 サイトが使用するデフォルトのスクリプトやスタイル、画像の全てが使用されないこともあります。 このプロファイルは、標準の「SharePoint 2013」プロファイルを改造したもので、ブラウザがダウンロードする SharePoint のデフォルトのスクリプトやリソースセットを必要としない、カスタマイズされた Web サイト用に設計されています。 デフォルトの SharePoint サイトには、このプロファイルではなく「SharePoint 2013」プロファイルを使用してください。
Other Web application	最適化の基本的な設定を提供するプロファイルで、ほとんどの Web アプリケーションを安全に高速化することができます。CSS スタイルシートのインライン化、CSS の最小化、バックグラウンド画像のインライン化、JavaScript の最小化、圧縮、画像の圧縮およびメタデータ削除などの最適化設定があります。このプロファイルは、Web アプリケーションの最適化要件にカスタマイズされた新しいプロファイルを作成する際に、テンプレートとして使用できます。
Basic (廃止予定)	幅広い Web サイトで適切に機能する、十分な数の基本的な最適化手法を有効化するプロファイルです。CSS の圧縮、CSS バックグラウンド画像のインライン化、JavaScript の最小化、画像の圧縮、URL のバージョニング、高度なキャッシングなどの最適化手法があります。
Advanced (廃止予定)	Basic (基本)プロファイルの機能に加え、CSS スタイルシートや HTML 画像を画像スプライトに組み合わせる、高度な最適化手法が有効になるプロファイルです。これらの手法でページのロード時間は大幅に短縮されますが、一部のサイトとの互換性を保つにはカスタマイズしたルールが必要になることがあります。この場合、カスタムの Web Accelerator プロファイルを作成するにはプロファイルを複製します(後述)。

これらのプロファイルで、Web サイトの全ての要素に対応する適切なレベルの最適化が得られない場合には、個別の要件を満たすカスタムの Web Accelerator プロファイルを新しく作成することができます。

カスタムプロファイルを作成するには、「Create a new Web Accelerator Profile Web Accelerator プロファイルの新規作成」セクションを使用して、テンプレートとしていずれかの内蔵プロファイルを指定します。必要なテンプレートプロファイルをドロップダウンリストから選択し、**Create Web Accelerator Profile (Web Accelerator プロファイルの作成)**をクリックしてください。*Edit profile* (プロファイルの編集) ページで **Save As** を使用し、内蔵プロファイルを複製してカスタムプロファイルを作成することもできます。

カスタムプロファイルには、一連のアクセラレーションオプションが追加されています。詳細については、320 ページの「カスタムアクセラレーションプロファイルについて」を参照してください。

最適化モードとページ情報

Web Accelerator は 3 つのモードのいずれかで動作します。これは `aptimizer_mode` 設定キーで定義されています。

- **Off:** アクセラレーションは無効ですが、Web Accelerator リソースへのリクエストが予約されます。
- **Stealth:** アクセラレーションは無効ですが、以下のクエリ文字列コマンドを使用して有効にする場合は例外です。

`?aptimizer=on`

- **On:** 全てのリクエストに対してアクセラレーションが有効です。

Web Accelerator プロファイルを介して渡される Web ページごとのリクエスト情報は、*Edit Profile* (プロファイルの編集) ページで `show_info_bar` 設定を使用すると表示できます。これが **Yes** (はい) に設定されていてアクセラレーションが現在アクティブな場合、このセッションの全てのページにステータスバーが表示され、そのリクエストに固有の情報を提供するコントロールが示されます。

バックグラウンドでの最適化

大きな画像セットや、ネスト化された複数のインポートを伴う CSS ファイルなど、複雑なリソースの最適化には時間がかかることがあります。Web Accelerator は、この処理を必要なリソースに対して 1 回のみ行いますが、その結果、この最適化されたリソースを初めて必要とするクライアントにとっては、その処理が完了するまで待機しなければならないことがあります。

このセクションでは、最適化されたデータの準備が整うまで、コンテンツを要求した全てのクライアントに対して最適化されていない元のサーバコンテンツを提供しながら、最適化の処理をバックグラウンドで完了させるように Web Accelerator を設定することができます。

注: 結合された画像セットなどのリソースは、最適化された後で自動的にキャッシングされます。Web ページは、仮想サーバの `webcache!enabled` 設定が **Yes** (はい) の場合にのみキャッシングされます。ただし、そのページの全てのリソースが既に最適化されている場合、完全に最適化されたページがすぐに提供されます。

トラフィックマネージャには、リソースをバックグラウンドで最適化するかどうか、およびどのように最適化するかを決めるための、2つのプロファイル設定が用意されています。

設定	説明
<code>background_after</code>	最適化がバックグラウンドに移され、元のサーバコンテンツがクライアントに提供されるまでの時間をミリ秒単位で指定します。 0 に設定すると、Web Accelerator は常に最適化の完了を待ってから、クライアントにレスポンスを送信します。 注: JavaScript ファイルなど、ファイルによっては最適化された状態で提供されなければなりません。そのため、そのようなファイルに関しては、この設定を使用してバックグラウンドで最適化することはできません。
<code>background_on_additional_resource</code>	あるページや、ネスト化されたインポートを伴う CSS ドキュメントなどのリソースの最適化処理を完了するために、Web Accelerator が追加リソースを取得する必要がある場合、バックグラウンドで最適化が行われます。

Web Accelerator による変化の測定

このセクションでは、各種のツールを使用して Web Accelerator をテストし、Web ページのパフォーマンスを測定する方法について説明します。この測定を利用すると、Web Accelerator による Web サービスのアクセラレーションの効果を判定することができます。

Web Accelerator がアクティブであることのチェック

Web コンテンツが高速化されているかどうかをチェックするには、Web ページのロード後にブラウザの「HTML ソース表示」オプションを使用します。次のコメントを検索します。

Web Accelerator による変化の測定 Web コンテンツを最適化するための Brocade Web Accelerator の使用

```
<!-- Optimized for speed - Stingray Aptimizer -->
```

このコメントが見つからない場合には、310 ページの「サービス用の Web Accelerator の設定」を参照してください。

Stealth モードを使用して Web Accelerator をテスト

開発環境やテスト環境を使用できない場合でも、Stealth モードでアクセラレータを実行して Web Accelerator をテストすることができます。

注: Web Accelerator Express では Stealth モードを利用できません。

Stealth モードの場合、Web Accelerator はクエリ文字列内でコマンドが渡されない限り、Web ページに最適化を適用しません。Web ページの動作や表示について、エンドユーザには何の変化も見えません。

Stealth モードで Web Accelerator を実行しているとき、ページやオブジェクトを要求して受信されるリクエストは全て、元のフォームのまま保存されます。リクエストのクエリ文字列に `aptimizer=on` というパラメータが含まれていると、Web Accelerator はこのブラウザセッションに対する全てのリクエストにアクセラレーションを明示的に指示するクッキーを送信します。

受信リクエストに対するクッキーの有効化が Web Accelerator で認識されると、そのスコープに適用される Web Accelerator プロファイルに定義されているルールと設定に従って、リクエストが傍受され高速化されます。

有効なリクエストの例を次に示します。

```
www.example.com?aptimizer=on  
www.example.com/destinations?aptimizer=on
```

Web Accelerator を Stealth モードで実行するように設定するには、適切なプロファイルで **aptimizer_mode** 設定キーを変更してください。

Web ページの速度の測定

Web Accelerator を有効化してから何が変わったのかを正確に把握するには、その前後で Web ページのロード時間を測定する必要があります。

最初の測定をベースラインとして使用し、後続の測定をそれと比較します。ベースラインは、まだ Web Accelerator を有効にせず、または Stealth モードで Web Accelerator を実行せずに Web サイトを運用しているときに記録するのが理想的です。

Web コンテンツを最適化するための Brocade Web Accelerator の使用 Web Accelerator による変化の測定

各段階で新しい測定のセットを取得し、ベースラインと比較することができます。各段階は、Web Accelerator の配備における以下のような任意のマイルストーンにすることができます。

- デフォルト設定の初期配備
- ルールオプションに対する変更
- CSS 設定に対する変更
- キャッシング動作に対する変更
- デフォルトの画像セットに対する変更

ツール

Web ページのロード時間を検証し、Web ページ構造にフィードバックを送る際には、様々なツールを利用できます。ごく細かいレベルで使用して個々の Web ページをテストできるツールもあれば、Web Accelerator 対応の実装に相關付けることのできるメトリクスを集めるツールもあります。

ページの構造と速度

Web ページのテストやその情報を提示することは、多くのツールで可能です。オンラインサービスとして利用できるものや、Firefox や Chrome などの Web ブラウザのアドインとして利用できるものがあります。厳密な要件は異なるかもしれませんが、代表的な例を以下に挙げます。

- WEBPAGETEST (www.webpagetest.org)は、ブラウザの動作をエミュレートして Web ページを取得する無償の Web サイトサービスです。Web ページを解析してその結果をカスケードスタイルのチャートで示します。
 - これによりレンダリングされるページをブラウザがどう処理しているかがわかります。サイトを可能な限り高速にするには、チャート上のオブジェクトのリストを手順どおりに進んで各オブジェクトに適用する最適なアクセラレーションを決定すると簡単です。例えば、チャートの初めに 2 つのリダイレクトとページのロードが表示されているとします。この場合に理想的な対応は、リダイレクトが使われないようにサイトを再構築することです。あるいは、チャートの下部にあるページにロードされるスクリプトを確認します。
- YSlow (<http://developer.yahoo.com/yslow/help>)は、Yahoo!デベロッパーネットワークプロジェクトによって開発されたブラウザアドインです。内蔵ルールを使用し、様々なパフォーマンス指標に関して Web ページを A (最高)から F (最低)までランク付けし、Web ベーススコアの改善方法を提案します。Firefox と Chrome に対応しています。

- Google Page Speed (<http://code.google.com/speed/page-speed>)も、Firefox と Chrome で使用できるブラウザアドインです。0 (最低)から 100 (最高)までの Page Speed スコアで Web ページにランクを付けます。Google Page Speed には、Web ページスコアの改善方法を提案する機能もあります。

その他のメトリクス

メトリクスによっては、長期間にわたってデータを収集しないと実用的でないものもあります。つまり、Web Accelerator を有効化するより前に収集を開始しなければならないということです。

- Google Analytics (www.google.com/analytics)は、ページのロード時間、ページビューの数、訪問ごとのページビュー、直帰率などのメトリクスを収集できる無償のツールです。Google Analytics を使用するには、Web サイトの全てのページに小さいスクリプトを追加する必要があります。結果は、このスクリプトをインストールした数日後から閲覧できますが、一定期間データを収集しないと実効的な比較はできません。
- Google Webmasters (www.google.com/webmasters)も無償のツールで、Google で Web サイトがどのように表示されるかを管理することができます。特典として、Web サイト全体のグローバルな Web ページ平均ロード時間を示すチャートが、6 か月間のローリングウィンドウで表示されます。Google Webmasters にはサイト所有者の認証が必要です。これは、DNS を一部変更するか、Web サーバに設定ファイルを追加する、またはメタタグをデフォルトの Web ページに追加して実行できます。

カスタムアクセラレーションプロファイルについて

注: Web Accelerator Express ではこのセクションを利用できません。

通常は、内蔵の Web Accelerator アクセラレーションプロファイルだけで、Web アプリケーションの汎用的な最適化は十分に可能ですが、Web サイトやアプリケーションの特定の要素に対して最適化をさらにきめ細かくコントロールするために、カスタムのアクセラレーションプロファイルが必要になる場合もあります。プロファイルの概要については、312 ページの「Web Accelerator プロファイル」を参照してください。

カスタムアクセラレーションプロファイルの Edit (編集)ページには、**Acceleration Settings** (アクセラレーション設定)ダイアログという追加のコントロールがあります。

Web コンテンツを最適化するための Brocade Web Accelerator の使用 カスタムアクセラレーションプロファイルについて

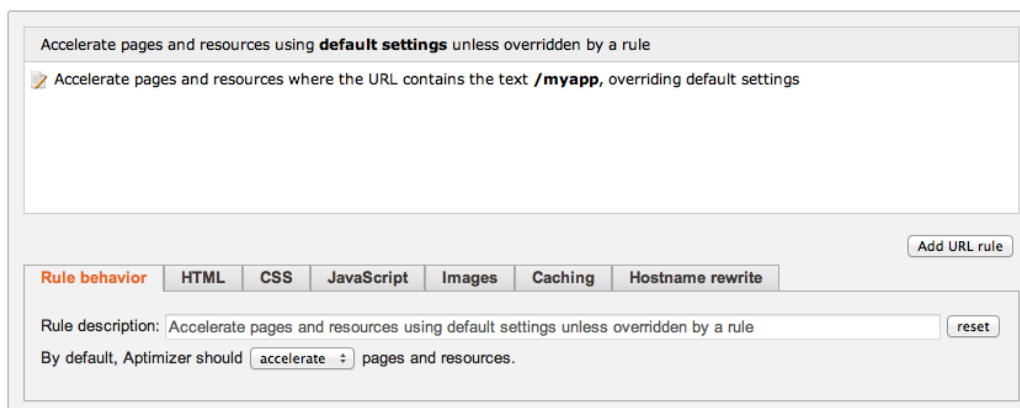


Fig 54. カスタムプロファイルの accelerations setting (アクセラレーション設定)ダイアログ

ここには、Web アプリケーションで各種のコンテンツタイプの最適化を処理するコントロールとルールがあります。以下の 2 つの部分に分かれています。

- 上部には、Web コンテンツに適用されるルールのリストが、通常は URL ロケーションまたはブラウザタイプに基づいて階層表示されます。
- その下に並んでいるタブには、現在のルールに適用されるプロパティが表示されています。特定のルールをハイライトすると、下のタブセットでそのプロパティが表示されます。

それぞれのアクセラレーションプロファイルには **default settings** (デフォルト設定) ルールがあり、これは削除できません。これを利用すると、プロファイルのデフォルト動作を定義し、プロパティの各タブのフォールバック値を設定することができます。この下に新しいルールを追加すると、最適化の動作をさらに定義したり、特定の URL (**Add URL rule** (URL ルールの追加)をクリック)に基づいてデフォルトのプロパティをオーバーライドしたりできます。各ルールバーの右にある上下の矢印をクリックすると、新しく追加したルールをプライオリティまたは目的の効果に従って並べ替えることができます。右の削除ボタンをクリックすると、完全に削除できます。

ダイアログのルールプロパティのセクションに表示されている各タブには、CSS、JavaScript、画像など一般的な Web アプリケーションを構成するコンテンツに関する設定があります。最初の **Rule behavior** (ルールの動作)タブには、このルールの一般的な説明が記載され、スコープ内の全体的な動作について、このルールで定義されているコンテンツを高速化するのか除外するのかを管理します。

プロファイルのデフォルトルール

機能制限があるモバイルブラウザで互換性を保つために、拡張と基本のどちらのプロファイルにも、各ブラウザで一部の Web Accelerator 機能を選択的に無効化する設定があります。

具体的には、以下のブラウザで画像の結合、スタイルシートの結合、CSS による画像のインライン化がそれぞれデフォルトで無効になります。

- IE mobile 7
- Opera mini
- Opera mobile
- Blackberry

また、ページのキャッシュ版が全てのブラウザに対応する汎用となるように、検索クローラエージェントに対しては特定の最適化が無効になっています。以下のクローラユーザーエージェントでは、*画像の結合*と CSS による*画像のインライン化*が無効です。

- bingbot
- googlebot

注: これらのプロファイルのデフォルトルールは、Web Accelerator 設定のカスタムルールビューには表示されません。

デフォルト動作のオーバーライド

リストに追加した新しい URL ルールにはそれぞれ独自のプロパティが設定されており、条件が一致すればデフォルトの動作をオーバーライドします。各プロパティタブには **Override** (オーバーライド)チェックボックスがあり、プロパティタブごとにこの機能を有効化または無効化することができます。特定のルールについて別の設定を指定したい場合には、**Override** (オーバーライド)チェックボックスをクリックしてそのタブの設定を有効化してください。デフォルトでは、**Override** (オーバーライド)チェックボックスを選択するまで全ての設定は無効で、淡色表示されています。

ルールのチェーン化

一定の状況におけるデフォルトの動作は、各リクエストが URL ルールのリスト中を進んでいき、一致が見つかった時点で止まるようになっています。このような一致時に停止の動作は、URL ルールで **Allow settings for this rule to chain to other rules** (このルールの設定に他のルールとのチェーン化を許可)チェックボックスを選択してオーバーライドすることができます。

このチェックボックスを有効にすると、一致したリクエストがそのルールで止まらず、2 番目のルール一致が見つかるまで検索を続行します。ただし、この設定が有効なのはそれ以降の一致が見つかるまでです。その時点で、2 番目のルールにチェーン化が設定されていない場合、リクエストはそれ以上のルールの一致を試行しません。全てのルールをプロファイル全体でテストしたい場合は、一致の数にかかわらず、それぞれでチェーン化を有効にする必要があります。

アクセラレーション設定

最適化の設定は、グループごとに別々のタブに表示されます。各タブは以下のとおりです。

Rule behaviour タブ

設定	説明
Rule Description	現在のルールの説明テキストです。このタブに含まれる基本ルールの動作設定から自動的に作成されます。例えば、URL test 文字列または rule mode を変更すると、このフィールドの説明テキストに自動的に反映されます。このテキストを独自の説明に置き換えることもできますが、その場合は自動更新が行われなくなります。 自動生成されたデフォルトの文字列に説明をリセットするには、reset (リセット) ボタンをクリックしてください。
Rule mode	このルールで識別される全てのページとリソースについて accelerate (高速化) か exclude (除外) を設定できます。 accelerate (高速化) モードがデフォルトで、一致するリクエストのルールプロパティに定義されている目的の最適化を実行します。 もうひとつのルール設定では、指定された URL test に一致するページまたはリソースについて最適化を除外します。 3 つ目の strip (ストリッピング) モードに設定されている場合は、一致が見つかったところでリソース参照がストリッピングされます。
URL test	指定したテキストに一致するリクエストにのみルールが適用されます。テキストは、Match external resources (外部リソースに一致) ボックスをチェックしている場合にはホスト名を含む完全な URL と比較され、それ以外の場合には URL パスのみと比較されます。URL に含まれる必要がある部分文字列や、URL 全体に一致するワイルドカードまたは正規表現を指定できます。

設定	説明
Applies to externally hosted resources	Web アプリケーションの外部で保持されているリソースに対する照合が可能になります。この設定を有効にすると、ルールはアプリケーションの範囲に含まれないホストに対するリクエストにのみ一致し、 URL test フィールドに含まれる文字列が、リクエストに含まれるホスト名とパスに対して照合されます。例えば、Web サイトをホストしている外部メディアから取得される Web ページ上のメディアファイルは除外したい場合があります。そのような場合には、このオプションを選択し、除外するメディアファイルの URL パスに加えてホスティングサイトの名前も含むワイルドカード文字列を URL test フィールドに指定することができます。
ルールのチェーン化	このルールの設定を他のルールにチェーン化することができます。詳細については、上の「ルールのチェーン化」セクションを参照してください。

CSS タブ

設定	説明
CSS and Stylesheet optimization	None (なし): スタイルシートは、どこで表示される場合でも元の形のまま維持されます。 Optimize CSS Content: ページと外部スタイルシートのスタイルシートブロックで、冗長な空白とコメントが全て削除されます。 ...and combine consecutive stylesheets: 連続参照として表示されるスタイルシート(間にインラインのスタイルブロックや、除外されたスタイルシートリンクない)が結合され、結合された参照のそれぞれがドキュメントで出現する最初のスタイルシートの場所に挿入されます。 ...and combine all stylesheets: ページでの出現場所にかかわらず、全てのスタイルシートが結合されます。結合された新しい参照は、ドキュメントのヘッドセクションの先頭に挿入されます。

設定	説明
CSS background image optimization	base64 でエンコードしたデータを data:uri 表記または MHTML 参照を使ってインライン化する際の、許容最大サイズ(バイト単位)です。この上限を超える画像は、URL のままになります。

JavaScript タブ

設定	説明
JavaScript optimization	Enabled (有効化): スクリプトファイルで、冗長な空白とコメントが全て削除されます。 Disabled (無効化): スクリプトファイルは元の形のまま維持されます。

Images タブ

設定	説明
Combine images	Enabled (有効化): HTML でタグとして示される画像が、以下のいずれかの手法で結合されます。 CSS スプライト化: 画像は、画像フォーマットに従って 1 つの画像スプライトに結合されます。 画像ストリーミング: 画像は、1 つ以上の画像ストリームに結合されます。 Disabled (無効化): HTML でタグとして示される画像は結合されません。
Shrink images	メタデータや冗長な色情報などの不要な情報を削除し、JPEG 画像の場合には画質を下げてファイルサイズを小さくすることによって、可能な場合に画像を圧縮します。 画像が BMP フォーマットで Web ページに表示される場合、サイズが小さくなるようにトラフィックマネージャによって自動的に JPEG フォーマットに変換されます。JPEG フォーマットへの自動変換は、ルールを使用して画像を処理対象から除外することにより避けることができます。

設定	説明
Resample quality	データ損失を伴う画像を再サンプリングする際の画質レベル(0~100%)です。画像の画質維持を優先する場合には、この値を 100 に設定してください。 デフォルトは 85 です。
Convert images	画像が、元とは異なるフォーマットに最適化されるようになります。フォーマットは、明示的に選択するか「Optimal (最適)」に設定することができます。後者に設定した場合、各画像に最適なフォーマットが自動的に選択されます。
Resize images	画像サイズが、ピクセル数で指定した幅と高さに変更されます。

Caching タブ

設定	説明
Honor resource Cache-Control header	Auto (自動): リソースのコンテンツタイプに基づき、Web Accelerator がリソースをキャッシングするかどうかを決定します。スタイルシートや画像などの静的リソースは常にキャッシングされるのに対し、JSON フォーマットの API レスポンスなどの動的リソースは、レスポンスの <i>Cache-Control</i> の指示に従います。 Always (必ず従う): Web Accelerator は、レスポンス内の全ての <i>Cache-Control</i> 指示に従います。 Never (従わない): Web Accelerator は、最適化した全てのコンテンツをキャッシングします。
Version resource URLs	Enabled (有効化): この場所にある URL が、ファイルの内容のハッシュが含まれるように書き換えられることによってバージョンングされます。バージョンングされた URL のキャッシュ有効期限は 1 年間です。 Disabled (無効化): URL は変更されず、クライアントは再チェックされるまで Cache on browser の全期間、この項目をキャッシングします。 詳細については、332 ページの「リソースの指定と URL のバージョンング」を参照してください。

設定	説明
Mark resources as cacheable by intermediary proxies	Enabled (有効化): この場所にキャッシングされるリソースが、<i>Cache-Control</i> パブリックヘッダを使ってマークされ、中間プロキシサーバによってキャッシングできるようになります。 Disabled (無効化): キッシング可能なリソースが、<i>Cache-Control</i> プライベートヘッダを使ってマークされ、中間プロキシサーバによってキャッシングできないようになります。
Optimize resources requested directly by their original URL and store them in the cache	Enabled (有効化): 画像や CSS スタイルシートなどのリソースが元の URL でリクエストされた場合に、このオプションを有効にするとリソースが最適化されキャッシングされてからクライアントに戻されます。これは、例えばページのロード後にサーバから画像その他のリソースを取得する JavaScript がサイトに含まれている場合などに便利です。 Disabled (無効化): 直接リクエストされたリソースは最適化されずに戻され、トラフィックマネージャによってキャッシングされません。
Cache on browser	リソースがクライアントのブラウザにキャッシングされ、期限切れになるまでの期間(秒単位)です。これは、<i>Expires</i> ヘッダを設定するスライド式期限として使用されます。つまり、クライアントはファイルを受け取った時点から指定された秒数の間、ファイルをキャッシングします。 この設定は、最適化され、元の URL によってリクエストされたリソースにのみ適用されます。 注: 結合リソースセット、または URL の自動バージョンングを使用するリソースは例外です。この場合、リソース(またはリソースセット)の URL はコンテンツのハッシュから生成されるため、<i>Expires</i> ヘッダを変更すると、結果としてリソース URL が変更されます。したがって、リソースは変更後の URL の強度に合わせて再リクエストが必要になります。

設定	説明
Background recheck	サーバキャッシュにあるリソースが再チェックの対象としてマークされるまでの経過時間(秒単位)です。このようにマークされたリソースがリクエストされると、キャッシュにある既存のバージョンがただちに戻されますが、変更があった場合にはキャッシングされていたバージョンがバックグラウンドタスクによって更新されます。

Hostname Rewrite タブ

設定	説明
URL hostname rewriting	一致する URL を全て書き換えることによって、現在のドメインから、指定したホストまたは URL ルートにリソースを移動します。単一のホスト名(cdn.example.com など)、またはオプションのスキームとパスも含めた完全なルート URL (https://cdn.example.com/images など)として指定できます。

最適化手法について

このセクションでは、Web Accelerator で使われている各種の最適化の原理について説明します。最初に、Web ページの速度の原則の概念と、Web Accelerator がその原則を適用して Web コンテンツを最適化する方法を取り上げます。次に、Web アプリケーションに最適化を適用するために、Web Accelerator で使用されるいくつかの手法について説明します。

Web ページの速度の原則

原則は 4 つのカテゴリに分類されます。クライアントブラウザのレスポンスが向上するように、それぞれの原則を Web アプリケーションに適用できます。

- Web ページによってロードされるオブジェクトの数を減らす
- サーバとの間で送受信されるあらゆるもののサイズを小さくする
- リピートビューの速度を上げるために可能な限りキャッシングする
- サーバがリクエストに応答する所要時間を減らす

それぞれの原則を、以下のセクションで詳しく説明します。

Web ページによってロードされるオブジェクトの数を減らす

Web ページ上の要素ごとに、ブラウザはそのリソースが保存されているホストに対して HTTP リクエストを送信します。各サーバに対して確立できる同時接続の数はブラウザによって制限されるため、Web ページの要素が少なくなるほど Web ページのロードは高速になります。Web ページのロードを最も高速化できるのは、この最適化手順です。

Web Accelerator は、適切な場合にファイルを結合して自動的にオブジェクトの数を減らし、リソースタイプ別に**結合リソースセット**を作成します。最も一般的なタイプの具体的な情報を以下に示します。

画像のインライン化と結合

画像は、どの Web ページでも重要な要素です。画像は CSS バックグラウンド画像と HTML 画像に分けることができ、それぞれでアプローチが異なります。

■ CSS バックグラウンド画像

CSS バックグラウンド画像は、カスタムスタイルシートでバックグラウンドプロパティから参照されるか、Web ページ内のスタイルプロパティから直接参照されます。このようなリソースに対するリクエストの数を減らすために、Web Accelerator はそれらを base64 データに変換し、スタイルシートファイルまたは HTML に埋め込みます。

実際のアクセラレーション動作は、ブラウザによって異なります。

- Internet Explorer (バージョン 6 および 7) の場合、base64 データは別のファイルに保存され、スタイルシートのバックグラウンド URL が MHTML 表記を使ってこのファイルを参照します。
- それ以外のブラウザの場合、base64 データは *data:uri* 表記を使って直接 CSS に埋め込まれます。

■ HTML 画像

HTML 画像は、ページの `<img>` タグから参照される画像です。HTML 画像を高速化するために、Web Accelerator は特定フォーマットの全ての画像を含む結合 CSS スプライトを作成します。

Web Accelerator は次に、このスプライトをバックグラウンド画像として参照するように CSS スタイル情報を `<img>` タグに追加し、透過 GIF を参照する *source* 属性を設定します。

JavaScript の最小化

HTML の内部から実行される JavaScript ファイルで余分なコンテンツの量を減らすと、ページのロード時間を全体的に短縮する上で効果があります。冗長な空白やコ

メントは、ソフトウェア開発の観点では有益ですが、ページの動作に対しては機能的に何の意味も持たないため、ファイルサイズを小さくするために削除されます。

スタイルシートの結合

スタイルシートのリクエスト数を減らすことは、ブラウザが Web ページのレンダリングを開始する時間を短縮する上で重要なステップです。スタイルシートを結合すると、多くの場合、安全かつ効果的にページのロード時間を短縮することができます。

第三者ドメインのコンテンツの高速化

デフォルトでは、ページの表示元である同じ Web サブドメインから提供される要素に対しても、Web ページのアクセラレーションが適用されます。例として、`www.example.com` という Web サイトで `<img>` タグの画像 URL を考えてみます。

URL	結果
<code>/images/sample1.jpg</code>	Web Accelerator で最適化
<code>www.example.com/images/sample1.jpg</code>	Web Accelerator で最適化
<code>images.example.com/sample1.jpg</code>	最適化なし
<code>images.otherexample.com/sample1.jpg</code>	最適化なし

この設定を変更して他の Web サブドメイン、さらに第三者ドメインまでを含めると、Web ページの表示に必要なパートナーのコンテンツを効果的に高速化することができます。

Web Accelerator は以下の目的でリソース参照を書き換えることができます。

- オブジェクトを CSS スプライトセットに結合する
- ドメインホストから供給される元の URL に、バージョニングを適用する(332 ページの「リソースの指定と URL のバージョニング」セクションを参照)

リソースの同時ダウンロード数を増やす手法を「ドメインシャーディング」と言い、この方法ではリソースの一部が別々の Web ドメインまたはサブドメインからダウンロードされます。ドメインシャーディングでは Web ページ上で使用される Web ドメイン名ごとに、個別に余計な DNS 解決の時間と TCP コネクション確立の時間がかかるため、このアプローチの方がドメインシャーディングよりも多くの点で優れています。また、コンテンツは Web サーバから提供されるため、一部のコンテンツが利用できないという可能性も大幅に少なくなります。

サーバとの間で送受信されるあらゆるもののサイズを小さくする

デフォルトでは、Web Accelerator はサーバから送信されるコンテンツのサイズを自動的に小さくしようとします。

- テキストベースのオブジェクトは、gzip 圧縮アルゴリズムを使用して圧縮されます。
- JavaScript とスタイルシートのファイルは、冗長な空白文字とコメントを削除して圧縮されます。
- 冗長な画像情報(メタデータなど)が削除され、JPEG 画像は元の画質の 85% (デフォルト)で再サンプリングされます。

リピートビューの速度を上げるために可能な限りキャッシングする

2 度目以降に Web ページにアクセスしたとき高速で再利用できるように、ブラウザはリソースをローカルでキャッシングします。

Web Accelerator は全てのリソースにキャッシュヘッダを自動的に適用し、それらのリソースの管理に必要な情報をブラウザに提供します。デフォルトでは、Web Accelerator がリソースに適用するキャッシング期間はそれが参照される方法に応じて異なります。以下の表のとおりです。

条件	キャッシュ期間
オブジェクトがリソースセットに結合されている	1 年間
オブジェクト URL が Web Accelerator によってバージョンニングされている	90 日間
元のオブジェクト URL がリクエストされた	1 時間

パブリックプロキシのキャッシング

オプションで、Web Accelerator は中間キャッシングを行うプロキシサーバを利用できます。これは企業環境やインターネットサービスプロバイダ(ISP)では一般的です。特定の属性を使用して、Web Accelerator は特定のリソースの全てまたは一部に「cache-control: public」ヘッダを追加します。

リソースをプロキシキャッシュ可能とマークすると、他のユーザが既にアクセスしたことがある Web サイトの Web ページにアクセスする場合には、最初のビューでもリピートビューでもロード時間が大幅に短縮されます。

サーバがリクエストに応答する所要時間を減らす

この原則が適用されるのは、サーバがクライアントのブラウザに HTML レスポンスの送信を開始するまでに要する時間です。前述したように、これで Web ページのロード時間は最大 20%短縮されます。

ロード時間の短縮は、サーバ、アプリケーション設計、保守に直接関係します。考えられる解決法としては、スケールアップ(既存のサーバにプロセッサやメモリなどのシステムリソースを追加する方法)、スケールアウト(サーバノードを増やし、さらに多くのノード間に負荷を分散する方法)、コードのプロファイリングとチューニング、データベースのチューニング、シャーディング(データベースを複数のサーバに分散する方法)、Web サーバのチューニングなどがあります。

この原則は Web サーバとアプリケーションに適用されるため、本章ではこれ以上扱いません。

リソースの指定と URL のバージョニング

Web Accelerator は、Web ページを高速化するプロセスで作成される結合リソースセットまたは URL が自動バージョニングされたリソースにそれぞれ新しい ID を作成します。この方法によってトラフィックマネージャでは、トラフィックマネージャを何度もチェックして変更の有無を確認することをブラウザに求める代わりに、リソースの変更をブラウザに通知することができます。これを可能にしているのが、リソースの URL をそのコンテンツによって決める仕組みです。

例えば、あるリソースセットを以下のように指定します。

```
/optimized-gif-kjmMLxSEdXYqXt3coAasow4F=-ds3432d.gif
```

こうすると、自動バージョニングされた URL が以下ようになります。

```
/images/optimized-jpg-23dgdstds5563gfv=-P9vry3Uq22.jpg
```

これらの URL における数字と文字の文字列は、特定のリソースを構成する設定、コンテンツ、および下位リソースのユニークなハッシュ値を表しています。設定を変更するたびに、その設定が適用されるリソースは自動的に再計算されます。したがって、リソースのコンテンツが変更されれば、このユニークな値も変更されます。

通常の場合、トラフィックマネージャは自動バージョニングを有効にせず単にページを表示するだけであるため、実行される典型的なデータフローは以下の図のようになります。

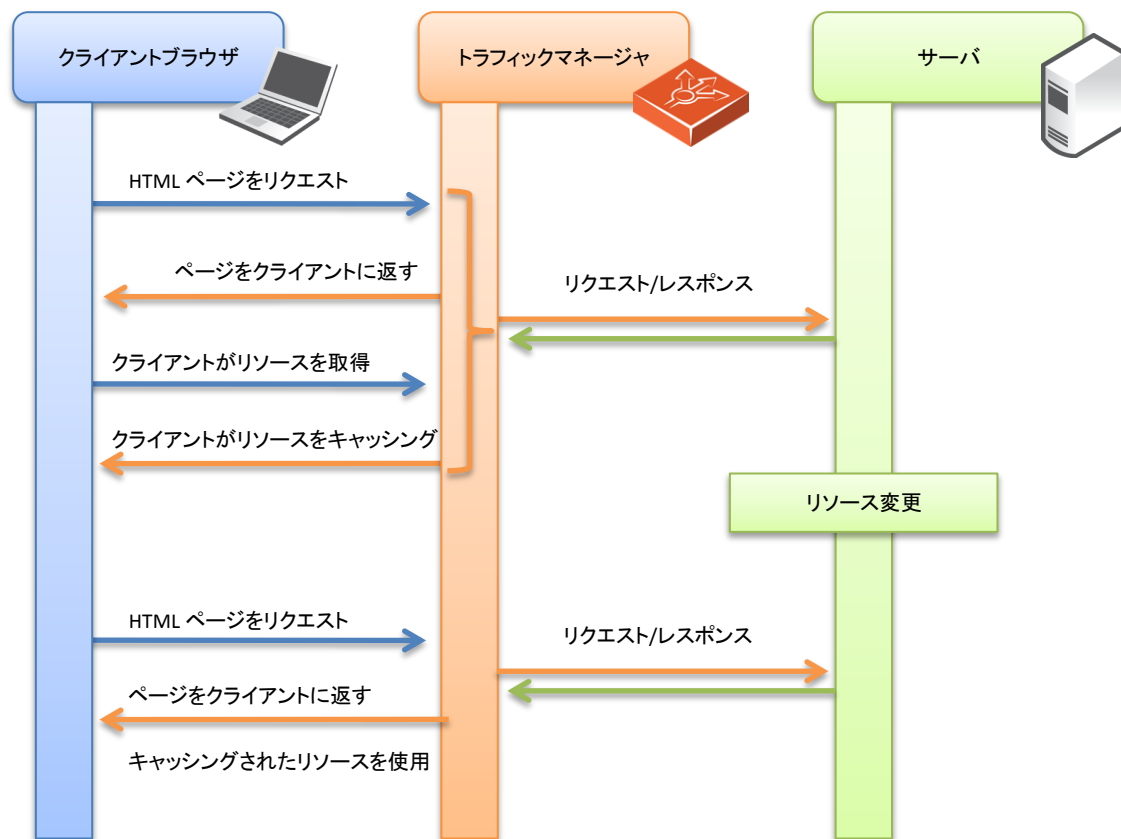


Fig 55. 自動バージョンングが無効な場合の典型的なデータフロー

1. クライアントが、トラフィックマネージャで実行されているサービスを介して HTML ページをリクエストします。
2. クライアントが、通常のリソース(スタイルシート、画像など)へのリンクを含む HTML ページを受信します。
3. クライアントがそれらのリソースを取得します。
4. トラフィックマネージャがバックエンドサーバからそのリソースを供給します。
Web Accelerator の設定によっては最適化されています。
5. 使用可能なリソースをクライアントがローカルにキャッシングします。
6. 何らかの理由(バグ修正、レイアウト変更など)により、バックエンドサーバでリソースの一部が変更されます。
7. クライアントが、同じ Web サイト上の別の HTML ページをリクエストします。
8. クライアントが、同じリソースへのリンクを含む HTML ページをトラフィックマネージャから受信します。
9. クライアントはサーバ上の変更を認識していないため、ローカルにキャッシングされた(期限が切れていない)バージョンのリソースを参照し、Web サービスからは再リクエストしません。

最後のステップでわかるように、クライアントは一定時間をおいてからキャッシュされたリソースを再リクエストするため、そのリソースは変更されている場合も変更されていない場合もありうるというのが、このシナリオで避けられない1つの欠陥です。そのためクライアントは、何らかのイベントが発生して再リクエストを要求するまで、古いバージョンのリソースを使い続けることになります。

自動バージョンニングを有効にすると、クライアントが常に Web ページに含まれる最新版のリソースを使っていることがトラフィックマネージャで保証されるようになります。これを示したのが以下の図です。

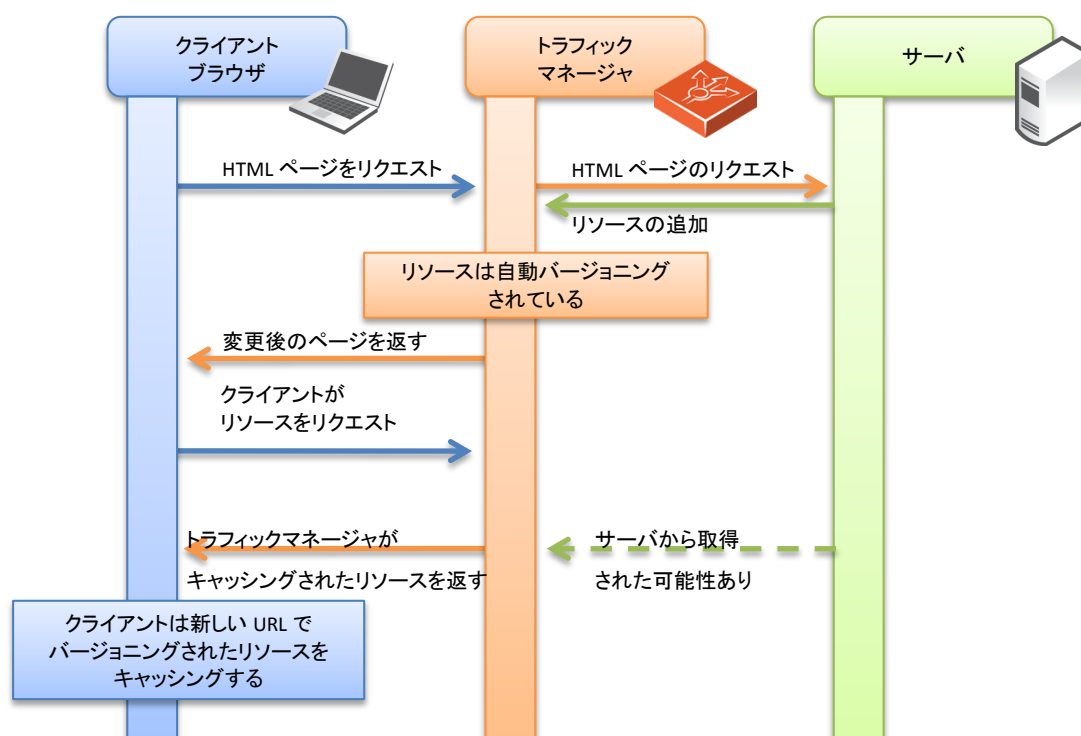


Fig 56. 自動バージョンングを有効にした場合の典型的なデータフロー

1. クライアントが、トラフィックマネージャを介して提供される Web サービスから HTML ページをリクエストします。
2. HTML ページで 1 つ以上のリソースが参照されていることをトラフィックマネージャが認識します。
3. トラフィックマネージャがそれらのリソースを取得し(また、Web Accelerator の設定によっては最適化し)、その内容確認に基づいてハッシュ値を作成します。
4. トラフィックマネージャが、計算済みのハッシュ値を含むリソースの新しい URL を作成し、HTML ページで参照箇所を全て置き換えます。
5. 変更されたページがクライアントに返されます。
6. クライアントが、新しい URL を介してページ内リソースをトラフィックマネージャからリクエストします。

7. トラフィックマネージャが、指定されたコンテンツをそのキャッシュから供給します(または、キャッシュになればバックエンドサーバから取得します)。
8. クライアントが、新しい URL を使用してリソースをローカルにキャッシングします。

これ以降のページリクエストに対しては、(バックエンドのリソースコンテンツに基づいて)各リソースの計算済みのハッシュ値が変更されていないことをテストした上で、トラフィックマネージャが応答します。

1. クライアントが、別の HTML ページをリクエストします。
2. トラフィックマネージャは、含まれるリソースがバックエンドサーバで変更されていないことを確認し、計算済みで前回と同じハッシュ値 URL に書き込みます。
3. クライアントは、キャッシングされたコピーのリソースを使用します。

バックエンドサーバでリソースが変更された場合、これ以降のページリクエストは計算済みのハッシュ値が異なることによって影響されます。以下のようなシナリオになります。

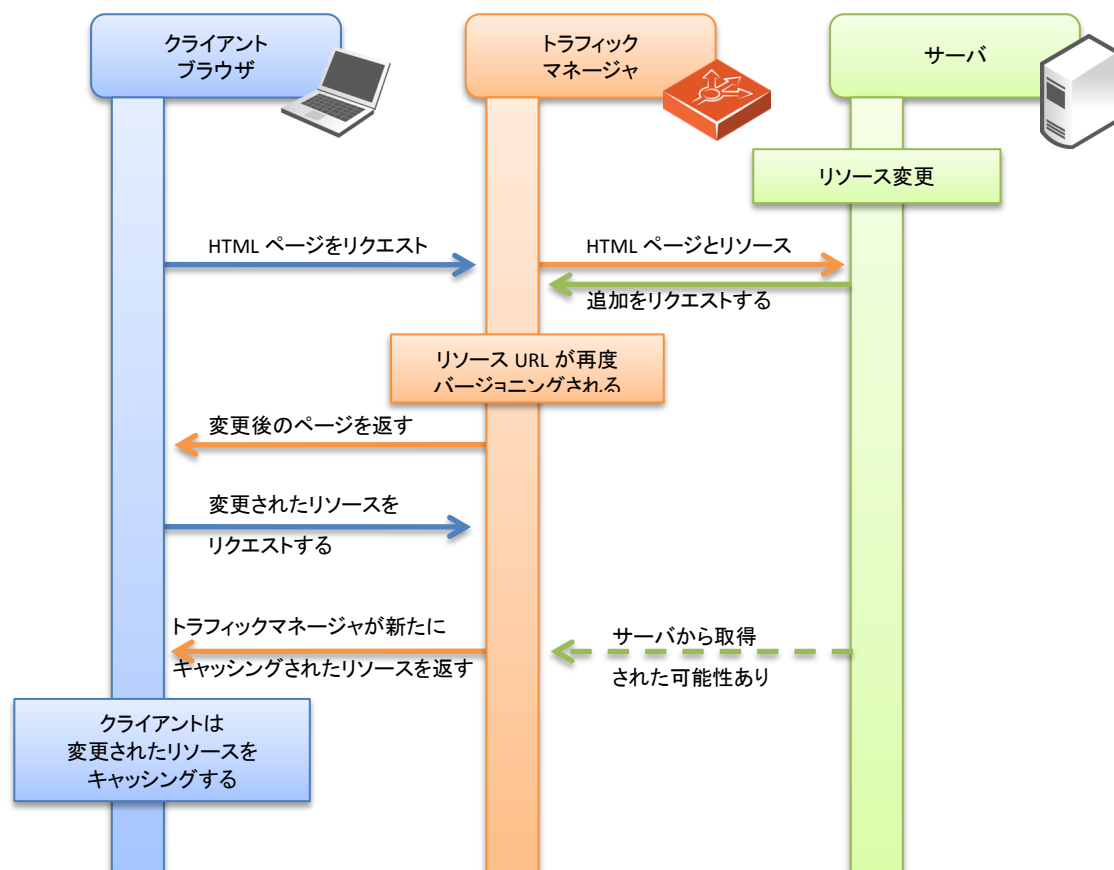


Fig 57. リソースが変更された後の典型的なデータフロー

1. リソースの変更により、コンテンツベースの URL が強制的に再計算されます。バックエンドサーバで 1 つ以上のリソースが変更されます。
2. クライアントが、別の HTML ページをリクエストします。
3. トラフィックマネージャは、目的のリソースがバックエンドサーバで変更されていることを確認し、新しいコンテンツに基づいて(以前の URL と異なる)新しい URL を作成します。
4. クライアントは新しい URL のキャッシュエントリを持たないため、トラフィックマネージャにそれをリクエストし、取得します。
5. クライアントはただちに、変更があったリソースの新しいバージョンを使い始めます。

コンテンツ分配ネットワークの使用

コンテンツ分配(または配信)ネットワーク(CDN)は、地理的に分散しているサーバ上のリソースの一部または全部をホストし、Web サーバとユーザの間に遅延を抑えます。CDN の詳細については、

http://en.wikipedia.org/wiki/Content_distribution_network を参照してください。

Web Accelerator プロファイルの編集ページで **URL rewrite to CDN** (CDN に URL 書き換え)設定を使用すると、Web Accelerator で CDN を利用できます。

この設定でトラフィックマネージャは、ルールに一致するリクエストの HTML を書き換えるよう指示します。これによって、リソースは独自の Web サーバではなく CDN で指定されたドメインからダウンロードされるようになります。

CDN ホストにアクセスする際、トラフィックマネージャは受信したリクエストのスキーム(HTTP/HTTPS)を自動的に使用しますが、必要な場合には **URL rewrite to CDN** (CDN に URL 書き換え)設定でスキームと仮想パスを任意に指定することもできます。このため、ターゲットの Web サイトが SSL コネクションを終端する別のデバイスを介してリバースプロキシされている場合には、全てのトラフィックを SSL に切り替えることが可能になります。

Web Accelerator のトラブルシューティング

注: Web Accelerator Express をご使用の場合、このセクションで説明されている一部の UI コントロールおよび設定オプションは該当しません。

予想外の動作の コントロール

トラフィックマネージャが、問題のあるコンテンツや標準外のユーザ定義されたコンテンツに直面すると、Web Accelerator のプロセスが強制的に障害状態になること

があります。このようなことが起きると、トラフィックマネージャはエラーメッセージとともにイベントログに障害を記録してからプロセスを再開します。

状況によっては、エラーを起こしやすいコンテンツによって障害の発生と Web Accelerator のプロセスの再開が繰り返されることもあります。このような事態は、トラフィックマネージャの全体的なパフォーマンスの妨げになり、トラブルシューティングをいっそう複雑化します。これに対応するため、トラフィックマネージャにはウォッチドッグ機能が用意されています。この機能を使用すると、設定した制限に達すると Web Accelerator のプロセス再開が無効化され、頻度の高い障害による影響を抑えることができます。

ウォッチドッグを設定するには、**System (システム) > Global Settings (グローバル設定) > Web Accelerator** をクリックします。このセクションには、以下の設定があります。

設定	説明
aptimizer!watchdog_limit	aptimizer!watchdog_interval 設定で定義された期間内に Web Accelerator のプロセスが開始または再開される最大回数です。1 台のトラフィックマネージャでこの回数だけプロセスが失敗すると、クラスタ内の全てのトラフィックマネージャで Web Accelerator が無効化され、 System (システム) > Diagnose (診断) ページから手動でリスタートしなければなりません。 この制限を 0 に設定すると、Web Accelerator は無効化されなくなります。
aptimizer!watchdog_interval	ここで設定した期間(秒単位)を経過すると、前回の障害は、aptimizer!watchdog_limit で設定されている制限に加算されなくなります。

注: ほとんどの場合、これらの設定を使用または変更する必要はありません。Web Accelerator で頻繁に障害が発生する場合は、サポート窓口にお問い合わせください。

Web Accelerator のその他のグローバル設定については、345 ページの「その他の設定可能なグローバル設定」を参照してください。

その他のトラフィックマネージャ機能との相互作用

実際には、仮想サーバで Web Accelerator を有効にしても、他のトラフィックマネージャ機能で相互運用性の問題が発生することはありません。しかし、内蔵されている一部の保護およびパフォーマンス機能の性質上、Web Accelerator と他の機能との相互作用を理解しておくことは重要です。

依存リクエストの問題

以下の図は、依存リクエストと呼ばれるものによる影響を示しています。これは、トラフィックマネージャによって処理された 1 つの Web ページリクエストに続いて、返されたページに含まれる依存リソースを要求するようなリクエストです。

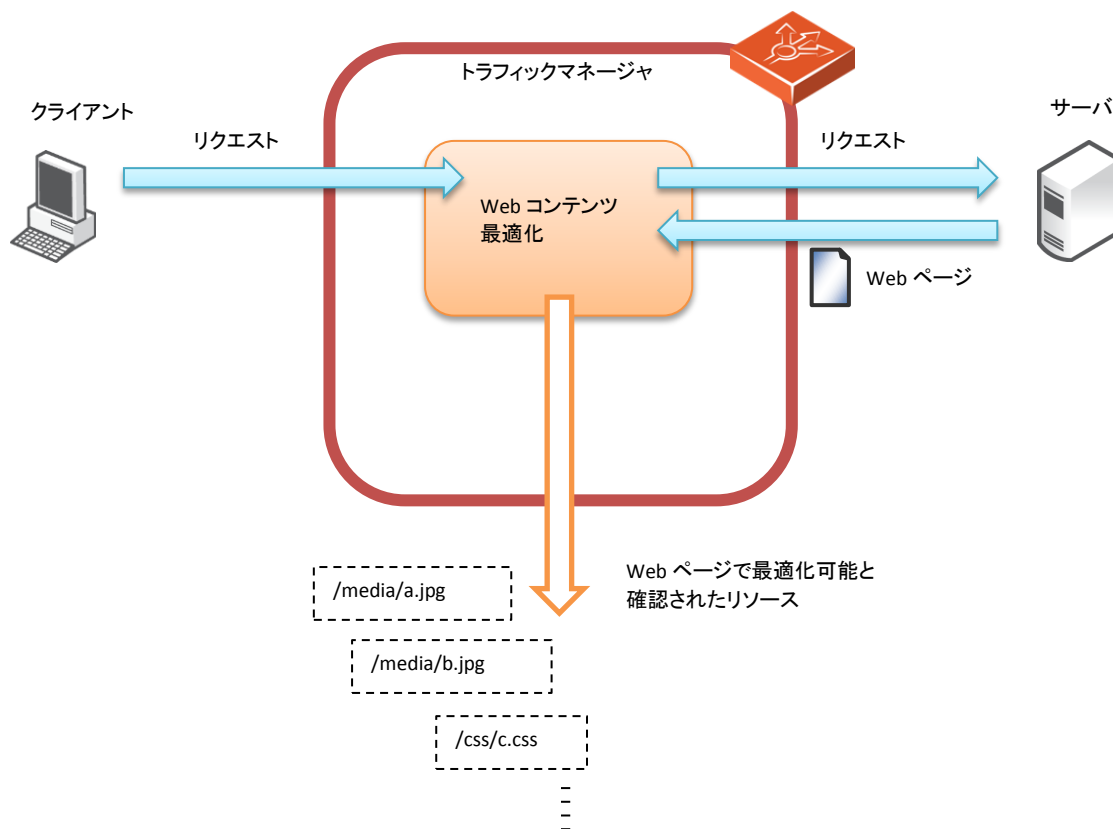


Fig 58. 最適化可能なリソースの識別

最初にクライアントが、トラフィックマネージャによって処理されるサービスを Web ページにリクエストします。Web Accelerator がこの Web アプリケーションに対して有効で、最適化可能なリソースを識別しようとします。

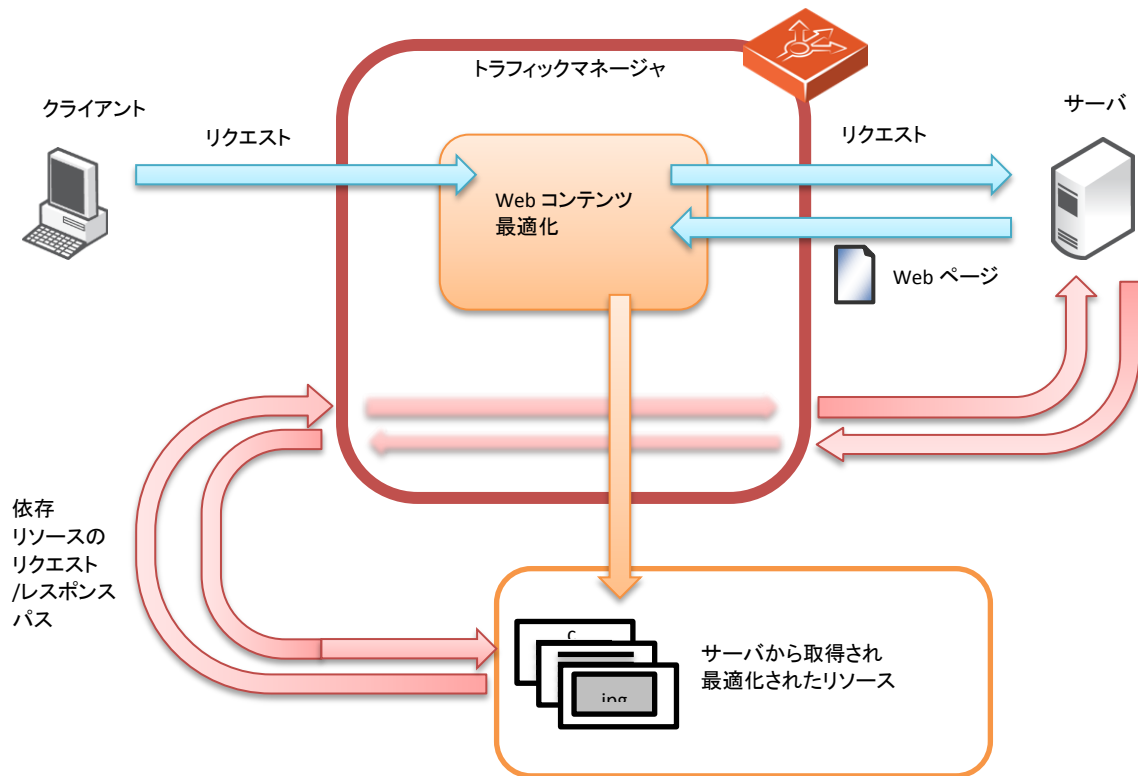


Fig 59. 依存リクエストレスポンスのモデル

次にトラフィックマネージャは、アプリケーションのページロード時間を短縮するためにストリッピングされ、または最適化された各種のリソースを探して取得できるように、追加の依存リクエストを多数実行する必要があります。

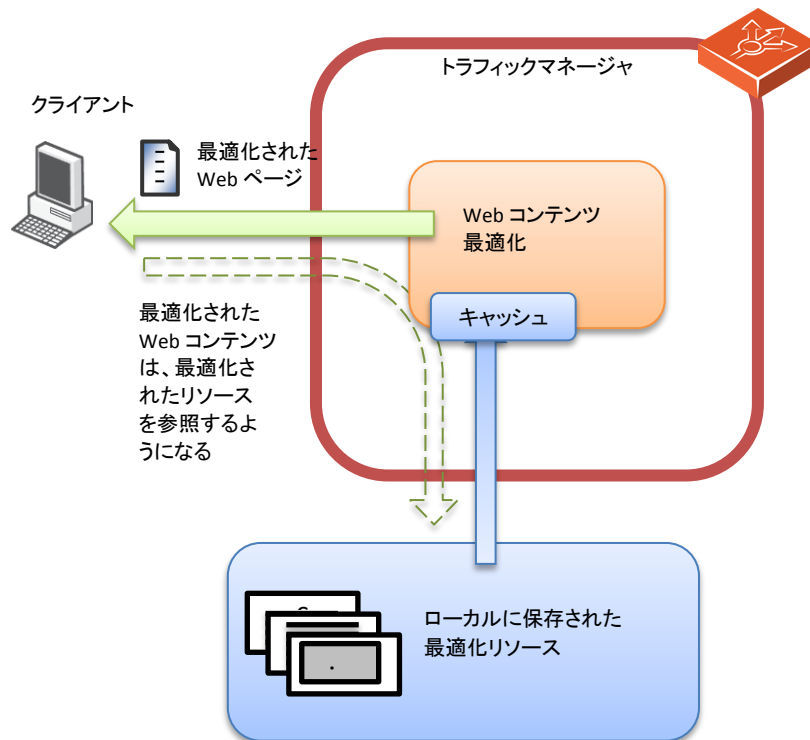


Fig 60. キャッシングされ、Web Accelerator によって供給される最適化されたリソース

それらのリソースを取得すると、適切なリソースが最適化され、Web Accelerator によってローカルでキャッシングされます。これ以降のページリクエストには、最適化されたコンテンツへの変更後の URL が含まれ、それがトラフィックマネージャから直接供給されます。

依存リクエストは、トラフィックマネージャを介して一般的なリクエストとして処理され、通常の処理が適用されます。ただし、これらのリクエストには、Web Accelerator を介して直接行われたリクエスト/レスポンスと同じ識別ヘッダ、クッキー、その他の要素が含まれているとは限らないため、予期しない結果が生じる可能性があります。

以下のセクションでは、このような依存リクエストの取得に関する問題点を取り上げます。

サービスプロテクション

依存リクエストを取得すると、場合によってサービスプロテクションクラスが実行されることがあります(クラスタでトラフィックマネージャの 1 つからリクエストが大量に受信されるため)。これは、クラス設定の **Access Restrictions** (アクセス制限) セクションで管理されている **Allowed IPs** (許可 IP) (ホワイトリストとも言う) のリストに適切なトラフィックマネージャの IP アドレスを追加することで防ぐことができます。サービスプロテクションの特徴と機能については、CHAPTER 15 を参照してください。

TrafficScript ルール

Web Accelerator が有効なとき、TrafficScript ルールの通常の処理順序は以下のようになります。

リクエストの場合:

クライアントリクエストを受信 -> TS リクエストルール -> Web Accelerator -> バックエンドノードに送信

レスポンスの場合:

バックエンドからレスポンスを受信 -> TS レスポンスルール -> Web Accelerator -> クライアントに送信

注: 依存リクエストの場合、TrafficScript ルールは通常どおりに実行されます。

クライアントの IP アドレス、ブラウザヘッダ、クッキー、認証データなど、クライアントに固有の情報を利用する TrafficScript ルールは、依存リクエスト側にそのような情報がないため、予期しない動作を引き起こす場合があります。

http.stream.* TrafficScript 関数の使用によっても、レスポンスの最適化が妨げられることに注意して下さい。

SSL 復号化

SSL 復号化を使用し、かつクライアント証明書を必要とする仮想サーバを最適化するためにトラフィックマネージャを使用することはできません。これは、Web Accelerator が追加のリソースに対して取得を開始するとき、クライアント証明書が提供されないためです。

認証

Express モードでは、認証を採用しているサービスを最適化するためにトラフィックマネージャを使用することはできません。Enterprise モードでは、トラフィックマネージャは NTLM を使用してサービスを最適化することができます。その他のクライアント認証方法はサポートしていません。

コンテンツキャッシング

Web Accelerator を介して渡され、Web Accelerator のスコープ内にあると見なされる Web コンテンツは、通常のトラフィックマネージャ Web キャッシュで通常どおりにキャッシングされることはありません。代わりに、最適化された Web コンテンツ/リソースのローカルコピーを保持するために設計されている専用の Web Accelerator キャッシュに保持され、そこから供給されます。この動作をコントロールするには、Web Accelerator Profile (Web Accelerator プロファイル)編集ページで「Caching (キャッシング)」タブの設定を使用します。

ランタイムエラー

ランタイムエラーは通常、サーバに関連して発生します。ランタイムエラーが発生する原因は、サーバの誤設定、またはサードパーティとの干渉(ルータ、ファイアウォール、ロードバランサ)です。一般には、Web サイトに全くアクセスできなくなるか、Web サイトにアクセスできてもリソースが欠落した状態になります。

兆候: Web Accelerator が有効な場合にのみエラーページが表示される。リソースが 400 または 404 HTTP ステータスで返される。アプリケーションプールが有効な場合でも、サーバから Web ページを返すには非常に時間がかかる。

問題: URL リライターまたは他の HttpModule が、Web Accelerator リソースセットの URL を処理しようとしています。

解決方法: /optimized-gif-*.gif を含む URL を、他の HttpModule による URL 書き換えまたは処理から除外します。

兆候: Web サービスが、ローカルでアクセスしたときには動作するが、外部クライアントからアクセスしたときには動作しない。

問題: プロキシまたはアップストリームのデバイスが/optimized-gif-*.gif URL をブロックまたは変更しているため、無効になっています。

解決方法: /optimized-*に対するリクエストを、変更しないままホストサーバに到達するように許可します。

兆候: HTML にコメント「<!-- Optimized for speed - Stingray Aptimizer -->」が表示されない。

問題: ブラウザで「HTML ソース表示」機能を使用すると、「<!-- Optimized for speed - Stingray Aptimizer -->」というコメントがなく、これは Web ページが高速化されていないことを意味します。

解決方法:

- Web Accelerator が特定の仮想サーバに対して有効であることを確認します。
- URL がルールセットで除外されていないことを確認します。
- Web Accelerator が **Stealth** モードで動作している場合は、URL リクエストで?aptimizer=on クエリ文字列を使用します。
- URL が除外されておらず、Web Accelerator が完全に有効な場合は、URL リクエストで?aptimizer=off が使用されていないことを確認します。

画像エラー

Web ブラウザで、指定された位置に画像が表示されない、または表示される画像のサイズが正しくない場合は、画像エラーです。Web Accelerator はページにアクセスするたびに新しいリソースセットを作成するため、画像エラーは「目に見えない」場合もあります。

リソースセットの作成に関して問題を特定するには、ページの要素を調べます。1 回作成されて何度もキャッシュから利用される一般的な要素には、特に注意してください。

兆候: ページにアクセスするたびに、新しいスプライトセットが作成されるように見える。

問題: ページの画像のリストが毎回異なるため、ページアクセスのたびに Web Accelerator が新しいスプライトセットを作成しています。これが原因で、プロセッサが過剰に消費される、キャッシュメモリの使用率が高くなる、サーバでパフォーマンスが全体的に低下する、などの影響が現れます。

解決方法: 動的な画像を結合しない形で含む URL ルールを追加します(**Images** タブで **Combine HTML images into image sprites** (HTML 画像を画像スプライトに結合)設定を無効にする)。

兆候: ページ上の画像でスペースが失われている、またはページ上で位置が揃っていない。

問題: スペースの挿入は、画像を隣接する UI 要素から分けるために使用されます。

解決方法: 動的な画像を結合しない形で含む URL ルールを追加します(例えば、**Images** タブで **Combine HTML images into image sprites** (HTML 画像を画像スプライトに結合)設定を無効にする)。

兆候: Web Accelerator を有効にすると、いずれかの画像が大きく見える。

問題: 画像を含むコンテナが、スタイルシートを適用して画像サイズを原寸から変更しています。

解決方法:

- `<img>`要素が `width` 属性と `height` 属性を持つようにページテンプレートを変更します。これによって、Web Accelerator は表示される画像の適切なサイズを判別できるようになり、ブラウザもページレイアウトを短時間で計算できるようになるため、レンダリングが高速になります。
- 動的な画像を結合しない形で含める URL ルールを追加します(例えば、**Images** タブで **Combine HTML images into image sprites** (HTML 画像を画像スプライトに結合)設定を無効にします)。

兆候: アクセラレーションは動作しているが、レイアウトに問題がある。

問題: これは通例、スタイルシートまたは HTML 画像に関係しています。

解決方法: スタイルシートオブジェクトと HTML 画像の数を減らします。この問題の原因になっている手法が何であっても、それを回避するにはアクセラレーションルールを使用する必要があります。

CSS エラー

Web ブラウザで、指定された位置に画像が表示されない、または表示される画像のサイズが正しくない場合は、画像エラーの可能性あります。Web Accelerator はページにアクセスするたびに新しいリソースセットを作成するため、画像エラーは「目に見えない」場合もあります。

リソースセットの作成に関して問題を特定するには、ページの要素を調べます。1 回作成されて何度もキャッシュから利用される一般的な要素には、特に注意してください。

兆候: Internet Explorer で、スタイルシートに定義されているバックグラウンド画像が適用されない。

問題: これは、Internet Explorer で設定されているセキュリティポリシーが原因です。あるドメインの HTML ページが別のドメイン上のリソースを参照しており、そのリソースタイプが rfc822 である場合、ブラウザにはロードされますが、信頼できるかどうか不明なため処理はされません。ページと同じ TLD 上の外部リソースは、暗黙的に信頼されます。

解決方法: メイン Web サイトのサブドメインとなる CDN ホスト名を変更し、そのリソースが暗黙的に信頼されるようにします。

JavaScript エラー

Web Accelerator を有効化する前に、最新のブラウザで検出されるような JavaScript エラーがないことを確認することが重要です。そうすれば、現在の問題が最適化の結果かどうかを判定できるからです。疑わしい場合には、Web Accelerator を無効にした状態で再度ページをリクエストしてください。アクセラレータがバイパスされるので、最適化とは別に JavaScript エラーが存在するのかどうかをチェックできます。

Internet Explorer 9 では、開発者ツールとデバッグ情報を利用できます。Firefox と Chrome では、Firebug アドオンをインストールして同様の情報を利用することができます。ほとんどのブラウザには、インストールされているツールやプラグインにかかわらず、JavaScript エラーが発生したときにメッセージ/通知を識別する機能があります。このメッセージは、どのスクリプトが問題の原因になっているかを開示するときや、コードで問題のあるセクションの位置または行番号を特定するときによく役立ちます。

JavaScript エラーのデバッグ

Web Accelerator では特定の URL にルールを適用できるため、Web サイトのデフォルト設定に手を加えることなく、個々の(またはグループの)リソースに対する特定の最適化をオフにすることができます。

Web Accelerator を有効にした状態で JavaScript エラーを解決するには、以下の 1 つ以上を実行します。

- 任意の照合ルールの **JavaScript** タブで、**JavaScript optimization** (JavaScript 最適化)オプションを無効にします。
- どのスクリプトが機能していないかを特定し、それを最適化から完全に除外する URL ルールを作成します。
- Web Accelerator が JavaScript エラーの原因になっているかどうかは、Web サイトで**全ての JavaScript 最適化**を除外してみればすぐに判明します。.js を含む URL に対する最適化を全て除外する一時的な URL ルールを作成してください。

一般的な JavaScript エラー

兆候: スクリプトが実行され、予期しない結果が返される。

問題: 外部スクリプト参照がページで削除されたことによるエラーです。

解決方法: 外部スクリプトと他のスクリプトファイル、またはインラインスクリプトブロックとの間に、順序に依存するコードが存在する場合は、それがサイトの正常な機能を阻害している可能性があります。JavaScript ライブラリを作成する際のベストプラクティスは、関数のライブラリとして扱い、直接のコードは実行しないことです。こうすれば、スクリプトは可能な限り結合されます。設計に問題のあるスクリプトの回避策としては、適切な URL ルールで完全に Web Accelerator から除外する方法があります。

兆候: スクリプトのロードと実行に失敗する。

問題: スクリプトライブラリを動的にページにロードする JavaScript フレームワークに問題があります。

解決方法: これらのフレームワークは通常、使用するスクリプトに対する参照を検索し、一種の登録機能を使用してロードするスクリプトを挿入します。Web Accelerator は結合されたスクリプトを代理 URL で参照するため、このように動作するフレームワークが混乱する原因になる可能性があります。この場合の解決方法としては、これらのスクリプトを除外するか、ロード側のコードが Web Accelerator による変更を理解できるように変更してください。

兆候: JavaScript の非同期オプションを使用すると、JavaScript の一部が Web ページで実行されない。

問題: 広告をブロックする一部のソフトウェアは、JavaScript を非同期にロードして実行するための埋め込み Web Accelerator コードを誤って除外してしまうことがあります。

解決方法: 必要なスクリプトを非同期コードから除外する URL ルールを使用するか、JavaScript を非同期モードでロードして実行するオプションを無効にします。

その他の設定可能なグローバル設定

System (システム) > Global Settings (グローバル設定) > Web Accelerator に設定可能な追加のグローバル設定があります。このセクションには、以下の設定があります。

設定	説明
aptimizer!max_original_content_buffer_size	Web Accelerator による最適化が可能な、単一のバックエンドレスポンスの最大コンテンツサイズです。これよりも大きいレスポンスは、最適化されることなくクライアントに返されます。 注: バックエンドレスポンスが圧縮される場合、この設定はレスポンスボディの圧縮後のサイズに適用されます。
aptimizer!max_dependent_fetch_size	Web Accelerator に送られ、Web Accelerator がサブプロセスで処理できる依存リソース(その時点で最適化が行われているコンテンツの一部が依存しているため、Web Accelerator がバックエンドから要求したリソース)の最大サイズです。この値よりも大きなリソースは、Web Accelerator によって常に無視され、最適化されることなくクライアントに返されます。 制限を無効化するには、このフィールドを 0 に設定します。

通常は、これらの設定を変更する必要はありません。トラフィックマネージャは、通常、あらゆるシステムにおいて良好なパフォーマンスレベルを提供するデフォルト値を使用します。

CHAPTER 21 イベント処理およびアラート

イベント処理機能を使用すると、管理者は、特定のイベントが発生した場合に、トラフィックマネージャにどのようなアクションを取らせるかを正確に設定することができます。イベント処理の動作は、管理インターフェースの **System** (システム) セクションの **Alerting** (アラート) タブで設定できます。

概要

イベントハンドラは、特定タイプのイベントが発生したときに実行するアクションを指定します。イベントハンドラは、**Alerting** (アラート) ページで設定します。

Event Type	Actions
All Events	Log to event log
License Key Problem	E-Mail Administrator

Select event type...

Manage Event Types Manage Actions

Apply Changes

Update

Fig 61. イベントハンドラの設定

トラフィックマネージャには、全てのイベントをグローバルイベントログに書き込むイベントハンドラが内蔵されています。この内蔵ハンドラとその **Log all Events** (全てのイベントを記録する) アクションは削除できませんが、必要に応じてアクションを追加することができます。

ライセンスキーに問題がある場合に電子メールを送信する上図のハンドラなど、イベントハンドラを追加することができます。イベントハンドラを追加するには、まずドロップダウンボックスからイベントタイプを選択し、次にそのタイプのイベントが発生した場合に実行するアクションを選択します。

注: 追加のイベントハンドラを作成するときに、**Bypass event log** (イベントログのバイパス) 内部アクションを追加できます。そのイベントハンドラで処理される全てのイベントは、グローバルイベントログに記録されません。

Fig 62. Connection Failures (接続失敗) イベントタイプに一致するイベントは Syslog アクションで処理され、トラフィックマネージャイベントログに記録されません

イベントタイプ

トラフィックマネージャは、ノード障害、トラフィック IP アドレス転送、サービスレベルの不適合、ルールから報告されたカスタムイベントなど多様なイベントに対処することができます。便宜上、各イベントは、あらかじめ定義された多数のタイプに分類されています。また、必要に応じて新しいイベントタイプを作成することも可能です。

イベントタイプ	説明
All Events (全てのイベント)	トラフィックマネージャが検出可能な全てのイベントが含まれます。このタイプに一致する全てのイベントは、内蔵ハンドラによりグローバルイベントログに記録されます。
Critical Problem Occurred (重大な問題発生)	内部障害、ノード障害、IP 転送(トラフィックマネージャの障害による)、その他のハードウェアの問題など重大な問題を示す全てのイベントが含まれます。
Critical Problem is Resolved (重大な問題が解決)	障害が発生したノードが再び動作開始したなど、Critical Problem Occurred (重大な問題発生) イベントタイプで報告された問題が解決されたことを示すイベントが含まれます。 問題発生時に電子メールメッセージを送信するようトラフィックマネージャを設定した場合、問題が解決して介入が不要になったときにも電子メールメッセージを送信するようトラフィックマネージャを設定することをお奨めします。

イベントタイプ	説明
Default Events (デフォルトイベント)	旧バージョンのトラフィックマネージャとの互換性を確保するためのイベントです。トラフィックマネージャによる電子メールメッセージの送信や SNMP トラップの報告を伴う全てのイベントが含まれます。 トラフィックマネージャを旧バージョンからアップグレードすると、アップグレードプロセスにより、必要に応じて旧バージョンの動作を維持するためのイベントハンドラが追加されます。
Infrastructure Problem (インフラストラクチャの問題)	これらのイベントは、サーバと通信できなくなるなど、トラフィックマネージャがインフラストラクチャの問題を検出した場合に報告されます。
Service Problem (サービスの問題)	これらのイベントは、サービスの正常な提供に影響を与える可能性のある問題が検出された場合に報告されます。

イベントタイプを表示するには、**Alerting (アラート)**ページの **Manage Event Types** (イベントタイプの管理) ボタンを使用します。

新しいイベントタイプの作成

内蔵イベントタイプを変更することはできませんが、コピーを作成して編集したり、新しいイベントタイプを最初から作成したりできます。

例えば、トラフィックマネージャでいくつかのサービスを管理しており、特定のサービスで障害が発生した場合に一致するイベントタイプを作成するとします。これにより、そのイベントタイプを使用して、そのサービスの担当者に電子メールを送信するイベントハンドラを作成できるようになります。

これを行うには、**Service Failed (サービス失敗)** イベントタイプのコピーを作成し、特定の仮想サーバ、プール、およびノードからイベントが報告された場合にのみ一致するようこのコピーを編集します。

1. **Manage Event Types (イベントタイプの管理)** をクリックし、次に「**Service Failed (サービス失敗)**」 イベントタイプをクリックします。
2. このイベントタイプは変更できませんが、コピーを作成することはできます。**Custom Service Failed (カスタムサービス失敗)** というイベントタイプにコピーします。

3. 新しいイベントタイプの **Edit (編集)** ページが表示されます。イベントはツリー構造に整理されます。ツリーのノードを展開すると、トラフィックマネージャが検出可能な全てのイベントが表示されます。イベントタイプごとに一致するイベントが表示されるため、イベントを簡単に確認できます。
4. 新しいイベントタイプを編集します。特定のサービスに関連するオブジェクトからイベントが報告された場合のみイベントタイプが一致するよう、イベントを報告するヘルスマニタ、ノード、プール、ルール、仮想サーバ、およびその他のオブジェクトのリストを変更できます。
5. 変更内容を適用し、**Custom Service Failed (カスタムサービス失敗)** イベントタイプに適したイベントハンドラを作成します。

アクション

イベントハンドラが実行されると、トラフィックマネージャは、そのハンドラに設定されているアクションを呼び出します。

注: トラフィックマネージャには、E-Mail (E メール)、SNMP Trap (SNMP トラップ)、Syslog の 3 つのアクションが内蔵されています。トラフィックマネージャを新たにインストールした時点では、E-Mail (E メール) および SNMP Trap (SNMP トラップ) アクションは完全には設定されていません(編集して必要な設定を指定する必要があります)。ただし、トラフィックマネージャを旧バージョンからアップグレードした場合は、旧バージョンでの設定が引き継がれます。

以下のタイプの追加アクションを作成できます。

タイプ	説明
E-Mail (E メール)	このアクションは、ハンドラの契機となったイベントのレポートを、アクションに設定されている電子メールアドレスに送信します。 それぞれ異なる電子メールアドレスを設定した複数の E-Mail (E メール) アクションを作成し、イベントタイプごとに異なる受信者に電子メールを送信するようトラフィックマネージャを設定することができます。

タイプ	説明
Log to external file (外部ファイルへのロギング)	イベントは常にグローバルイベントログに記録されますが、このアクションでは、さらに指定したログファイルにイベントを記録することができます。 使用フォーマットは、グローバルイベントログのフォーマットと同じです。このアクションは、特定イベントの専用ログを作成する場合に便利です。 注: トラフィックマネージャアプライアンス/クラウドのユーザ: この機能を仮想インスタンスまたはクラウドインスタンスで使用する場合は、ファイルを ZEUSHOME 内に保存してください。ユーザが ZEUSHOME 外に作成した全てのファイルおよびディレクトリは、アップグレードを適用すると失われる可能性があります。
Log to syslog (syslog へのロギング)	このアクションは、指定したサーバの syslog デーモンにメッセージを送信します。 このアクションを使用すると、syslog メッセージを受け入れ可能な外部のモニタリングおよび管理ツールに特定のイベントを送信できます。
Program (プログラム)	このアクションは、アクションの Edit (編集) ページの Upload and Manage Programs (プログラムのアップロードと管理) ボタンを使用してトラフィックマネージャシステムにアップロードされたスクリプトまたはその他のプログラムを実行します。プログラム例については、CHAPTER 21 「プログラムやスクリプトの呼び出し」 セクションを参照してください。

タイプ	説明
SNMP Notify/Trap (SNMP 通知/トラップ)	このアクションは、指定されたホストの指定の SNMP レシーバアプリケーションに SNMP 通知またはトラップを送信します。トラップはトラフィックマネージャ MIB で定義されており、アクションの Edit (編集) ページからダウンロードできます。 トラフィックマネージャは、SNMP v1、v2c、および v3 をサポートしています。このアクションに対して指定できる設定は、Edit Action (アクションの編集) ページで指定したバージョンによって異なり、System (システム) > SNMP ページで利用できる設定(SNMP コマンドレスポンドオプション)と全く同じです。詳細については、CHAPTER 7 の「SNMP による性能のモニタリング」セクションを参照してください。
SOAP Callback (SOAP コールバック)	このアクションは、指定サーバに SOAP メッセージを送信します。 SOAP アクションとそれを処理するサーバの例については、CHAPTER 21 の「SOAP メッセージの送信」セクションを参照してください。

アクションを編集したり、新しいアクションを作成したりするには、**Alerting (アラート)** ページの **Manage Actions (アクション管理)** リンクをクリックします。

「Bypass event log (イベントログのバイパス)」内部アクションを適用して、イベントがグローバルイベントログに記録されないようにすることもできます。

アクションのテスト

アクションを作成または編集したら、「テスト」イベントを使用してそのアクションをすぐに呼び出すことができます。これにより、アクションの設定や動作が正しいかどうかをテストすることができます。

イベントハンドラの設定

イベントハンドラを設定する前に、そのイベントハンドラの契機となるイベントに一致するイベントタイプを作成し、電子メールの送信など適切なタスクを実行するアクションを作成する必要があります。

イベントハンドラにより、イベントタイプと、イベントタイプに一致するイベントが報告された場合に呼び出す 1 つ以上のアクションが紐付けられます。

イベントハンドラを設定するには、**System** (システム) > **Alerting** (アラート) ページに移動します。ドロップダウンリストからイベントタイプを選択して新しいイベントハンドラを作成し、呼び出すアクションをそのイベントハンドラのエントリにあるドロップダウンリストから選択します。

変更を完了したら、「**Update** (更新)」をクリックします。

重複イベント

1つのイベントが複数のイベントタイプに一致し、これによって複数のイベントハンドラが実行される場合があります。これらの各ハンドラに同じアクションが設定されている場合、そのアクションが複数回呼び出されることになります。

送信される小さい電子メールの量を減らすために、**E-Mail** (E メール) アクションはすぐには実行されず、電子メールメッセージをまとめて 30 秒間隔で送信します。**E-Mail** (E メール) アクションのこの動作は、**System** (システム) > **Global Settings** (グローバル設定) ページの **Other Settings** (その他の設定) セクションで設定できます。

重要性の低いイベントが短時間に連続して報告された場合、トラフィックマネージャはイベントの反復を抑制します。深刻なイベントがこのように抑制されることはありません。

カスタムアクション

プログラムやスクリプトの呼び出し

Program (プログラム) アクションは、トラフィックマネージャにアップロードしたプログラムまたはスクリプトを実行するために使用します。プログラムやスクリプトは、**Catalogs** (カタログ) > **Extra Files** (追加ファイル) > **Action Programs** (アクションプログラム) ページを使用してアップロードします。

このアクションが呼び出されると、トラフィックマネージャは、そのアクションの契機となったイベントを識別するコマンドライン引数を渡してプログラムをバックグラウンドで実行します。

トラフィックマネージャがプログラムまたはスクリプトに渡す引数は以下のとおりです。

- アクションに設定したコマンドライン引数(オプション)
- プログラムの実行元アクションを呼び出したイベントハンドラの名前が指定された `eventtype` という引数
- イベントの記述: フォーマットはイベントログに書き込まれるイベントのログ行と全く同じ

例えば、TrafficScript ルールにより TestEvent というイベントが報告され、このイベントが、Program (プログラム)アクションを呼び出す「My Event Handler」というイベントハンドラに捕捉された場合、プログラムは以下のように呼び出されます。

```
programname "--eventtype=My Event Handler" \  
  "INFO trafficscript/TestEvent CustomEvent \  
rules/<rulename> \  
  vservers/<vservename> <description>"
```

このスクリプトには2つの引数を与えられていることに注意してください(分かりやすくするため改行を挿入)。

SOAP メッセージの送信

SOAP アクションは、アクションに設定されたプロキシに SOAP リクエストを送信します。SOAP メッセージは、以下の WSDL 仕様に適合しています。

ZEUSHOME/zxtm/etc/wsd1/AlertCallback.wsdl

このファイルは、SOAP アクションを編集するときに **Actions (アクション) > Edit (編集)** ページからダウンロードできます。この仕様については、トラフィックマネージャ Control API のドキュメントに記載されています。

SOAP リクエストは、インターフェース

`http://soap.zeus.com/zxtm/1.0/AlertCallback` を使用し、以下の引数を渡して `eventOccurred` メソッドを呼び出します。

- イベントを報告したトラフィックマネージャの名前(タイプ `xsd:string`)。
- `xsd:dateTime` フォーマット¹¹のイベント発生時間。
- イベントの重大度(列挙タイプ `AlertCallback.Severity`)。
- イベントを識別するタグ(列挙タイプ `AlertCallback.Tag`)。
- タグの配列。現在は空ですが、将来使用するために予約されています。
- オブジェクトの配列(タイプ `AlertCallback.ObjectArray`)。これらのオブジェクト(タイプ `AlertCallback.Object`)により、イベントの報告元のオブジェクト(ノード、ルールなど)が記述されます。

一部のイベントは複数のオブジェクトを参照します。例えば、TrafficScript ルールから報告されるイベントには、それぞれイベント名、ルール、およびルールを実行した仮想サーバを識別する3つのオブジェクトが含まれます。

¹¹ <http://www.w3.org/TR/xmlschema-2/#dateTime>

- 人間が読み取ることのできるイベントの記述(タイプ `xsd:string`)。これは、多くのイベントにあらかじめ定義されていますが、イベントが **TrafficScript** 関数の `event.emit()` によって報告されるときは、**TrafficScript** が提供するメッセージが記述に含まれます。
- アクションによって提供される追加データ(タイプ `xsd:string`)
- アクションを呼び出したイベントハンドラの名前

以下の Perl CGI スクリプトは、**AlertCallback** メッセージの単純な SOAP コンシューマを実装します。このレスポンドは、メッセージの詳細をローカルファイルに記録します。この CGI スクリプトを適切な Web サーバ(Perl および SOAP::Lite Perl モジュール搭載)にインストールし、プロキシ値を CGI スクリプトの URL に設定して SOAP アクションを設定することができます。

```
#!/usr/bin/perl

package AlertCallback;

sub _addSerializerForEnum($;$)
{
    my ( $xsdtype, $ns_prefix ) = @_;
    $ns_prefix = "zeusns" unless $ns_prefix;

    $func = "SOAP::Deserializer::as_${xsdtype}";
    *$func = sub { return $_[1] };
}

BEGIN {
    _addSerializerForEnum( "AlertCallback.Tag" );
    _addSerializerForEnum( "AlertCallback.ObjectType" );
    _addSerializerForEnum( "AlertCallback.Severity" );
}

sub eventOccurred($$$$$$$)
{
    my $self = shift;
    my( $stm, $time, $severity, $tag, $tags, $objects,
        $desc, $additional, $handler ) = @_;

    open LOG, ">>events.log" or die( "Couldn't open log
events.log: $!" );

    print LOG "$stm - $time - $severity: $tag [";
```

```
    print LOG join ', ', map { "$_->{type}:$_->{name}" }
    @$objects;
    print LOG "]" $desc ($additional); Event handler:
    $handler\n";

    close LOG;
}

package main;

use SOAP::Transport::HTTP;

SOAP::Transport::HTTP::CGI
-> dispatch_with(
    { "http://soap.zeus.com/zxtm/1.0/AlertCallback/" =>
    'AlertCallback' } )
-> handle;
```

アクションが複数のイベントによって呼び出される場合は、SOAP メッセージ内のタグ値に基づいて正確なイベントを特定することが望ましい場合があります。これを行うには、イベントツリー(カスタムイベントタイプを設定したときのツリー)内のイベントの記述と WSDL ファイル内の記述を照合して、AlertCallback.Tag の値を特定します。

TrafficScript または Java Extensions からのイベント報告

イベントは、TrafficScript ルールまたは Java Extension から報告することができます。この場合、適切なイベントハンドラは非同期にバックグラウンドで実行されるため、TrafficScript ルールや Java Extension がブロックされることはありません。

イベントは、event.emit(event_name, event_message) 関数(TrafficScript) または emitEvent(event_name, event_message) 関数(Java)を使用して報告します。

event.emit() または emitEvent() によって報告されたイベントに基づいてアクションを呼び出すには、Custom Events(カスタムイベント)を含むイベントタイプを作成する必要があります。全てのカスタムイベントに一致させることも、特定のイベントにのみ一致させることもできます。

▼ Basic Settings

These settings allow you to specify which events make up this event type.

Name:

Events: **Events** Unfold All / Fold All

- ☐ Cloud Credentials
- ☐ Configuration Files
- ☐ Fault Tolerance
- ☐ General
- ☐ GLB Services
- ☐ Java
- ☐ License Keys
- ☐ Locations
- ☐ Monitors
- ☐ Pools
- ☐ Rules
- ☐ Service Protection Classes
- ☐ SLM Classes
- ☐ SSL Hardware
- ☐ Traffic Managers
- ☐ Virtual Servers
- ☒ Custom Events

Custom events can be triggered using the TrafficScript rule 'event.emit'

☐ No custom events
☐ All custom events
☒ Some custom events...

Add custom event

Fig 63. TrafficScript イベント TestEvent1'およびTestEvent2'に一致するイベントタイプ

イベントタイプの特定イベントの名前は、そのイベントが TrafficScript ルールで報告されるときに使用される名前と一致している必要があります。

```
# raise an event named "TestEvent1", to be caught by an
Event
# Type containing a custom event called "TestEvent1".

event.emit( "TestEvent1", "The request was ".$url );
```

例

単純な侵入検知システムは、特権 URL へのアクセスを試行する全てのリモートクライアントを禁止しようとする可能性があります。

```
# A TrafficScript rule that detects unauthorized
# accesses and raises an event.

$path = http.getPath();

if( string.StartsWith( $path, "/secure/admin" ) ) {
```

```
$ip = request.getRemoteIP();  
if( !string.ipMaskMatch( $ip, "10.0.0.0/8" ) ) {  
    event.emit( "UnauthorizedIP", "IP: ".$ip );  
}  
}
```

「UnauthorizedIP」というカスタムイベントを含むイベントタイプと、そのイベントタイプに基づくイベントハンドラを作成する必要があります。

このイベントハンドラから、侵入検知ルールを更新してその IP からの接続を禁止するカスタムプログラムアクションを呼び出すことができます。

```
#!/usr/bin/perl -w  
  
# An Action script to process event messages raised by  
# the above TrafficScript rule.  
  
my $logLine = $ARGV[1] or exit;  
  
$logLine =~ /IP:([\d]+\.[\d]+\.[\d]+\.[\d]+)/;  
my $ip = $1;  
  
open OUT, ">>/tmp/events.log" or die $!;  
print OUT "The bad IP was $ip\n";  
close OUT;  
  
# reconfigure upstream firewall, or the  
# Traffic Manager's banned IP list to ban access  
# from $ip
```

CHAPTER 22 システムレベルの設定

注: 本章の内容は、一部のトラフィックマネージャ製品のみに該当します。詳細については、ご使用の製品仕様を参照してください。

トラフィックマネージャのハードウェアアプライアンス、仮想アプライアンス、クラウドインスタンス、および一部のソフトウェア製品には、ネットワーク設定や時間および日付などの設定を管理するための追加の設定ページが多数あります。それらの製品では、通常は OS レベルで処理されるような、OS の様々なコンポーネントおよび機能の完全性チェックなど、数多くの追加的なシステムチェックおよびテストをトラフィックマネージャの管理インターフェースから実行できます。

トラフィックマネージャを工場出荷時の初期設定にリセットする詳細な方法など、初期のセットアップおよび設定方法については、ご使用の製品の『**Installation and Getting Started Guide**』をご覧ください。

ネットワーク設定

注: このセクションで説明する機能は、ご使用の製品によって異なる場合があります。詳細については、ご使用の製品仕様を参照してください。

トラフィックマネージャインスタンスには多数のネットワークインターフェースが搭載されており、そのインスタンスを識別するためにホスト名が必要になります。管理インターフェースの **System** (システム) セクションの **Networking** (ネットワーキング) ページでは、これらの設定と DNS、ルーティング、IP フォワーディング、および NAT を設定することができます。

注: このページで行った設定は、管理 UI にアクセスしているローカルのトラフィックマネージャのインスタンスにのみ適用されます。ただし、**Return Path Routing** (リターンパスのルーティング) は例外で、ご使用のトラフィックマネージャクラスタ全体に適用されます。クラスタに含まれる他のトラフィックマネージャのネットワーク設定をチェックまたは設定するには、そのインスタンスの管理 UI を使用する必要があります。

ホスト名と IP アドレスの設定

ホスト名により、クラスタ内でトラフィックマネージャシステムがユニークに識別されます。また、IP アドレスにより、システム上のアクティブな各ネットワークインターフェースが識別されます。管理ポートを使用しない場合、選択するホスト名がトラフィックマネージャシステムの永続的 IP アドレスに名前解決されなければならないため、それに応じてローカル DNS を設定する必要があります。

一般に、最初のネットワークインターフェース(eth0)は管理ポートとして指定され、指定するホスト名はそのマシンの IP アドレスに名前解決される必要はありません。トラフィックマネージャの管理 IP では管理情報のみの受信および送信が行われるため、内部または外部の公開トラフィックをルーティングするネットワークから切り離されたセキュアなネットワークに接続する必要があります。外部または内部トラフィックの管理には、その他のネットワークインターフェースを使用します。

管理ポートを使用しない場合、トラフィックマネージャは管理トラフィックを全てのインターフェースで受信するため、指定するホスト名は、トラフィックマネージャシステムでアクティブないずれかの IP アドレスに名前解決されなければなりません。

ご使用のトラフィックマネージャ製品のインターフェースとその番号については、『**Installation and Getting Started Guide**』に記載されています。

ネットワークキングの設定

デフォルトでは、トラフィックマネージャは、各インターフェースの接続先スイッチとの間でネットワーク設定について自動的にネゴシエーションを行います。

場合によっては、インターフェース設定を手動で指定する必要があります。例えば、多数のスイッチが、サポートしている最高転送速度のネゴシエーションに失敗することがあります。1Gb の転送速度は自動的なネゴシエーションによって決定され、強制的に使用することはできません。

トランキング/ボンディング

同じ IP アドレスを複数の物理ネットワークインターフェースに設定すると、トラフィックマネージャは、これらのインターフェースをトランクのメンバにします(仮想アプライアンスのみ)。これが機能するためには、スイッチが IEEE 802.3ad をサポートしていなければならない、その再設定が必要になることがあります。この機能は、スイッチによってはリンク集合、ボンディング、またはトランキングと呼ばれます。

VLAN の設定

注: Microsoft Hyper-V ハイパーバイザで実行されている仮想アプライアンスに対して、トラフィックマネージャの管理 UI から VLAN を設定することはできません。そのような環境で VLAN を有効化する必要がある場合は、ご使用の Windows Server ワークステーションの Hyper-V マネージャアプリケーションから直接 VLAN を設定してください。詳細については、Hyper-V のドキュメントを参照するか、サポート窓口にお問い合わせください。

トラフィックマネージャ管理 UI を使用して、トラフィックマネージャの任意のネットワークインターフェースまたはボンディング/トランキングされたインターフェースに VLAN を設定することができます。この設定は、**System (システム) > Networking (ネットワーキング)** ページで行います。

1. 仮想 LAN インターフェースを適切な物理インターフェースに追加します。

Virtual LANs

Add Virtual LAN interfaces on top of your existing physical interfaces.

VLAN ID	Physical Interface	Remove
7	eth1	☐

Add New VLAN Interface:

Interface: eth0 ▼

VLAN ID:

Fig 64. Virtual LAN 設定

2. 新しい VLAN にネットワーキング(IP アドレスとサブネット)を設定します。

Interface	IP Address	Netmask	Remove
eth0	10.100.9.180	255.255.0.0	☐
	fd6e:9138:1b6c:6401:21e:c9ff:fe30:657d	64	☐
eth1	10.100.12.180	255.255.255.0	☐
VLAN 7 (eth1)	10.100.14.180	255.255.255.0	☐

Add IP Address:

Interface: eth0 ▼

IP Address: eth0
eth1
eth2
eth3
eth4
VLAN 7 (eth1)

Netmask:

Virtual LANs

Add Virtual LAN interfaces on top of your existing physical interfaces.

Fig 65. Virtual LAN Network 設定

VLAN インターフェースを通してルーティングされるトラフィックには、VLAN ID のタグが付けられます。

DNS の設定

トラフィックマネージャは、ノード名を IP アドレスに名前解決するときや外部サービスと通信するとき、また受信したネットワークコネクションの IP アドレスを名前解決することがアクセス制限やその他の設定によって求められるときに、クラスター間通信用のローカル DNS サーバにクエリします。

トラフィックマネージャシステムに、1 台以上のローカル DNS サーバの IP アドレスを設定する必要があります。通常、この IP アドレスは、トラフィックマネージャを最初にインストールするときに設定されますが、管理サーバの **Networking (ネットワーキング)** ページの **DNS** 部を使用して、この設定を編集したり、ホストと IP アドレス間の固定マッピングを追加したりできます。

ルーティングの設定

トラフィックマネージャは、ネットワークインターフェースに割り当てられた IP アドレスに従って、最適なネットワークインターフェースでトラフィックをルーティングします。トラフィックマネージャが外部ネットワークトラフィックを転送するためのデフォルトゲートウェイも設定する必要があります。

重要: デフォルトゲートウェイは、CHAPTER 6 の「フォールトトレランスの設定」セクションで説明しているようにネットワークコネクションテストで使用し、外部(受信)トラフィックに使用されるネットワーク上に存在している必要があります。管理ネットワーク上に存在するゲートウェイを指定しないでください。または、より適切な IP アドレスを **flipper!frontend_check_addrs** に設定してください。

デフォルトゲートウェイが適切でない場合は、必要に応じて、外部ネットワークへの追加ルートを手動で追加できます。この場合、外部ネットワーク(宛先およびネットマスク)、そのネットワークへのローカルゲートウェイ(オプション)、および外部ネットワークにアクセスするためのインターフェースを指定する必要があります。

これらの設定は、管理サーバの **Networking** (ネットワーキング)設定ページの **Routing** (ルーティング)部で管理します。

リターンパスルーティングの設定

注: このセクションの設定は、単体のトラフィックマネージャに特有ではなく、クラスタ内のその他全てのトラフィックマネージャのインスタンスに反復されます。

通常、どのルートデータパケットがネットワークを出てから宛先に到達するのかについて、それらが着信する限りは気にすることはありません。

従来のルーティングアルゴリズムは、データパケットの宛先アドレスを確認し、その宛先まで最もコストの低いパスであると考えられるルータにパケットを送信します。トラフィックマネージャとインターネットの間に複数のルートが存在する場合、クライアントからサーバへのルートがサーバからクライアントへのルートとは異なることがあります。通常、どちらのルートも期待される通りのパケットを受信している限り、この状態は問題になりません。

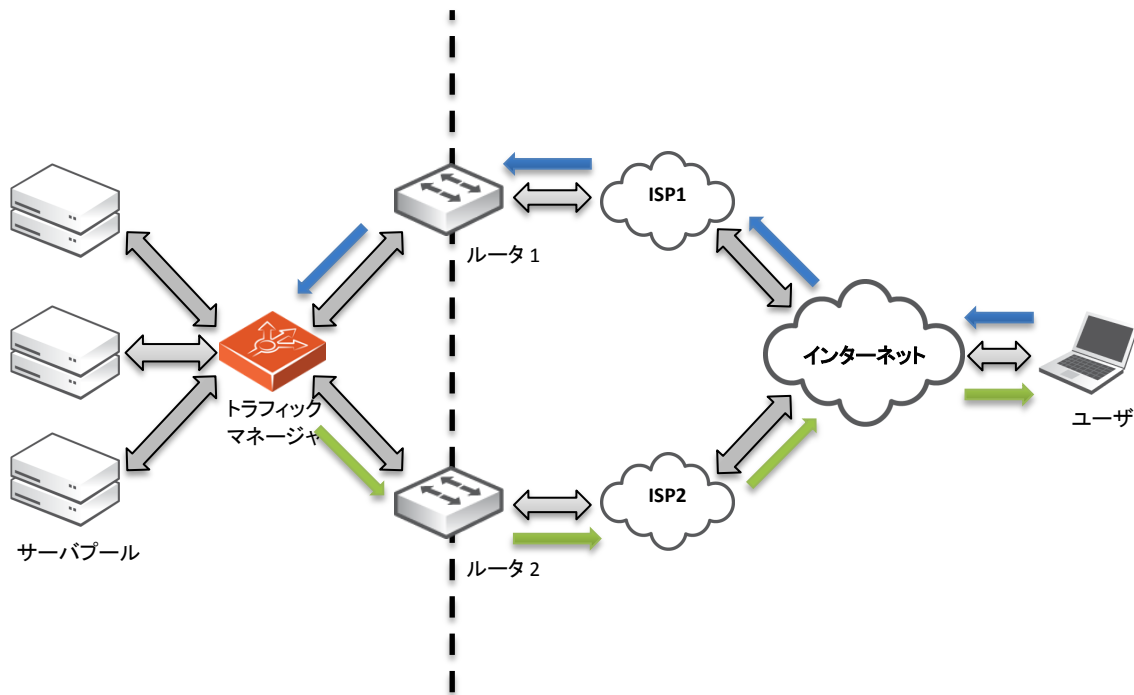


Fig 66. 典型的な最低コストのルーティング

しかし、外部からの受信パスに対応するために、外部への送信パスが必要になることがあります。例えば、パス内にステートフルなファイアウォールまたは NAT ルータがある場合、そのデバイスは両方向でパケットを処理する必要があります。さもないと、デバイスが把握しているコネクションに属さないパケットは破棄されてしまいます。その代わりに、2つの ISP を使用して負荷分散することができます(冗長性に加えて)。サーバからクライアントへの全てのトラフィックが同じ ISP を介して流れる場合、負荷の不均衡を引き起こす可能性があります。

この問題に対するソリューションの1つに、トラフィックマネージャ用のデフォルトゲートウェイとしてゲートウェイルータを設置することが挙げられます。ゲートウェイルータは、それ自体のコネクション追跡を使用して、適切な ISP にパケットを転送します。

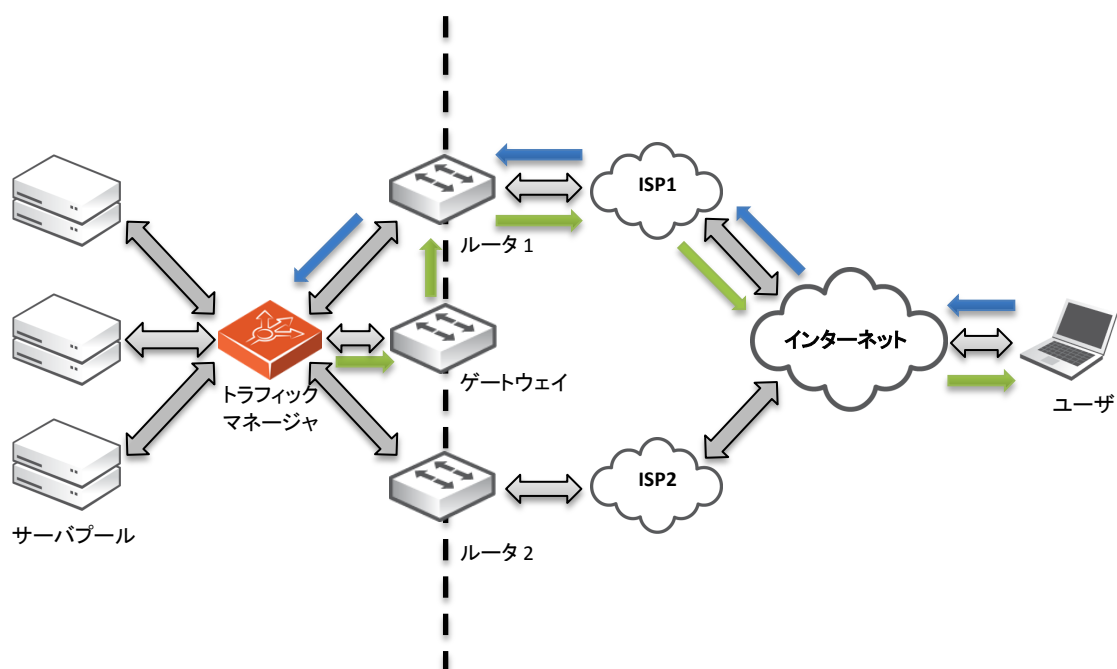


Fig 67. コネクション追跡を目的としたゲートウェイルータの採用

ネットワークインフラストラクチャの都合でこのソリューションを利用できない場合は、トラフィックマネージャアプライアンスのリターンパスルーティング機能を使用すると、適切なリターンルートの提供がスムーズに行われます。

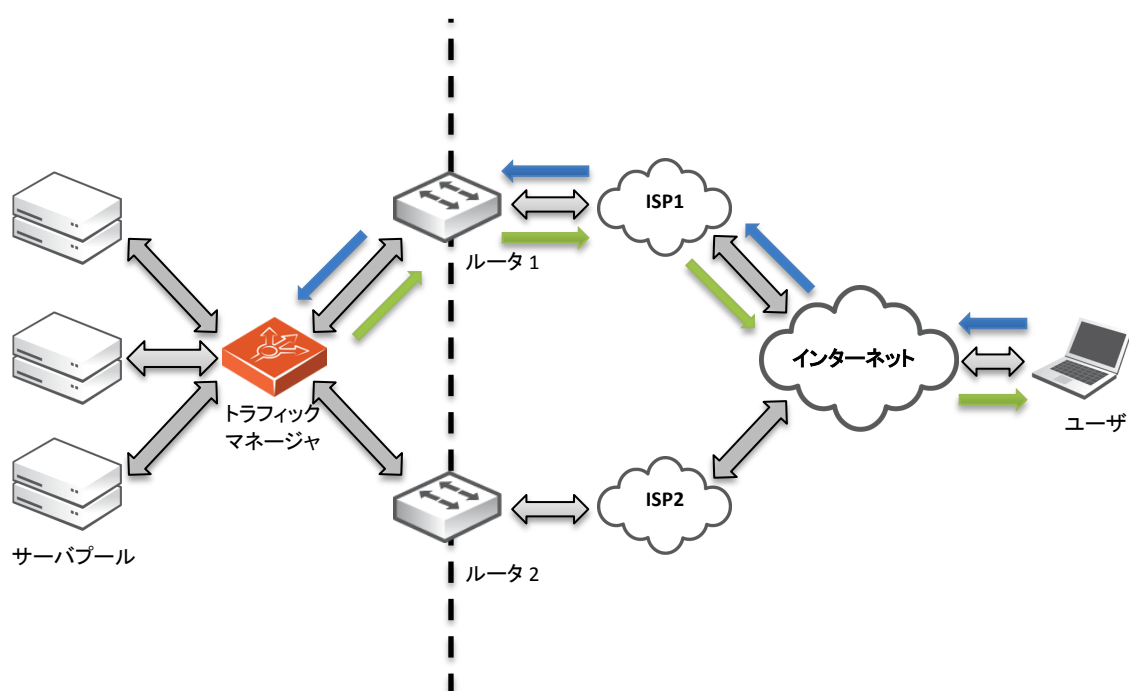


Fig 68. リターンパスルーティングが有効化されている

注: トラフィックマネージャは、最初のルータホップを介してのみリターンパスルーティングを戻すことができます。ネットワークが、必要なオンワードルーティングをクライアントに戻すことができるかどうかについては、お客様の責任においてご確認ください。

トラフィックマネージャは、所要のルータを介してリターンパケットをルーティングするために、「netfilter」のコンネクション追跡機能を使用します。トラフィックマネージャは新しいコンネクションを受け取ると、イーサネットパケットヘッダの MAC アドレスを使用して、どのルータを介してクライアントが接続されているかを特定します。トラフィックマネージャはこのアドレスを使用して、そのコンネクションに属する全てのパケットにトラッキングマークでラベル付けします。

トラフィックマネージャがクライアントにレスポンスパケットを送信する際、トラッキングマークの存在によって IP フォワーディングのルールが呼び出され、デフォルトゲートウェイや静的ルートではなく、適切なルータにパケットが送信されます。

トラフィックマネージャは、外部に送信される以下のデータパケットに対してリターンパスルーティングを適用します。

- クライアントによって開始された TCP コネクションのパケット
- FTP などの特定のプロトコルに対してクライアントが開始した TCP コネクションについては、トラフィックマネージャが開始した予備 TCP コネクションのパケット
- クライアントからの UDP リクエストに関連する UDP レスポンス

注: 全ての FTP コネクションのリターンパスルーティングを適切に確保するには、トラフィックを処理する仮想サーバを、デフォルトの FTP ポート(21)を監視するように設定してください。

以下は適用対象外です。

- トラフィックマネージャが開始したその他の TCP コネクション
- その他の UDP パケット
- SIP (UDP)または SIP (TCP)プロトコルで動作する仮想サーバが処理する全てのデータ

重要: コネクション追跡には、非常に大きな性能オーバーヘッドがかかります。そのため、リターンパスルーティングは、ネットワークインフラストラクチャにとって必要なときのみ有効化してください。

この機能を使用するには、リターンパスルーティング用に使用したいネットワークインフラストラクチャ内のルータを指定します。それぞれのルータに対しては、IPv4 アドレス(および該当する場合は IPv6 アドレス)と、その IP が有効になるインターフェースの MAC アドレスを特定します。ご使用のルータの設定インターフェ

ースに、この情報が表示されることがあります。または、「ping」、「arp」、「tcpdump」などのネットワークツールを使用します。

この情報を、トラフィックマネージャ管理 UI の **System (システム) > Networking (ネットワーキング)** ページの *Return Path Routing* (リターンパスルーティング) セクションの設定テーブルに追加します。このテーブルは、クラスタ内の全てのトラフィックマネージャインスタンスに反映されます。

リターンパスルーティングを有効化するには、**Yes (はい)** ラジオボタンをクリックしてから、**Update (更新)** をクリックします。

注: バリデーションまたは通信の問題によるエラーや警告がイベントログに記録されることがあります。しかし、トラフィックマネージャは、データパケットが期待されるリターンルートを介して受信されたかどうかを初期のルータコネクションを超えてまで検出することはできません。要件に従ってパケットが受信されていることを確認するためには、補足的なネットワークテストを実施することをお奨めします。

IP フォワーディングおよびネットワークアドレス変換(NAT)の設定

ネットワークトポロジによっては、トラフィックマネージャがネットワーク上のバックエンドノードまたはその他のデバイスのネットワークゲートウェイとして機能することがあります。例えば、IP トランスペアレンシを使用する場合、トラフィックマネージャを介して外部へと送信されるトラフィックをルーティングするようバックエンドノードを設定する必要があります(CHAPTER 2 の「ルーティング設定」を参照)。

トラフィックマネージャ(またはその他のネットワークデバイス)はゲートウェイとして機能する場合に、トラフィックを送信元ノードから宛先へと転送し、レスポンスを送信元ノードへと転送して返します。

トラフィックマネージャが転送する全てのトラフィックが、パブリックにルーティング可能な IP アドレスを持つソースから転送された場合、複雑な事態は発生しません。送信元から宛先までのリンクにある全てのホップが、レスポンスを送信元 IP アドレスまでどのようにルーティングして返すかを認識しています。

ただし、バックエンドノードは、パブリックインターネットからルーティングできないプライベートネットワーク上に配置されるのが一般的です。このような場合、外部のインターネットクライアントはネットワークトラフィックをノードに直接送信することができず、パブリック IP アドレスを節約したり、セキュリティのレベルを高めたりすることができません。ゲートウェイデバイスは NAT という手法を使用して、ルーティングできない送信元 IP アドレスをパブリックにルーティング可能なゲートウェイの IP アドレスに透過的に置き換えます。

ネットワークトポグラフィによっては、ダウンストリームのルータに NAT を実行させることによって、全ての内部システムが、トラフィックマネージャを介した内部トラフィックがどのように適切にルーティングされているのかについて把握してい

ることがあります。しかし、これに該当しない場合、転送パケットに対して NAT を実行するようトラフィックマネージャを設定することにより、レスポンスがトラフィックマネージャゲートウェイを経由してルートバックされるようにできます。

注: トラフィックマネージャの NAT 機能は、IPv4 でのみ利用できます。

NAT を設定するには、管理 UI の **System** (システム) > **Networking** (ネットワーク) ページの NAT セクションにある **Manage NAT Settings** (NAT 設定の管理) リンクをクリックします。このページは、トラフィックマネージャ上での NAT の動作を管理するルールを定義するために使用します。

トラフィックマネージャを内部ネットワークトラフィック用の単純な IPv4 または IPv6 ゲートウェイとして使用する場合など、NAT を設定せずに IP フォワーディングを使用したいことがあります。そのような場合は、管理 UI の **System** (システム) > **Networking** (ネットワーク) ページの NAT セクションにある **Enable IPv4 Forwarding** (IPv4 転送の有効化) または **Enable IPv6 Forwarding** (IPv6 転送の有効化) を選択します。

注: 1 つ以上の IP マッピング NAT ルールを有効化している場合や、トラフィックマネージャの任意のインターフェースを選択して NAT を実行している場合、デフォルトで IPv4 転送は有効化されます。したがって、**Enable IPv4 Forwarding** (IPv4 転送の有効化) コントロールは無効化されます。このような状況でも、IPv6 転送は影響を受けません。

IP マッピングルール

トラフィックマネージャは、3 つのタイプの外部送信 IP NAT ルールをサポートしています。それぞれのルールは、そのルールが処理するトラフィックのソース IP アドレスを、指定したトラフィック IP グループのプライマリ IP アドレスに置き換えます。

これらのルールは、以下の優先順位で適用されます。

ルールのタイプ	説明
Many to One: Port Locked	NAT は、指定したプロトコルを使用するとともに、指定した宛先ポートをターゲットとすることにより、指定したプールのメンバからのトラフィックに適用されます。 このタイプのルールに有効なプロトコルは、icmp、sctp、tcp、udp、および udplite です。

ルールのタイプ	説明
One to One	NAT は、単一の指定された IP アドレスからのトラフィックに適用されます。 Add Inbound Rule? (外部からの受信ルールを追加)を選択すると、このルールが双方向に変わり、選択されたトラフィック IP グループのプライマリ IP アドレスにアドレス指定されたトラフィックが、指定された IP アドレスにも転送されます。この場合は、ソース IP アドレスは変更されません。
Many to One: All Ports	このルールは「Many to One: Port Locked」のルールと同様に動作します。しかし、ターゲットのポート番号にかかわらず、指定されたプールに対する全ての外部送信トラフィックを対象にする点で異なります。そのため、あるプールから開始され、「Many to One: Port Locked」のルールでは対象とされない任意のポート番号へのトラフィックに対するフォールバックルールとしてこれらのルールを使用することができます。

ポートマッピングルール

トラフィックマネージャは、以下の外部から受信する NAT ルールを仮想サーバでサポートしています。

ルールのタイプ	説明
Virtual Server Port Forwarding	指定されたポート範囲をターゲットとする受信トラフィックの宛先ポートが、選択された仮想サーバが監視するポートに置き換えられます。これにより、仮想サーバにより監視されるポートが 1 つではなく、一定範囲のポートが監視されるようになります。 注: ポート転送は、外部へと送信される IP NAT ルールとは相互に影響し合いません。

インターフェースに対するデフォルトルール

このセクションは、内部ソースに代えて転送されるネットワークトラフィックに対して NAT を実行する、トラフィックマネージャインターフェース全体を選択するために使用します。一般に、これらのインターフェースはパブリックにルーティング可能な IP アドレスを持ち、ネットワークの外側に位置します(外部に送信されるトラフィックの場合)。

注: プライベート IP アドレスとパブリック IP アドレスが、NAT が有効化された同じネットワークインターフェースに設定されている場合、NAT は機能しません。有効な NAT 設定にするためには、2 つ以上のネットワークインターフェースを用意するか、このページでさらに具体的なルールを定義する必要があります。

このオプションを有効化すると、以下の効果があります。

- このインターフェースによって転送される全てのトラフィックに対して NAT が有効化されます。トラフィックマネージャは、トラフィック内のソース IP アドレスをゲートウェイインターフェースの IP アドレスに置き換えます(トラフィックマネージャは、ソースポート番号を恣意的なポート番号に置き換えることもあります)。
- 宛先ノードからは、トラフィックがトラフィックマネージャのゲートウェイ IP アドレスから送信されているように見えます。宛先ノード(および全ての中間ホップ)がトラフィックをその IP アドレスにルートバックする方法を認識します。
- 宛先ノードがレスポンスを返すと、トラフィックマネージャは、NAT が有効化されたインターフェースでその応答を受信します。その後、元の送信元 IP アドレスとポートがレスポンスに挿入され、接続元のバックエンドノードに転送されます。

時間と日付の設定

注: Xen 仮想アプライアンスの場合、本セクションで説明する設定とは多少異なります。時間はホストハードウェアと同期されるため、この場合に変更できる設定はタイムゾーンのみです。

トラフィックマネージャとバックエンドノードのクロックを相互に同期しておくと、各マシンのログファイルを比較するときに、各システムで発生したイベントがどのように関係しているかを特定できるため便利です。

時間が相互に同期されていないと、トラフィックマネージャが記録したエラーとバックエンドサーバが記録したエラーが対応しているかどうかなど、イベントを相互に関連付けることが困難になります。

ローカルのトラフィックマネージャインスタンスの時間設定を管理するには、管理インターフェースの **System (システム)** セクションの **Time (時間)** ページを使用します。

注: 管理インターフェースのその他多くの設定とは異なり、このページで行う設定は、アクセスしている管理インターフェースが属するローカルのトラフィックマネージャにのみ適用されます。クラスタに含まれる他のトラフィックマネージャマシンの時間設定をチェックまたは設定するには、そのマシンの管理インターフェースを使用する必要があります。

時刻の手動設定

現在のシステム時間、日付、およびタイムゾーンを表示したり、必要に応じて設定を手動で行うには、**Basic Settings (基本設定)**セクションを使用します。

NTP サーバの使用

同じ NTP サーバを使用するよう全てのトラフィックマネージャインスタンスおよびバックエンドサーバを設定すると、これらのインスタンスおよびサーバのクロックを正確に設定し、相互にほぼ同期させることができます。

パブリック NTP サーバを使用できますが、ネットワークの待ち時間や障害の可能性があることから、十分な信頼性が得られない場合があります。そのため、ローカル NTP サービスを実行し、全てのローカルシステムをこのサーバに同期することをお奨めします。ローカル NTP サーバは、ISP が管理する NTP サーバまたはパブリック NTP サーバと自動的に同期することができます。

何らかの理由により、トラフィックマネージャの仮想アプライアンスまたはクラウドインスタンスの時間が正しい時間から大幅に(30 分など)ずれている場合、NTP はこのような大きなずれを補正しません。この場合に時間のずれを補正するには、**System (システム) > Time (時間)**をクリックし、**Sync Time Now (時刻の同期化)**をクリックします。

トラフィックマネージャからの時刻の同期

一般に、トラフィックマネージャアプライアンス/インスタンスは複数のネットワークにまたがることから、ネットワーク接続された様々なデバイスの一貫したローカル時間管理ソースとして適しています。トラフィックマネージャは、全てのインターフェースでアクセス可能なローカル NTP サーバを実行するため、このサーバを基準に時間を同期するよう他のサーバを設定することができます。

トラフィックマネージャへのリモートログイン

トラフィックマネージャ仮想アプライアンスまたはクラウドインスタンスのコマンドラインにアクセスするには、SSH を使用します。「管理」グループまたは「Appliance Console (アプライアンスコンソール)」パーミッションが有効化された他のグループのユーザのクレデンシャルを使用します。詳細については、392 ページの「パーミッショングループ」を参照してください。

適切なクレデンシャルでログインしたユーザには、トラフィックマネージャアプライアンス/インスタンスへの **root** アクセスが与えられ、ホームディレクトリは **/root** に設定されます。

パスワードを使用しない、SSH 公開/秘密鍵による識別を使用したログインを設定するには、トラフィックマネージャで SSH 設定ディレクトリ **/root/.ssh** を使用します。パスワードを使用しないログインへの SSH アクセスを制限するには、管理 UI で **System (システム) > Security (セキュリティ) > SSH Server (SSH サーバ)** の設定を使用します。

カスタムカーネルパラメータの入力

sysctl は、システムの作動中にリアルタイムでカスタムアプライアンスカーネルパラメータを追加するために使用されるインターフェースです。これらのパラメータは、アプライアンスの管理 UI の **System (システム) > Sysctl** ページから追加および変更することができます。

重要: このインターフェースは、高度な使用を行うためのために用意されています。これらの値を変更しようとする、システムの性能が予想しない結果になったり、サポートされない結果になったりするおそれがあります。変更することによって生じる結果について十分に理解している場合にのみ、これらの設定を変更することをお奨めします。

一部のカーネルパラメータはトラフィックマネージャによって制御され、他の場所で適用した他の設定から生成されます。これらのパラメータのいずれかを変更しようとしても、トラフィックマネージャからエラーメッセージが返され、加えた変更は適用されません。

パラメータの追加または変更

カスタムカーネルパラメータを新たに追加するには、**Key** フィールドにパラメータ名を、**Value** フィールドにそれに対応する値を入力します。複数の値を入力する場合は、各値の間にスペースを 1 つ入れて区切ります。後で参照するときや、問題が発生したときのために、オプションの **Description** フィールドにパラメータに関する説明を追加しておくことをお奨めします。 **Submit Entry** (エントリを提出) をクリックして、アプライアンスにパラメータを追加します。

既存のパラメータの値を変更するには、新しい値をもう一度入力します。全ての変更は、*Existing Entries* のテーブルに反映されます。

重要: 新しいパラメータおよび変更されたパラメータは、全てただちに有効になります。

既存のエントリ

既存のパラメータの説明を確認するには、マウスカーソルを *Info* タグの上に置いて説明を表示させます。説明が入力されていなかった場合、*Info* タグは存在しません。

カスタムパラメータを削除するには、削除したいパラメータの隣のボックスをチェックし、**Delete selected entries** (選択したエントリを削除)をクリックします。選択した全てのパラメータはただちに削除されますが、削除プロセスを完了するためには、サーバをリブートする必要があります。それまでにオーバーライドされたデフォルト値は、全て復元されます。

トラフィックマネージャは、入力されたパラメータの有効性をチェックします。エラーが見つかった場合は、このページ、イベントログ、および **Diagnose** (診断)ページに表示されます。

CHAPTER 23 システムセキュリティ

本章では、トラフィックマネージャソフトウェアをセキュアな環境で運用するための重要事項について説明します。この章で OS の設定を考慮する必要のある項目は、トラフィックマネージャソフトウェア製品のみには適用され、アプライアンスまたはクラウド製品には適用されません。

ファイアウォールと OS の設定

トラフィックマネージャはネットワークレベルのファイアウォールではないため、ファイアウォールの背後に配置することを強くお奨めします。ご使用の OS および UNIX 全般に潜在するセキュリティ上の問題を減らすために、追加措置を取ることができます。ここでは、これらの措置について簡単に説明します。

重要: 以降に示す注意事項は、特にトラフィックマネージャ製品に重点を置いて要点を述べたものです。ここに記載されている内容以外にも、インターネットサービスの提供に関連するあらゆるセキュリティ上の考慮事項には十分留意し、必要に応じて専門的な助言を求めることをお奨めします。

ファイアウォールの設定手法

トラフィックマネージャは、リクエストに対してレスポンスを返すことができません。また、一部のプロトコル(FTP など)では、クライアントとの間に追加のコネクションを確立することが必要になる場合があります。ステートフルファイアウォールを設定すると、トラフィックマネージャから任意のポートで外部へのコネクションを確立し、接続先のクライアントから送信される以降のレスポンスをサーバまで通過させることができます。

トラフィックマネージャには、ステートフルファイアウォールを設定することをお奨めします。ステートフルファイアウォールは、デフォルトでは外部から受信する全てのトラフィックを拒否しますが、トラフィックマネージャの指定ポートとの通信は許可され、トラフィックマネージャへの全てのレスポンスはファイアウォールを通過することができます。

トラフィックマネージャでのファイアウォールの設定

トラフィックマネージャは、設定されている仮想サーバが使用するポートやプロトコルに加えて、以下のようなポートやプロトコルも必要とします。

1. **管理サーバポート(HTTPS)** - 通常、ポート **9090** (TCP および UDP)で、マシン名を使用して管理サーバインターフェースにアクセスするために使用するポートです。このポートは、管理サーバインターフェースにアクセスするときに使用する全てのマシンに対して開かれていなければなりません。

2. **コントロールポート** - デフォルトは **TCP:9080** で、トラフィックマネージャ間で設定データの変更をやり取りするために、トラフィックマネージャのコントロールプロトコルで使用されます。このポートは、クラスタ外部に公開されている必要はありません。コントロールポート番号は、`ZEUSHOME/zxtm/global.cfg` で確認できます。
3. **REST API ポート** - デフォルトのポートは **TCP:9070** です。REST API は、トラフィックマネージャを設定するための、プログラムによる代替方法です。詳細については、『*Brocade Virtual Traffic Manager: REST API Guide*』を参照してください。このドキュメントは、Brocade の Web サイト(<http://www.brocade.com>)から入手できます。

管理サーバと REST API の両ポートは、トラフィックマネージャの管理 UI の **System** (システム) > **Security** (セキュリティ) ページから、必要に応じて定義し直すことができます。それにより、ファイアウォールルールの更新が必要になります。

トラフィックマネージャは、他のトラフィックマネージャサーバの状態を確認するためにブロードキャストも使用します。これらのブロードキャストは、マルチキャストアドレス **239.100.1.1**、ポート **9090** に送信され、クラスタ内の全てのトラフィックマネージャがこのアドレスでブロードキャストを監視します。ブロードキャストの IP アドレスおよびポートを変更するには、**System** (システム) > **Fault Tolerance** (フォールトトレランス) の設定を使用します。

CHAPTER 2 の「**専用管理ネットワーク**」セクションで説明した、専用管理ネットワークを使用するようクラスタ内の各トラフィックマネージャを設定した場合、この管理トラフィックはそのネットワークにのみ制限され、トラフィックマネージャの管理ポートに他のネットワークから接続することはできません。ただし、設定エラーによってトラフィックマネージャの管理制限が緩和されたり、管理ネットワークが誤って公開された場合などに備えて、ファイアウォールによってこれらのポートを信頼されていない送信元から保護する必要があります。

トラフィックマネージャ仮想/クラウドインスタンスは、全てのインターフェースで NTP (時間) リクエストを監視するローカル NTP サービスを実行します (CHAPTER 22 の「**トラフィックマネージャからの時刻の同期**」セクションを参照)。この NTP サービスはポート **123** で実行されます (UDP および TCP)。トラフィックマネージャをローカル時間管理ソースとして使用しない場合は、これらのポートをファイアウォールによって保護すると安全です。

重要: トラフィックマネージャがネットワークの可用性を検出するために使用するテストの 1 つに **Ping** があります。ファイアウォールやローカル TCP/IP の調整を無効化したり、デフォルトゲートウェイ、バックエンドノード、およびクラスタ内のその他のトラフィックマネージャ宛ての **ICMP** パケットにレート制限を適用したりすると、トラフィックマネージャが正常に機能しなくなる可能性があります。

ネットワーク設計

トラフィックマネージャは、高度なトラフィック管理機能を備えたプロキシです。すなわち、通常、トラフィックマネージャはパブリックネットワークとプライベートネットワークの間に配置され、これらのネットワークへとリクエストを渡します。各トラフィックマネージャは、トラフィックネットワークとバックエンドネットワークの両方のマシンと通信できなければなりません。

例えば、バックエンドプールが RFC1597 プライベートネットワーク上にあり、その IP アドレスが以下のいずれかのグループに該当するとします。

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
```

インターネット側からこれらのネットワークに直接ルーティングすることはできません。このような構成では、内部ネットワーク空間を都合よく割り当てることができる上、外部ルータが適切に設定されていればこれらのネットワーク宛てのパケットが受け入れられないため、セキュリティ上のメリットが得られます。

そのため、トラフィックマネージャでは、インターネットコネクション用にはトラフィック IP アドレス(トラフィック IP グループ)を使用し、全てのバックエンドプールとのコネクションには、別の物理ネットワーク(または正しく設定された VLAN)上の RFC1597 ネットワークアドレスを使用するのが最も安全です。これにより、インターネットからバックエンドサーバに直接アクセスできなくなり、外部からの全てのコネクションをトラフィックマネージャで安全かつ効率的に管理することができます。

さらに、トラフィックマネージャ管理サーバへのアクセスなどの内部管理トラフィック用に、別の RFC1597 プライベートネットワークを使用することもできます。詳細については、CHAPTER 2 の「専用管理ネットワーク」セクションを参照してください。

Unix ユーザパーミッション

トラフィックマネージャソフトウェアは、root として起動する必要があります。これにより、zeus.zxtm プロセスを特権ポート(1024 番未満)にバインドすることができます。例えば、ポート 80 (HTTP)は特権ポートです。これらの権限は、イーサネットインターフェースを有効化/無効化するときに必要になります。

受信トラフィック(信頼されていない入力)を管理するトラフィックマネージャプロセスは、プロセスの開始後にスーパーユーザ権限を放棄します。ソフトウェアのこの部分は、インストール中に指定した UNIX ユーザおよびグループを使用して実行されます。トラフィックマネージャ管理サーバ(zeus.admin)は、root ユーザが所有する設定を管理するときにスーパーユーザ権限を使用します。

インストールプロセスの詳細については、ご使用の製品の『**Installation and Getting Started Guide**』を参照してください。トラフィックマネージャプロセスのパーミッションが不明な場合は、ps を使用するか `ZEUSHOME/zxtm/global.cfg` を調べることによって、zeus.zxtm プロセスの所有ユーザを確認してください。通常、このユーザは nobody、グループは nobody または nogroup です。

許可を変更する場合は、システムのスーパーユーザとしてトラフィックマネージャの「configure」スクリプト(`ZEUSHOME/zxtm/configure`)を実行することをお奨めします。詳細については、『**Installation and Getting Started Guide**』を参照してください。

ファイルシステムのセキュリティ

`ZEUSHOME` 内の全てのファイルはインストール中に設定され、以降の変更はトラフィックマネージャ管理サーバによって行われます。すなわち、`ZEUSHOME` 内の全てのファイルは、スーパーユーザが所有します。特定のファイルには、非特権許可で実行される zeus.zxtm プロセスがトラフィックマネージャの管理ファイルを読み取れるよう、全てのユーザに読み取り許可を与えるフラグが設定されています。

必要に応じて、トラフィックマネージャを chroot ジェイル内で実行することにより、ファイルシステムに対する `ZEUSHOME` 以外の全てのアクセスを効果的に阻止することができます。このとき、必要なシステムライブラリおよびその他のシステム固有の設定情報を用意し、`/dev` 内にデバイスエントリを作成する必要があります。詳細については、ご使用のシステムのマニュアルを参照してください。

注: Syslog のロギングは UNIX ソケットに基づくため、ファイルが `ZEUSHOME` 外に書き込まれる場合があります。

OS の設定

最高のセキュリティを確保するために、トラフィックマネージャ以外のアプリケーション、特にネットワークサービスを提供するアプリケーションを実行するためにトラフィックマネージャを使用しないでください。新しいトラフィックマネージャの使用を開始すると、事前インストールチェックにより、使用されない UNIX サービスが削除または無効化されます。例えば、多くの場合、lpd (UNIX プリントサーバ)や RPC Portmapper などのサービスはデフォルトで起動しますが、トラフィックマネージャの動作には必要ありません。これらのサービスを完全にアンインストールすることにより、トラフィックマネージャサーバへのアクセス手段となる公開インターフェースを大幅に減らすことができます。

サービスの中には必要なものもあります(Syslog、セキュアシェルなど)。これらのサービスが外部で必要ない場合は、可能であれば、これらのサービスをプライベートネットワークインターフェースまたはループバックインターフェースにのみバイン

ドしてください。これにより、サーバに対する不要な関心を避けることができます。この対策は、ファイアウォールが配備されている場合も行ってください。

インストール前に、サーバのポート 22 を、外部からの ssh コネクション用として管理ポート(設定されている場合)にのみ開いておくことをお奨めします。開いているポートは、無料で入手可能な netstat および nmap ツールで確認できます。

UNIX サーバの管理には、定期的な OS の保守作業が必要になります。また、セキュリティの確保は継続的なプロセスです。特に、ベンダーから提供されるパッチをトラッキングし、セキュリティ関連の問題に対する新しいパッチが提供されたら、インターネットに公開されているサービスをすぐにアップグレードするのが最善の方法です。これは、トラフィックマネージャが管理しているサービスだけでなく、管理用の SSH サーバなどトラフィックマネージャにインストールされているツールも対象になります。Linux など多くの OS では、セキュリティ問題が特定されて修正されるたびに、カーネル自体のアップグレードが必要になる場合があります。

CHAPTER 24 管理サーバのセキュリティ

本章では、Web ベースの管理サーバであるトラフィックマネージャシステムの管理コンポーネントへのアクセスをコントロールする方法、REST API、コントロール API (CLI も含む)、および SSH によるリモートアクセス(仮想アプライアンス/クラウドインスタンスのみ)について説明します。

管理サーバの基本設定

基本的なセキュリティ設定を管理するには、**System** (システム) ボタンをクリックし、次に **Security** (セキュリティ) タブをクリックします。これにより、**Administration Security** (管理セキュリティ) ページが表示されます。このページでは、SSL 証明書、IP ベースのアクセスコントロール、および管理サーバが使用するポートを設定することができます。

管理サーバの SSL 証明書の変更

最初のオプションである **SSL Certificate** (SSL 証明書) では、トラフィックマネージャ管理サーバインターフェースに使用する SSL 証明書を変更できます。自己署名された新しい証明書を生成したり、新しい証明書と秘密鍵ファイルをアップロードしたりできます。完了したら、**Update** (更新) をクリックします。

注: 管理サーバはパブリックサービスではなく、トラフィックマネージャフレームワーク内の別の外部プロセスであるため、証明書をカタログから選択することはできません。

SSL 証明書の SHA-1 フィンガープリントはここに表示されます。これは、以下の場合に役立ちます。

- Web ブラウザとの接続を初めて確立するときに、SSL 証明書を検証する場合
- クラスタへの組み込み時にトラフィックマネージャの身元の真正性を検証する場合

新しいトラフィックマネージャを初めて設定するときに、フィンガープリントを書き留めておくことをお勧めします。ソフトウェア製品の場合、フィンガープリントは、インストールプロセスの完了後にコマンドラインに表示されます。アプライアンスとクラウドインスタンスの場合は、コンソールを参照してください。

フィンガープリントを再度確認する必要がある場合、コマンドラインで以下のコマンドを使用して表示することができます。

```
$ZEUSHOME/admin/bin/cert -f fingerprint -in $ZEUSHOME/admin/etc/admin.public
```

管理サーバに対するアクセスの制限

トラフィックマネージャは、オプションで管理ネットワークを使用できるだけでなく、以下の3つ方法で管理サーバに対するアクセスを制限することができます。

- クライアント IP アドレス(xx.xx.xx.xx)
- クライアント IP とネットマスク(xx.xx.xx.xx/nn または xx.xx.xx.xx/xx.xx.xx.xx)
- クライアントの DNS 名または DNS ワイルドカード(admin.mysite.com または*.mysite.com)

例えば、10.1.1.1, 10.0.0.0/24 または*.mysite.com のように指定することができます。

DNS ルールを使用する場合、トラフィックマネージャは DNS 検索を実行しなければなりません。これらの検索を行えるよう、必要な DNS 設定が行われていることを確認してください。

Update (更新)をクリックすると、トラフィックマネージャにより、ログインボックスへのアクセスを許可する前に管理サーバへの全てのコネクションがチェックされるようになります。

重要: これらの制限を機能させるためには、外部ファイアウォールおよびサービスが信頼されている必要があります。IP アドレスはスプーフィングされる可能性があります。また、DNS サービスが妨害またはハッキングされると、DNS テストを行う意味がなくなる可能性があります。

管理サーバポートの変更

通常、トラフィックマネージャ管理サーバは、各トラフィックマネージャのポート 9090 で運用します。このポートは、**Admin Server Port** (管理サーバポート)セクションの値を調整することにより変更できます。変更内容は、**Update (更新)**をクリックした直後に有効になります。このポートを変更すると、トラフィックマネージャによって新しい URL に自動的にリダイレクトされるようになります。

これは、コントロール API を使用する全てのアプリケーションと、CLI によって確立されたコネクションに影響を与えます。ただし、REST API は、**REST API** セクションで設定可能な代替ポート(通常は 9070)を使用します。

インストール時に専用の管理ポート(ネットワークインターフェース)を指定することができます。この値は、ユーザインターフェースから変更することはできません。この値を変更するか、管理ポートを完全に無効化するには、初期インストール時に使用した **configure** スクリプトを再度実行する必要があります。

詳細については、ご使用の製品の『**Installation and Getting Started Guide**』を参照してください。

Traffic Manager SSH サーバのセキュリティ

注: このセクションは、トラフィックマネージャソフトウェア製品には適用されません。

トラフィックマネージャ仮想アプライアンスとクラウドインスタンスでは、標準の SSH サーバを通じてコマンドラインアクセスが提供されます。コマンドラインアクセスは、主にログの読み込みおよび特定のシステム保守操作に使用し、通常の管理機能には必要ありません。

組織のセキュリティポリシーの一環として、SSH アクセスを制限することができます。この場合、**appliance!ssh!enabled** を Yes (はい) または No (いいえ) に設定して、全てのユーザに対して SSH アクセスを有効化または無効化します。アクセスを有効化した場合、**appliance!ssh!port** を使用して優先ポート(通常は 22)を設定します。

appliance!ssh!passwordallowed を No (いいえ) に設定し、許可された鍵を `/root/.ssh` にインストールすることによって、鍵ベースの認証を使用するように SSH アクセスを制限することもできます。

トラフィックマネージャ仮想アプライアンスとクラウドインスタンスには、ブルートフォース SSH 攻撃の防止に役立つ SSH Intrusion Prevention ツールが含まれています。このツールによって、設定した時間内に、何度もコネクション試行に失敗したリモートホストがブロックされます。SSH Intrusion Prevention を有効化するには、**Enabled** (有効化) を Yes (はい) に設定します。

SSH Intrusion Prevention ツールの動作を設定するには、以下の設定を使用します。

設定	説明
Whitelist hosts (ホワイトリストホスト)	SSH Intrusion Prevention ツールによって禁止されないホストのスペース区切りリスト(そのホストがブラックリストにも含まれている場合を除きます)。IP アドレス、DNS ホスト名、またはサブネットマスクによって指定される IP アドレスの範囲を使用します。
Time (時間)	あるホストが禁止される時間(秒単位)。

設定	説明
Retry limit (再試行上限)	コネクション試行が禁止されるまでに、ホストがトラフィックマネージャに対して実行できるコネクション試行の失敗の回数。 ホストが禁止されている間に実行されたコネクション試行は、次の禁止でカウントされません。ただし、ホストが禁止される時間(「Time (時間)」)が、モニタリング対象のタイムスパン(「Time span (タイムスパン)」)よりも短い場合を除きます。これに該当する場合は、ホストの禁止が解除されたときに、禁止の開始前に記録された失敗試行が、現在のモニタリング対象のタイムスパン内に残っている可能性があります。 禁止されている間のコネクション試行は、認証を受けずに、ただち破棄されます。
Time span (タイムスパン)	SSH Intrusion Prevention ツールが失敗したコネクション試行をモニタリングする時間(秒単位)。
Blacklist hosts (ブラックリストホスト)	SSH Intrusion Prevention ツールが、SSH を通じてトラフィックマネージャへのアクセスを決して許可してはならない IP アドレスまたは DNS ホスト名のスペース区切りリスト。ホワイトリストに登録した IP アドレスとホスト名を、ブラックリストに登録しないことをお奨めします(ただし、ブラックリストが優先されます)。 注: ホワイトリストとは異なり、サブネットマスクによって指定される IP アドレス範囲をブラックリストに登録することはできません。

トラフィックマネージャには、「**Banned Hosts (禁止されたホスト)**」の下に、ブラックリストに登録されたホストを含む現在禁止されているホストが表示されます。

禁止および禁止解除された IP アドレス、SSH Intrusion Prevention ツールが起動または停止された時間、および設定の変更の記録を含むログファイルをダウンロードするには、**Download Log (ログのダウンロード)**をクリックします。

クラスタ通信

クラスタメンバ間の通信は、中間者攻撃やその他のタイプの攻撃または介入の影響を受けないよう設計されたセキュアな通信によって処理されます。トラフィックマネージャは、SSL を使用したセキュアな公開/秘密鍵暗号化を提供することにより、場所に関係なくクラスタメンバが安全かつ確実に通信できる環境を実現します。

本セクションでは、クラスタ内通信に制限を適用して、ネットワーク環境に正確なセキュリティを実装する方法について説明します。

各トラフィックマネージャの内部管理ポートと通信できるホスト IP アドレスは、`controlallow` 設定で指定できます。この設定には、IP アドレス、CIDR IP サブネット、および `localhost` から成るリストを指定します。または、`all` に設定して全てのホストにコネクションを許可することもできます。

バージョン 7.0 以降の各トラフィックマネージャでは、「信頼された」または「信頼されていない」というコンセプトが使用されています。信頼された単一のネットワークの外にまで及ぶクラスタの場合、この信頼されたネットワーク外のクラスタメンバが、信頼性が損なわれた状態になっていないかどうかを常に判断できるわけではありません。このようなトラフィックマネージャによって他のトラフィックマネージャの信頼性が損なわれないようにするために、管理者が特定のクラスタメンバに対し、他のメンバより安全性が劣ることを示すマークを付けることのできるシステムが導入されています。言い換えれば、信頼性を損なう高いリスクにさらされているクラスタメンバに対し、その旨を示すマークを付けることができます。信頼されていないトラフィックマネージャは、引き続き設定の更新を受信でき、状態/統計データをブロードキャストしますが、実質的に設定の更新を他のクラスタメンバに複製できなくなります。

この機能を有効化するには、クラスタ内の各トラフィックマネージャに `control!canupdate` フラグを設定します。このフラグは、信頼されたトラフィックマネージャに対しては**有効化**し、信頼されていないトラフィックマネージャに対しては**無効化**することができます。この設定を使用するには、クラスタが複数のトラフィックマネージャで構成されている必要があります。

注: 信頼されていないトラフィックマネージャの管理 UI とコントロール API は、セキュリティ上の理由から無効化されます。マシン固有の設定(ネットワークング、時間/日付、SNMP 設定など)を変更する必要がある場合、最初に `control!canupdate` を有効化してから行う必要があります。ネットワークにおけるセキュリティの必要条件によっては、変更を行うために、トラフィックマネージャをクラスタから一時的に削除することが必要になる場合があります。

デフォルト設定 `control!canupdate!default` は、新しいトラフィックマネージャがクラスタへの組み込み時に設定を継承するためのものです。信頼性の低い場所(クラウド環境など)にある新しいトラフィックマネージャを組み込む前に、この設定を **No** (いいえ)に変更することができます。

重要: 信頼されていないトラフィックマネージャでは管理 UI が無効化されるため、これらのクラスタメンバに対してログインしたり変更を行ったりすることはできません。そのため、何らかの理由で信頼されたメンバと通信できなくなった場合に、クラスタ全体を管理できなくなる可能性が生じます。そのクラスタのコントロールを取り戻すために、新しいトラフィックマネージャを組み込むこともできません。このリスクを回避するために、常にクラスタ内に信頼された冗長トラフィックマネージャを 1 台以上確保しておくことを強くお奨めします。詳細については、サポート窓口にお問い合わせください。

重要: 信頼性が損なわれた 7.0 より古いトラフィックマネージャは、バージョン 7.0 へのアップグレード後も、信頼されていることを示すステータスを指定しても、信頼性が損なわれた状態のままになる *可能性があります*。これは、トラフィックマネージャが、システムセキュリティの問題を回避するためにアップグレード前にどのような変更が行われた可能性があるかを特定できないためです。そのため、アップグレード前に、セキュアでない場所にあったトラフィックマネージャの信頼性が損なわれていなかったことを確認することを強くお奨めします。

同じ理由で、信頼されていないネットワークに及ぶクラスタ内にバージョン 7.0 と 7.0 より古いバージョンのトラフィックマネージャソフトウェアを混在させないことをお奨めします。バージョン 7.0 で行ったクラスタ内のセキュリティ強化は、クラスタ内で動作する全てのトラフィックマネージャのバージョンが同じになるまで有効になりません。

管理サーバおよび内部コネクションの SSL 設定

これらの設定は、管理サーバへのコネクションおよびトラフィックマネージャ内部のセキュアコネクションに関する高度な SSL オプションをコントロールします。デフォルト設定では、多くの環境に適した幅広いセキュリティレベルと互換性が提供されます。

特定のプロトコルまたはコネクション設定との互換性を確保する方法については、システム管理者またはサポート窓口にお問い合わせください。クラスタ内通信への潜在的な効果を期待して SSL または TLS バージョン(または暗号スイート)を切り換える場合は、十分に注意することをお奨めします。例えば、TLS 1.1 から TLS 1.2 に切り換えるには、`admin!support_tls1_1` を無効化する前に `admin!support_tls1_2` を有効化し、クラスタ全体にわたって変更した設定が再現されるようにします。この遷移の間、非互換性や通信不良を示すような永続的な警告またはエラーが発生していないか、トラフィックマネージャのイベントログを確認してください。

注: トラフィックマネージャの REST API を使用しているときに、`admin!support_tls1_2` の設定を変更する場合、変更した設定で REST API を使用し続ける前に、まず REST API を無効化してから再度有効化する必要があります。REST API のリスタートについては、384 ページの「REST API へのアクセス」を参照してください。

REST API へのアクセス

このセクションには、トラフィックマネージャの REST API サービスで利用できる多数の設定があります。

REST API サービスを有効化するには、**rest!enabled** を「Yes (はい)」に設定します。REST API サービスを無効化するには、**rest!enabled** を「No (いいえ)」に設定します。デフォルトでは、このサービスは無効化されています。

サービスが監視する TCP ポートを設定するには、**rest!port** を使用します。トラフィックマネージャシステムですでに実行されている他のサービスと競合しない限り、空いている任意のポートを使用することができますが、デフォルトのポートは 9070 です。

Update (更新) をクリックして変更を保存します。

このページの残りの設定については、『*Brocade Virtual Traffic Manager: REST API Guide*』を参照してください。このドキュメントは、Brocade の Web サイト (<http://www.brocade.com>) から入手できます。

ユーザ管理

管理 UI、コントロール API、REST API、および CLI では、各コネクションを認証するためにユーザ名とパスワードが必要になります。このログインによりユーザが認証され、そのパーミッショングループに基づいて、設定およびシステムに対してユーザに認可される読み取り、書き込み、または管理操作が定義されます。

ユーザとグループを設定するには、**System (システム) > Users (ユーザ)** をクリックします。

ユーザ認証

管理サーバは、ユーザのクレデンシャル(ユーザ名とパスワード)を以下の 2 つの認証ソースと照合して検証します。

- **ローカルユーザ** - クレデンシャルは管理サーバに(トラフィックマネージャの設定の一部として)保存されます。

管理サーバには、完全な管理パーミッションを持つデフォルトのローカルユーザ **admin** (**admin** パーミッショングループのメンバ)が設定されています。基本的な管理環境では、このアカウントでログインして設定タスクを実行します。

ユーザアカウントの名前は変更できませんが、ユーザアカウントを削除することはできます。「管理」ユーザを削除する場合は、ログインしてクラスタを管理できるように、削除する前に同等の管理パーミッションを持つ別のユーザアカウントが存在することを確認してください。

- **リモート認証されたユーザ** - クレデンシャルは、RADIUS、LDAP、または TACACS+ サービススペースの外部に存在するシステムに照らして認証されます。

User Management (ユーザ管理) ページで 1 つ以上の認証コードを設定できます。認証コードでは、管理サーバがユーザ名とパスワードをデータベースに照合して検証する方法と、ユーザが属するパーミッショングループを特定する方法を定義します。

ユーザがログインを試行すると、管理サーバは、まずクレデンシャルをローカルユーザのリストと照合し、ユーザが属するパーミッショングループを特定します。一致するエントリが見つからなかった場合、管理サーバは、ユーザが認証されてパーミッショングループが特定されるまで、各認証コードを順番に試します。

ローカルユーザ

新しいローカルユーザを追加するには、**Create New User** (新規ユーザ作成) ボックスにユーザ名とパスワードを指定し、ドロップダウンボックスからグループを選択します。これにより、このユーザが属するパーミッショングループが指定されます。

既存のユーザのパスワード、パーミッショングループ、または UI プリファレンスを変更するには、**Local Users** (ローカルユーザ) ページで該当するユーザ名をクリックします。

ローカルユーザの UI プリファレンス

ローカルユーザは、管理 UI の要素をコントロールする以下のプリファレンス設定にアクセスできます。

- **use_applet:** 全ての UI ページに表示されるステータスアプレットの有効化/無効化を切り替えます。このアプレットを無効化すると、帯域を減らしたり、LiveHTTPHeaders や FireBug などのツールを使用してトラフィックマネージャ IP への HTTP トラフィックをモニタリングするときの HTTP リクエストを減らすことができます。
- **appletwidth:** アプレットの幅を変更します。これにより、仮想サーバごとのトラフィックをグラフ化したチャートに表示されるバーを増やす(または減らす)ことができます。

- **trafficscript_editor:** 高度な TrafficScript エディタの有効化/無効化を切り替えます。無効化すると、単純なテキストボックスに置き換わります。

ユーザを削除するには、**Delete User** (ユーザ削除)をクリックします。

パスワードポリシー

Password Policy (パスワードポリシー)ページでは、ローカルユーザを作成したりローカルユーザがパスワードを変更するときに適用されるパスワードポリシーを設定することができます。ここでは、パスワードの長さ、含めなければならない文字タイプ、変更できる頻度、および更新しなければならない頻度の制限を設定できます。

以下のセキュリティ設定を指定できます。

- **No restrictions:** あらゆるパスワードを使用でき、制限は適用されません。
- **Default restrictions:** 新しいパスワードに標準のパスワードセキュリティが適用されます。具体的には以下のとおりです。
 - パスワードの長さは 8 文字以上でなければなりません。
 - パスワードに 2 文字以上のアルファベット文字を含める必要があります。
 - パスワードに 1 文字以上の大文字を含める必要があります。
 - パスワードに 1 文字以上の数字を含める必要があります。
 - パスワードにアルファベット以外の特殊文字を 1 文字以上含める必要があります。
 - パスワードで「aaaaa」のように同じ文字を繰り返すことはできません。
- **Custom restrictions:** パスワードの最小長さと、含めることのできる文字のタイプを手動で指定できます。

また、トラフィックマネージャでは、各ユーザが使用したパスワードの履歴が維持されます。**password_reuse_after** 設定を使用すると、ユーザがパスワードを何回変更した後に同じパスワードを再び使用できるかを指定できます。これは、ユーザがパスワードを変更するか変更を求められるたびに、単純に同じパスワードを再設定しないようにする上で役立ちます。値 0 は、ユーザが以前使用した任意のパスワードを再利用できることを示します。

password_changes_per_day 設定では、ユーザが 24 時間以内にパスワードを変更できる回数を指定します。0 に設定すると、1 日のパスワード変更回数は制限されません。

認証コード

ユーザがローカルユーザのリストで見つからない場合、管理サーバは、クレデンシャルを、完全に設定されている(また無効化されていない)全ての認証コードに照合し

ます。認証コードにより外部データがクエリされ、有効なクレデンシャルを持つユーザのパーミッショングループの名前が返されます。一般に、パーミッショングループの名前はリモート認証データベースのフィールドに保存され、有効なパーミッショングループを持たないユーザには管理サーバへのアクセスが許可されません。

管理サーバは、LDAP、RADIUS、および TACACS+ 認証データベースをサポートしています。

認証コードの作成

認証コードを追加するには、**Users (ユーザ) > Authenticators (認証コード)** ページに移動します。名前を指定し、適切な認証コードタイプを選択して、**Create Authenticator (認証コード生成)** をクリックします。これにより、未設定の認証コードが作成されます。選択したタイプに適した設定を行う必要があります。認証コードを設定してテストしたら、「**enabled (有効化)**」設定を使用してその認証コードを有効化することができます。

管理サーバでは、この方法で認証コードが有効化されるまでその認証コードは使用されません。複数の認証コードを有効化した場合、管理サーバは、パーミッショングループの読み込みに成功するまで各認証コードを(アルファベット順に)試します。

LDAP 認証コード

LDAP 認証コードについて、以下の設定を行うことができます。

設定	説明
ldap!server	LDAP サーバの IP アドレスまたはホスト名です。
ldap!port	LDAP サーバに接続するために使用するポートです。
ldap!timeout	LDAP サーバへの接続のタイムアウト時間(秒単位)です。
ldap!basedn	ディレクトリ検索用のベース DN (識別名)です。
ldap!filter	ベース DN にあるユーザをユニークに識別するフィルタです。文字列「%u」はユーザ名に置き換えられます。例: 「sAMAccountName=%u」 (Active Directory) または 「uid=%u」 (Unix LDAP)

設定	説明
ldap!dnmethod	ユーザのバインド DN は、既知の文字列 (「Construct」) から作成できます。または、ディレクトリ内を検索できます (「Search」 - ユーザが他のディレクトリパスにある場合に必要)。
ldap!binddn	ユーザ名から binddn を作成するためのテンプレート。文字列「%u」はユーザ名に置き換えられます。例: 「%u@mycompany.local」または「cn=%u, dn=mycompany, dn=local」。 dnmethod が「Construct」の場合のみ使用されます。
ldap!searchdn / ldap!searchpass	ディレクトリでユーザのバインド DN を検索するときに使用するバインド DN とパスワードです。匿名バインドを使用したバインド DN の検索が可能な場合は、ブランクのままにすることができます。これらの設定は、dnmethod が「Search」の場合に使用されます。
ldap!groupfilter	上記の ldap!filter によって返されたユーザレコードに必要なグループ情報が含まれていない場合、ここで代替グループ検索フィルタを指定できます。通常、これは Unix/POSIX 形式のユーザレコードがある場合に必要になります。複数のレコードが返された場合、その全てからグループ名のリストが抽出されます。文字列%u はユーザ名に置き換えられます。 例: (&(memberUid=%u)(objectClass=posixGroup))
ldap!groupattr	ユーザのグループを指定する LDAP 属性です。複数の値が LDAP サーバから返された場合、最初の有効な値が使用されます。
ldap!groupfield	ユーザのグループを指定するグループ属性のサブフィールドです。例えば、groupattr が「CN=mygroup, OU=groups, OU=users, DC=mycompany, DC=local」のような内容の「memberOf」の場合、groupfield を「CN」に設定します (一致する最初のフィールドが使用される)。

設定	説明
ldap!fallbackgroup	上記の groupattr が定義されていない、もしくはユーザに設定されていない場合、ここで指定するグループが使用されます。指定しない場合、 groupattr に一致する属性を持たないユーザはアクセスを拒否されます。

RADIUS 認証コード

RADIUS 認証コードについて、以下の設定を行うことができます。

設定	説明
radius!server	RADIUS サーバの IP アドレスまたはホスト名です。
radius!port	RADIUS サーバに接続するために使用するポートです。
radius!timeout	RADIUS サーバへのコネクションのタイムアウト時間(秒単位)です。
radius!secret	RADIUS サーバと共有する秘密鍵です。
radius!groupvendor	アカウントのグループを指定する RADIUS 属性のベンダーを表す RADIUS 識別子です。Filter-Id などの標準属性*を使用する場合は、ブランクのままにします。
radius!groupattr	アカウントのグループを指定する属性を表す RADIUS 識別子です。 radius!fallbackgroup を指定した場合は、ブランクのままにすることができます。
radius!fallbackgroup	ベンダーおよびグループ識別子によってグループが見つからないか、見つかったグループが有効でない場合、ここで指定するグループが使用されます。

設定	説明
radius!nas-ip-address	この値は RADIUS サーバに送信されます。ブランクのままにすると、サーバへの接続に使用されたインターフェースのアドレスが使用されます。
radius!nas-identifier	この値は RADIUS に送信されます。

TACACS+認証コード

TACACS+認証コードについて、以下の設定を行うことができます。

設定	説明
tacacsplus!server	TACACS+サーバの IP またはホスト名です。
tacacsplus!port	TACACS+サーバに接続するためのポートです。
tacacsplus!timeout	TACACS+サーバへの接続のタイムアウト時間(秒単位)です。
tacacsplus!secret	TACACS+サーバと共有する秘密鍵です。
tacacsplus!authtype	使用する認証タイプです。 PAP または ACSII に設定できます。
tacacsplus!groupsvc	各ユーザのグループフィールドを提供する TACACS+の「service」です。
tacacsplus!groupfield	各ユーザのグループを提供する TACACS+の「service」フィールドです。
tacacsplus!fallbackgroup	tacacsplus!groupsvc が定義されていないか、TACACS+サーバからユーザにグループ値が提供されていない場合、ここで指定するグループが使用されます。指定しない場合、TACACS+で定義されたグループを持たないユーザはアクセスを拒否されます。

認証コードのテスト

Edit (編集) ページでは、**Test Authenticator** (認証コードのテスト) セクションを使用して認証コードをテストできます。テストするユーザ名とパスワードを指定できます。認証コードが実行され、成功した場合は、ユーザが属するパーミッショングループの名前が返されます。認証コードにより詳細なデバッグ情報も提供されるため、必要に応じて、この情報を基に設定を微調整したり修正したりできます。

使用例: Active Directory に対する認証

LDAP 認証コードを使用して、ユーザを Active Directory データベースに対して認証することができます。

設定	説明
ldap!server	dir.company.com
ldap!port	389
ldap!basedn	OU=Company Users, DC=company, DC=local
ldap!filter	sAMAccountName=%u
ldap!dnmethod	Construct
ldap!binddn	%u@company.local
ldap!groupattr	memberOf
ldap!groupfield	CN

この認証コードを、ユーザ名とパスワードを指定してテストします。

```
Created LDAP connection to dc2:389
Constructed LDAP user filter: sAMAccountName=username
Using dnmethod: construct
Constructed user binddn: username@company.local
Attempting to bind using supplied password: <hidden>
(length=9)
Searching for user, parameters: filter:
sAMAccountName=username, basedn: OU=Company
Users,DC=company,DC=local
Search returned 1 matches
Extracted groups using attribute memberOf
  CN=users,OU=Email Lists,OU=Company
Users,DC=company,DC=local
```

```
CN=admin,OU=Email Lists,OU=Company
Users,DC=company,DC=local
CN=dev,OU=Email Lists,OU=Company Users,DC=company,DC=local
Extracting group using field: CN
Extracted groups:
  users
  admin
  dev
No ldap!fallbackgroup defined
Groups returned by authenticator: users, admin, dev
SUCCEDED, group: admin
```

パーミッショングループ

パーミッショングループは、トラフィックマネージャ管理サーバの各部分に対するアクセス権を様々なユーザに割り当てるために使用します。全てのローカルユーザとリモートユーザはいずれかのグループに属し、これらのグループによって、トラフィックマネージャの様々な部分に対するユーザのアクセス権が定義されます。デフォルトでは、ローカルのニーズに応じて、変更または削除できる定義済みグループがいくつか提供されます。新たにカスタマイズした任意の数のグループを追加することもできます。

新しいグループを追加するには、「**Group name (グループ名)**」ボックスに名前を指定し、**Update (更新)**をクリックします。変更内容が適用されたら、新しいグループ名をクリックします。

各グループには、基本的な「**Description (記述)**」および「**Login Timeout (ログインタイムアウト)**」設定があります。**Description (記述)**には、このグループの目的を示す簡単な説明を指定できます。また、**Login Timeout (ログインタイムアウト)**には、非アクティブな管理サーバセッションを終了するまでの時間を指定します。このグループに属する全てのユーザは、このタイムアウト値を継承します。

グループ内のユーザパーミッションの編集

トラフィックマネージャには、管理サーバで利用可能な様々なページのリストが表示されます。*admin* (管理)グループでは、全てのアクションが許可されます。新しいグループは、作成時点では全ての許可が **None** (なし)に設定されます。

各グループでは、それぞれのページに対する許可を以下のいずれかに設定できます。

- **None:** グループメンバはこの機能を使用できません。
- **Read Only:** グループメンバは、このページに関連付けられた項目を表示できますが、変更することはできません。
- **Full:** メンバは、このページに関連付けられた項目を表示および変更できます。

例えば、**Catalog** (カタログ) > **SSL Certificates** (SSL 証明書) ページについて考えます。許可を「None」に設定した場合、このグループのメンバは、**SSL Certificates Catalog** (SSL 証明書カタログ) ページを表示できません。Read Only 許可を指定した場合、メンバはこのページを表示できますが、このページの設定を変更することはできません。許可を「Full」に設定した場合、制限は適用されません。

注: 一部のページ(カタログ概要ページなど)には、設定可能な項目がありません。また、特定ページへのアクセスをブロックしても、階層内でそのページの下位にあるページへのアクセスは制限されず、これらのページでの設定も制限されません。

重要: **Users** (ユーザ) ページに対するアクセス権を持つグループは、各自のアカウントの詳細を編集できます。これらの権限を与えるユーザを設定する場合は、注意することをお奨めします。

ログインタイムアウト

ログインタイムアウトは分単位で測定されます。トラフィックマネージャが非アクティブな状態でこの時間が経過すると、管理サーバとのセッションが終了します。セッションは、ブラウザウィンドウを閉じたり、**Logout** (ログアウト) をクリックして手動でログアウトした場合にも、自動的に終了します。

ログインタイムアウトは、パーミッショングループのプロパティです。**System** (システム) > **Users** (ユーザ) > **Permissions Groups** (パーミッショングループ) をクリックし、変更するパーミッショングループの名前をクリックします。「**Inactivity Timeout** (非アクティビティタイムアウト)」に新しいタイムアウト値を設定し、**Update** (更新) をクリックします。

サスペンドされているユーザ

このページでは、サスペンドされている 1 人以上のユーザを再度有効化することができます。ユーザのサスペンド理由として、最大許容ログイン試行回数の超過や、その他の管理上の理由があります。

再度有効化する各ユーザの横にあるボックスを選択し、**Enable Selected Users** (選択したユーザの有効化) をクリックします。

Edit (編集) ページで「**Status** (ステータス)」を「**Active** (アクティブ)」に設定しても、個々のユーザを有効化できます。

ログインセキュリティとログイン行動

System (システム) > **Users** (ユーザ) > **Local Users** (ローカルユーザ) > **Password Policy** (パスワードポリシー) ページでパスワードを定義することにより、ユーザの

ログインパスワードをコントロールし、これらのパスワードに制限を適用することができます(CHAPTER 24 の「ローカルユーザ」セクションを参照)。これに加え、ログイン行動設定を定義することにより、システムのユーザに対するコントロールとセキュリティ対策を強化することができます。**System (システム) > Global Settings (グローバル設定)**ページの **Login and Security Settings (ログインおよびセキュリティ設定)** セクションには、設定が多数表示されます。これらの設定は大まかに以下の 2 つのテーマに分かれています。

UI 画面バナー

システムの使用時にユーザに適切なメッセージまたはリマインダを提供する必要がある場合、ここで定義できます。

設定	説明
login_banner	管理 UI またはトラフィックマネージャ SSH コマンドラインにログインしようとするユーザに表示する、バナーのテキストメッセージです。
banner_accept	管理 UI へのログイン前に、ユーザが login_banner のテキストを明示的に受け入れて同意する必要があるかどうかを指定します。そのためのチェックボックスがログインページに追加されます。
uipage_banner	管理 UI の各ページの上部および下部に表示するテキストメッセージです。

ログインコントロール

ユーザが試行できるログイン回数と、この回数を超えた場合の結果に関するコントロールを配置することができます。

設定	説明
max_login_attempts	ユーザアカウントがサスペンドされるまでに許容されるログインの失敗回数。値に 0 を指定すると、この機能が無効化されます。デフォルト: 0 サスペンドされたアカウントは、max_login_suspension_time に設定された待ち時間後に自動的に再びアクティブになりますが、この待ち時間の経過前に System (システム) > Users (ユーザ) > Local Users (ローカルユーザ) > Edit (編集) ページまたは System (システム) > Users (ユーザ) > Suspended Users (サスペンドされているユーザ) ページ (CHAPTER 24 の「サスペンドされているユーザ」セクションを参照) から再びアクティブにすることも可能です。 重要: パスワードベースの認証でログイン試行制限を使用する場合は、注意することをお奨めします。DoS 攻撃によって、管理ユーザのログインでトラフィックマネージャにアクセスできなくなる可能性を防止するために、ファイアウォールまたは独自のユーザ命名規則など、必ず追加のセキュリティ対策を利用してください。
max_login_external	外部で認証された(LDAP、RADIUS、または TACACS+)ユーザが max_login_attempts 回ログインに失敗した場合に、このユーザをサスペンドするかどうかを指定します。
max_login_suspension_time	max_login_attempts 回ログインに失敗したユーザアカウントをサスペンドする時間。
login_delay	ログインに失敗してから次のログインを試行できるようになるまでの待ち時間。デフォルト: 4

ログイン情報バナー

```

❗ Last successful login by admin: 2011-02-24 12:35:03 +0000 from 10.100.1.14 (UI) on coeus.
Failed login attempts since then: 1, last was: 2011-02-24 13:41:31 +0000 from 10.100.1.14 (UI) on coeus.

```

Fig 69. ログイン情報

ホームページの上部に表示される情報バナーには、現在のユーザがこのシステムで成功した前回のログイン試行についての情報が表示されます。また、それ以降に失敗した試行も表示されるため、ログイン信用証明データが予測外または不正に使用された場合に、それを特定することができます。

イベントおよび監査ログ

イベントログには、クラスタ内の全てのトラフィックマネージャからのシステムメッセージおよび設定メッセージ、警告、およびエラーが含まれます。トラフィックマネージャ管理 UI の Home (ホーム) ページには、イベントログの短縮バージョンがあります。完全なログを表示するには、**Examine Logs** (イベントログ) または **Diagnose (診断) > Event Log** (イベントログ) をクリックします。

監査ログは、最新のログイン試行、設定の変更(ユーザおよび送信元 IP 別)、およびユーザログアウトを監視します。また、コントロール API および REST API を使用して行われた変更の記録も行います。監査ログを表示するには、トラフィックマネージャ管理 UI の **Diagnose (診断) > Audit Log** (監査ログ) をクリックします。

仮想アプライアンスおよびクラウドインスタンスの場合は、トラフィックマネージャがイベントログと監査ログのローテーションを自動的にコントロールします。ログは、現在のログで 50MB の最大ファイルサイズに達するとローテーションします。ログがローテーションするときに、その時点までのログエントリの新しいアーカイブファイルが作成され、現在のログがクリアされます。その後、アーカイブファイルは「gzip」で圧縮され、以下のフォーマットのファイル名が付けられます。

- 監査ログの場合: audit-<date>-<sequence>.gz
- イベントログの場合: audit-<date>-<sequence>.gz

上記の<sequence>は、同じ日に複数回ローテーションが行われた場合に付けられる連番です。

トラフィックマネージャのソフトウェアをデプロイする場合は、独自のローテーションポリシーを制定する必要があります。

アーカイブされた監査ログファイルにアクセスするには、**Diagnose (診断) > Audit Log** (監査ログ) ページの「Manage archived audit logs (アーカイブされた監査ログの管理)」をクリックします。アーカイブされたイベントログファイルにアクセスするには、**Diagnose (診断) > Event Log** (イベントログ) ページの「Manage archived event logs (アーカイブされたイベントログの管理)」をクリックします。

アーカイブされた各ログは、表示されるコントロールを使用してダウンロードしたり削除したりできます。

注: 監査ログおよびイベントログファイルのローテーションのためにトラフィックマネージャが使用するデフォルトの最大ファイルサイズは、システムレベルの環境変数 `AUDITLOG_ROTATE_THRESHOLD` または `EVENTLOG_ROTATE_THRESHOLD` をそれに応じて設定することによりオーバーライドできます。特定のアプライアンス/クラウド製品には、この環境変数の代わりに実装される自動しきい値がある場合があります。詳細については、ご使用の製品の『**Installation and Getting Started Guide**』を参照してください。

CHAPTER 25 トラフィックマネージャのコントロール API

本章では、トラフィックマネージャのコントロール API について説明します。詳細については、製品に付属の『*Brocade Virtual Traffic Manager: Control API Guide*』を参照してください。

REST API については、製品に付属する『*Brocade Virtual Traffic Manager: REST API Guide*』を参照してください。

トラフィックマネージャのコントロール API の概要

トラフィックマネージャコントロール API は、トラフィックマネージャクラスタをリモートから管理および設定することのできる、規格に準拠した SOAP ベースの API です。Web ベースの管理サーバの代わりとして使用できるため、トラフィックマネージャを他のアプリケーションから設定する場合に適しています。

例えば、侵入検知システムによってリモート攻撃の試行を検出する場合、トラフィックマネージャコントロール API を使用して、疑わしい IP アドレスからの全てのコネクションを破棄するようトラフィックマネージャを設定することができます。

プロビジョニングシステムでは、トラフィックマネージャのサービスレベルモニタリング機能と SNMP インターフェースを使用してサーバノードのレスポンス時間をモニタリングすることにより、サーバの過負荷状態を検出することができます。サーバを追加してプロビジョニングした後に、トラフィックマネージャコントロール API を使用してサーバプールを再設定することにより、トラフィックマネージャに通知できます。

トラフィックマネージャコントロール API は、SOAP サービスをサポートしている任意のプログラミング言語およびアプリケーション環境で使用できます。一般には、Perl、Python、Java、および C# が使用されます。

注: トラフィックマネージャ製品によっては、トラフィックマネージャコントロール API を利用できない場合があります。詳細については、ご使用の製品の仕様を参照してください。

例: 実行中の仮想サーバのリスト表示

Perl と SOAP::Lite

```
#!/usr/bin/perl -w
```

```
use SOAP::Lite 0.60;

# This is the url of the Traffic Manager admin server
my $admin_server =
'https://username:password@host:9090';

my $conn = SOAP::Lite
    ->
    uri('http://soap.zeus.com/zxtm/1.0/VirtualServer/')
    -> proxy("$admin_server/soap");

# Get a list of virtual servers
my $res = $conn->getVirtualServerNames();
my @names = @{$res->result};

# Establish which are enabled
$res = $conn->getEnabled( \@names );
my @enabled = @{$res->result};

# Print those which are enabled
for( my $i = 0; $i <= $#names; $i++ ) {
    if( $enabled[$i] ) {
        print "$names[$i]\n";
    }
}
```

この例を以下のように実行します。

```
$ ./listVS.pl
Main website
Mail servers
Test site
```

C Sharp または Mono

```
using System;
using System.Net;
using System.IO;
using System.Security.Cryptography.X509Certificates;

public class AllowSelfSignedCerts : ICertificatePolicy {
```

```
public bool CheckValidationResult(
    ServicePoint sp, X509Certificate cert,
    WebRequest request, int problem )
{
    return true;
}

public class listVS {

    public static void Main( string [] args )
    {
        System.Net.ServicePointManager.CertificatePolicy =
            new AllowSelfSignedCerts();

        string url= "https://host:9090/soap";
        string username = "username";
        string password = "password";

        try {
            ZXTM.VirtualServer p = new
            ZXTM.VirtualServer();
            p.Url = url;
            p.Credentials = new
            NetworkCredential( username, password );

            string[] names = p.getVirtualServerNames();
            bool[] enabled = p.getEnabled( names );

            for ( int i = 0; i < names.Length; i++ ) {
                if( enabled[i] ) {
                    Console.WriteLine( "{0}", names[i] );
                }
            }
        } catch ( Exception e ) {
            Console.WriteLine( "{0}", e );
        }
    }
}
```

このコードは、.NET 1.1 SDK および Mono で動作します。

.Net 1.1 を使用する場合、この例を以下のようにコンパイルして実行します。


```
C:\> wsdl -o:VirtualServer.cs -n:ZXTM VirtualServer.wsdl
C:\> csc /out:listVS.exe VirtualServer.cs listVS.cs
C:\> listVS.exe
Main website
Mail servers
Test site
```

Mono を使用する場合、以下のようにコンパイルして実行します。

```
$ wsdl -o:VirtualServer.cs -n:ZXTM VirtualServer.wsdl
$ msc /out:listVS.exe /r:System.Web.Services \
    VirtualServer.cs listVS.cs
$ listVS.exe
Main website
Mail servers
Test site
```

その他の例

その他のプログラム言語を使用した追加のサンプルについては、『*Brocade Virtual Traffic Manager: Control API Guide*』を参照してください。

CHAPTER 26 コマンドラインインターフェース

通常、トラフィックマネージャの管理は、Web ベースの管理サーバを使用して行います。設定をスクリプト化したり他のシステムと統合する場合は、SOAP ベースのコントロール API (CHAPTER 25) および REST API を使用することもできます。

ただし、コントロール/REST API を使用するアプリケーションの作成には時間がかかり、小規模で単純なタスクやその場限りの設定には適さない場合があります。これに代わる手段として、トラフィックマネージャの全ての機能に素早くアクセスできるコマンドラインインターフェース(CLI)を使用できます。

CLI はインタラクティブなシェルとして機能し、スクリプティングも可能です。CLI は、コントロール API を使用してトラフィックマネージャクラスタと通信します。また、CLI で使用できるコマンドは、コントロール API で提供されるメソッドに対応しています。

CLI へのアクセス

CLI を起動するには、以下の場所にある `zcli` プログラムを実行します。

```
ZEUSHOME/zxtm/bin/zcli
```

CLI にアクセスするには、SSH を使用して仮想アプライアンス/クラウドインスタンスに直接ログインし(CHAPTER 22 の「トラフィックマネージャへの」セクションを参照)、`zcli` コマンドを入力します。または、`zcli` コマンドの引数としてユーザとホストを指定することにより、リモートから接続することもできます。

```
zcli [user@]host:port
```

以下に例を挙げます。

```
zcli admin@ztml:9090
```

`zcli` プログラムの使用方法与オプションを表示するには、以下のように引数として「`--help`」を指定します。

```
$ zcli --help
Usage: ./zcli [OPTIONS] [user@][host:port] [script file]
Configure the Traffic Manager from a script file or
standard input.

--user USER          Set the username (default: admin)
--passfile FILE      Read the password from this file
--nossl              Your Traffic Manager does not have
```

```
SSL enabled admin server
--verbose          Verbose output (for testing)
--continue         When running a script, continue
even if an error occurs
--help            Show this help
--version         Show the version of this program
--formatoutput    Use 'human readable' output, even
in script mode
--json            Use strict JSON output, even in
interactive mode
--timeout SECONDS Set timeout for commands

If no host:port is specified, the local Traffic Manager
will be used.
```

許可

CLI は、コントロール API に対する許可を持つユーザ名とパスワードを使用して、トラフィックマネージャの認証を受ける必要があります(CHAPTER 23 の「ネットワーク設計」セクションを参照)。「admin (管理)」グループの全てのユーザは適切な許可を持っており、必要に応じてこれらの許可を他のユーザグループにも与えることができます。

デフォルトでは、CLI は、ユーザ名 **admin(管理)** を使用してトラフィックマネージャと通信します(--user オプションを使用して他のユーザ名を選択することもできます)。

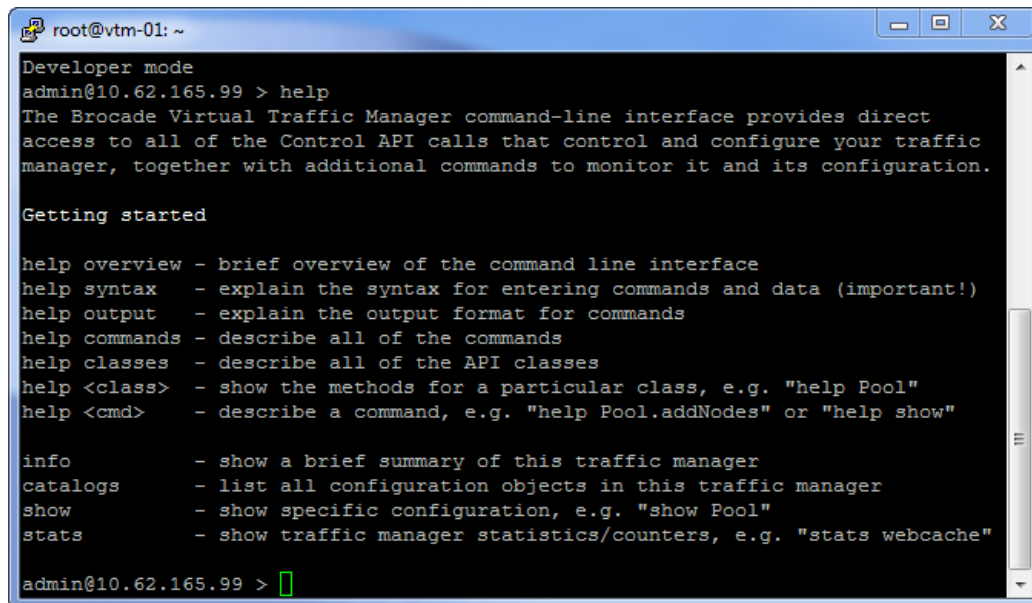
```
$ zcli
Please enter your password for user 'admin' on the local
Traffic Manager
Connected to 127.0.0.1:9090
admin@127.0.0.1 >
```

コネクションが確立されると、CLI に(トラフィックマネージャのユーザ名とホスト名が示された)プロンプトが表示され、コマンド入力待ち状態になります。

```
admin@127.0.0.1 >
```

コマンドは、一度に1つずつ入力できます。コマンド履歴が維持されるため、上下の矢印キーを押して以前入力したコマンドを確認できます。多くのコマンドではオートコンプリート機能も利用できます。Tab キーを1回押すと、途中まで入力したコマンドやオブジェクト名が補完されます。「Tab」キーを2回押すと、有効なオプションのリストが表示されます。

help コマンドを使用すると、CLI の概要と、使用可能なコマンドに関する詳細を参照できるヘルプトピックが表示されます。



```
root@vtm-01: ~
Developer mode
admin@10.62.165.99 > help
The Brocade Virtual Traffic Manager command-line interface provides direct
access to all of the Control API calls that control and configure your traffic
manager, together with additional commands to monitor it and its configuration.

Getting started

help overview - brief overview of the command line interface
help syntax   - explain the syntax for entering commands and data (important!)
help output   - explain the output format for commands
help commands - describe all of the commands
help classes  - describe all of the API classes
help <class>  - show the methods for a particular class, e.g. "help Pool"
help <cmd>    - describe a command, e.g. "help Pool.addNodes" or "help show"

info          - show a brief summary of this traffic manager
catalogs      - list all configuration objects in this traffic manager
show          - show specific configuration, e.g. "show Pool"
stats         - show traffic manager statistics/counters, e.g. "stats webcache"

admin@10.62.165.99 > 
```

Fig 70. Top-level CLI ヘルプ

コマンド

使用可能なコマンドを調べる場合に、詳細なリストを表示するには **commands** と入力し、コマンドとオブジェクトの簡易リストを表示するには **Tab** コンプリート機能を使用します(**Tab** キーを 2 回押す)。

コマンドは、以下の 3 つの主要グループに分類されます。

1. コントロール API メソッド

これらのコマンドは、以下のように **Class.method [arguments...]** の形式を取ります。

```
admin@127.0.0.1 > Pool.getNodes "Web Servers"
[ "92.52.65.222:80", "92.52.65.213:80" ]
```

2. ヘルパコマンド

これらのコマンドは、トラフィックマネージャの設定の概要を表示します。主なコマンドは以下のとおりです。

- **info** - このトラフィックマネージャの概要を表示します。
- **catalogs** - このトラフィックマネージャの全設定オブジェクトのリストを表示します。

- `show` - 特定の設定を表示します(例: `show Pool`)。
- `stats` - トラフィックマネージャの統計情報/カウンタを表示します(例: `stats webcache`)。

3. CLI ヘルプ

- `help syntax` - コマンドおよびデータを入力するときの構文について説明します。
- `help output` - コマンドの出力フォーマットについて説明します。
- `help commands` - 全てのコマンドについて説明します。
- `help classes` - 全ての API クラスについて説明します。
- `help <class>` - 特定クラスのメソッドを表示します(例: `help Pool`)。
- `help <cmd>` - 特定のコマンドについて説明します(例: `help Pool.addNode` または `help show`)。

コントロール API メソッド

CLI では、全てのコントロール API メソッドにアクセスできます。コントロール API メソッドに対応するコマンドは、『Traffic Manager Control API Manual』に記載されているクラス名とメソッド名で構成されます。各コマンドの詳細については、このマニュアルを参照してください。

例えば、全ての仮想サーバのリストを表示するには、以下のコマンドを使用します。

```
admin@127.0.0.1 > VirtualServer.getVirtualServerNames
[ Intranet, "Secure Site", webmail ]
```

CLI コマンドの引数

コマンドの引数は、スペースで区切った名前として指定します。引数間のカンマはオプションです。スペースまたはその他の特殊文字を含む引数値は、構文上異なる意味に解釈される可能性があるため、「引用符」で括ってください。リストは「[」および「]」の角括弧で括ります。

例えば、トラフィックマネージャのコントロール API マニュアルには、Pool クラスの「`getNodes`」メソッドが以下のように記載されています。

`getNodes( names ) throws ObjectDoesNotExist`

指定した各プールのノードのリストを取得するには、以下のように記述します。

```
String[][] getNodes(  
String[] names  
);
```

このメソッドに対応する CLI コマンドは `Pool.getNodes` です。マニュアルには、このコマンドがプール名の配列(`String[] names`)を使用することが示されています。2つのプールに対してこのメソッドを呼び出すコマンド行は以下のようになります。

```
admin@127.0.0.1 > Pool.getNodes [ pool1 pool2 ]  
{ pool1: [ "serverA:80", "serverB:80" ],  
  pool2: [ "serverC:80", "serverD:80" ] }
```

`Pool.getNodes` の出力は、`String[][]`、すなわち配列の配列として宣言されます。指定した各プールについて、ノードの配列が返されます。

便宜上、インタラクティブモードでは、CLI はコマンド出力にインデックスを付け、参照先のプール名のプレフィックスを付けてノードのリストを表示します。

単一引数の使用

コントロール API の多くのコマンドでは引数のリストを使用できるため、一度に複数のオブジェクトに対して操作することができます。単一の項目に対してコマンドを使用する場合、以下のように配列を省略して簡略化することができます。

```
admin@127.0.0.1 > Pool.getNodes pool1  
[ "serverA:80", "serverB:80" ]
```

ワイルドカードの使用

「*」記号は、複数のオブジェクトに一致させることのできるワイルドカードです。

```
admin@127.0.0.1 > VirtualServer.getDefaultPool *  
{ Intranet:      pool1,  
  "Secure Site": "Secure pool",  
  webmail:       "Mail servers" }
```

名前の一部にワイルドカードを使用することもできます。例えば、名前が「Pool」で始まっている全てのプールで `keepalive` を有効化する場合、以下のようになります。

```
admin@127.0.0.1 > Pool.setKeepalive Pool* 1
```

複雑な引数

一部のコマンドは、さらに複雑な引数構造体を必要とします。例えば、コントロール API マニュアルでは、`VirtualServer.addVirtualServer` が以下のように定義されています。

```
void addVirtualServer(  
    String[] names,  
    VirtualServer.BasicInfo[] info )
```

`BasicInfo` 構造体は以下のように定義されています。

```
struct VirtualServer.BasicInfo {  
    # The port to listen for incoming connections on.  
    Integer port;  
  
    # The protocol that this virtual server handles.  
    VirtualServer.Protocol protocol;  
  
    # The default pool that traffic to this virtual  
    server  
    # will go to.  
    String default_pool;  
}
```

構造体を CLI で使用するときは「{」および「}」中括弧で括り、各メンバをキー:値ペアの形式で指定します。以下に例を挙げます。

```
admin@127.0.0.1 > VirtualServer.addVirtualServer  
Intranet { port: 80, protocol: http, default_pool:  
pool1 }
```

コマンドで想定される引数については、CLI から利用可能なヘルプで参照できます。

```
admin@127.0.0.1 > help VirtualServer.addVirtualServer
```

カスタム設定のセット

トラフィックマネージャ設定システムの任意の *name:value* 設定ペアは、CLI を使用して保存したり、読み込んだりできます。この設定はクラスタ全体で反復され、REST API、SOAP API、および CLI を介してのみアクセスすることができます。

カスタム設定の保存および読み込みに関連付けられているコマンドの一覧は、プロンプトで `help Custom` と入力することで確認できます。

カスタム設定を保存するには、`Custom.setStringsLists` コマンドを使用します。以下は、「**counters**」という名前のリストの配下に *name:value* のペアを設定する場合の例です。

```
Custom.setStringLists [counters], [ {"name":  
"Test_Counter", "value" : ["12345", "67890"]} ]
```

必要に応じて、`{"name":<name>, "value":<value>}` の項目を追加することによって、複数の *name:value* 設定ペアをリストに追加できます。

「**counters**」設定を読み込むには、以下のコマンドを使用します。

```
Custom.getStringLists counters
```

このシステムを使用すると、最初に *name:value* ペアごとに新規リストを作成し、さらにそのなかで、カスタム設定を論理グループに区分することができます。*value* 自体は、単一の項目でも、項目のリストでも構いません。

内蔵コマンド

CLI には、簡単に設定を表示または変更できる追加のコマンドが多数用意されています。詳細なヘルプおよび使用法は、`help <commandname>` を使用して参照することができます。

- **info:** 管理 UI のホームページと同様に、このトラフィックマネージャの概要情報を表示します。クラスタ内のトラフィックマネージャ、設定されている仮想サーバとプール、トラフィック IP アドレス、設定または操作エラー、およびイベントログに記録されている最新エントリが表示されます。
- **connect:** 別のトラフィックマネージャに接続します。CLI では、ローカルマシンと同じクラスタ内にないトラフィックマネージャなど、リモートのトラフィックマネージャと通信することができます。スクリプトファイルにはトラフィックマネージャのホスト名、ユーザ名、およびパスワードを全て記述できるため、このコマンドはスクリプトを作成するときにも役立ちます。
- **errorlog:** トラフィックマネージャのイベントログに記録されている最新のメッセージを表示します。
- **catalogs:** モニタ、TrafficScript ルール、Java Extensions など、設定カタログのコンテンツのリストを表示します。

- **show:** トラフィックマネージャの設定オブジェクトの設定を表示します。以下に例を挙げます。
 - `show VirtualServer Intranet` - 仮想サーバ「**Intranet**」の概要を表示します。
 - `show Pool` - 全てのプールの概要を表示します。
 - `show VirtualServer customer*` - 仮想サーバ **customer***の概要を表示します。

以下のように、オブジェクト名のみを指定して **show** を使用することもできます。この場合、指定した条件に一致する全てのオブジェクトの概要が表示されます。

- `show Staging` - 「**Staging**」という名前の全てのオブジェクトの概要を表示します。
- `show *` - 全てを表示します(時間がかかる場合があります)。

show コマンドを使用して、現在トラフィックマネージャによって処理される全ての接続の詳細を指定することもできます(`show connections` コマンドを使用します)。出力は、管理 UI の **Activity** (アクティビティ) > **Connections** (コネクション) ページで表示されるコネクション情報に基づいており、コネクションの進行中には以下のデータ項目がリストされます。

データ項目	説明
From	接続するクライアントの IP アドレスとポート
Via	クライアントが接続したローカル IP アドレスとポート
To	コネクションに使用されるバックエンドノード

データ項目	説明
State	以下のいずれかのような、現在のコネクション状態を表示します。 C: クライアントへのコネクションが切断された f: 接続完了 R: クライアントから読み込み中 W: クライアントへ書き込み中 X: クライアントリクエストに対してルールを実行中 c: バックエンドノードに接続中 w: バックエンドノードへ書き込み中 r: バックエンドノードから読み込み中 C: クライアントとのコネクションを切断中 K: クライアントコネクションを keepalive 状態で保持中
仮想サーバ	このコネクションを処理する仮想サーバ
ルール	コネクションで最後に実行されたルール
プール	使用されたプール
BytesIn	クライアントから受信されたバイト数
BytesOut	クライアントへ送信されたバイト数
Timings	<num1> / <num2> / <num3>のフォーマット。意味は以下のとおりです。 <num1>: クライアントがこのコネクションを確立してからの時間 <num2>: クライアントコネクション上で最後にデータが送信されてからの時間 <num3>: 別のバックエンドコネクション上で最後にデータが送信されてからの時間
Retries	サーバがバックエンドノードへのコネクションを試行する回数

データ項目	説明
SLM	使用しているサービスレベルモニタリングクラス
VSBW	仮想サーバが使用した帯域クラス
PoolBW	プールが使用した帯域クラス
Code	バックエンドノードにより返されたレスポンスコード(プロトコルによって異なる)
リクエスト	実行されたリクエストの概要。HTTP の場合はリクエストされた URL

注: 追加の情報がコネクションに適用可能と識別された場合には、追加のフィールドが表示されることもあります。

show (表示)の引数として指定できるオブジェクトタイプは、以下のとおりです。

action、authenticator、bandwidth、ca、certificate、
clientcertificate、connections、event、global、info、java、
licensekey、monitor、persistence、pool、protection、rate、rule、
slm、trafficip、virtualserver

- **stats:** トラフィックマネージャの様々なオブジェクトのカウンタと統計情報を表示します。以下に例を挙げます。
 - stats VirtualServer Web - 仮想サーバ「Web」の統計情報を表示します。
 - stats Pool - 全てのプールの統計情報を表示します。
 - stats * - 全ての統計情報を表示します。

以下のように、オブジェクト名のみを指定して **stats** を使用することもできます。この場合、指定した条件に一致する全てのオブジェクトの統計情報が表示されます。

- stats Staging - 「Staging」という名前の全てのオブジェクトの統計情報を表示します。
- stats * - 全ての統計情報を表示します(時間がかかる場合があります)。

指定できる統計オブジェクトは以下のとおりです。

authenticator、bandwidth、event、interface、node、perpoolnode、
pool、protection、rate、rule、session、slm、slmpernode、ssl、
system、trafficip、virtualserver、webcache、zxtm

- **help:** ヘルプトピックを表示します。以下に例を挙げます。
 - `help overview` - コマンドラインインターフェースの概要を表示します。
 - `help syntax` - コマンドおよびデータを入力するときの構文について説明します。
 - `help output` - コマンドの出力フォーマットについて説明します。
 - `help commands` - 全てのコマンドについて説明します。
 - `help classes` - 全ての API クラスについて説明します。
 - `help <class>` - 特定クラスのメソッドを表示します(例: `help Pool`)。
 - `help <cmd>` - 特定のコマンドについて説明します(例: `help Pool.addNode` または `help show`)。
- **rule:** トラフィックマネージャの TrafficScript ルールのアップロード、ダウンロード、および構文チェックを行います。
- **getbackup:** トラフィックマネージャからバックアップをダウンロードします。
- **putbackup:** バックアップファイルをトラフィックマネージャにアップロードします。
- **download:** オフラインで編集またはアーカイブするために、トラフィックマネージャから特定の設定ファイルをダウンロードします。
- **upload:** 特定の設定ファイルをトラフィックマネージャにアップロードします。
- **edit:** トラフィックマネージャから特定の設定ファイルをダウンロードして編集し、変更後のファイルをアップロードします。
- **watch:** コマンドを繰り返し実行し、出力を監視します。ログの監視や統計情報の記録など、変化をモニタリングする場合に役立ちます。以下に例を挙げます。

```
admin@127.0.0.1 > watch 10 stats webcache
```

これにより、全ての Web キャッシュの統計情報が表示されます。この情報は 10 秒ごとに更新されます。

CLI のスクリプティング

通常、CLI は、コマンドをキーボードから入力して即座に実行できる「インタラクティブ」モードで動作します。

ユーザ入力を行わずに、CLI コマンドをスクリプトとして実行することも可能です。例えば、以下のスクリプトは、Web キャッシュの統計情報を出力し、キャッシュから一部の項目をクリアし、その後再度 Web キャッシュの統計情報を出力します。

```
System.Stats.getWebCacheEntries
System.Cache.clearCacheContentItems "Intranet" "http"
"www.example.com" "*/product/*"
System.Stats.getWebCacheEntries
```

このスクリプトをローカルマシン上のファイルに保存し、以下のように実行します。

```
$ ZEUSHOME/zxtm/bin/zcli /home/user/webcache-script
```

CLI により、このファイルが読み取られ、スクリプト内の各コマンドが順番に実行されます。スクリプトが適切に完了した場合、終了コード 0 (ゼロ) が返されます。コマンドの実行中にエラーが発生した場合、スクリプトの実行が中止され、CLI はゼロ以外の戻り値で終了します。ただし、`--continue` がコマンドラインで指定されている場合、処理中に発生したエラーに関係なく、スクリプトは続けられます。この場合の戻り値は、CLI が何らかの内部エラー(不適切なパラメータの使用など)に直面した場合を除き、0 のままです。

スクリプトを完全に自動化するには、ユーザ名とパスワードを指定する必要があります。これは以下の 2 通りの方法で行えます。

1. パスワードをファイルに保存し、このファイルを `zcli` スクリプトに渡します。

```
ZEUSHOME/zxtm/bin/zcli --user username --passfile
/home/user/password /home/user/webcache-script
```

パスワードが含まれるファイルは、他のユーザに読み取られないようアクセス権を適切に制限する必要があります。

2. 以下のように、`connect` コマンドをスクリプトの先頭に追加します。

```
connect localhost 9090 username password
System.Stats.getWebCacheEntries
System.Cache.clearCacheContentItems "Intranet" "http"
"www.example.com" "*/product/*"
System.Stats.getWebCacheEntries
```

スクリプトの出力

スクリプトとして実行される CLI コマンドの出力は、インタラクティブモードの CLI コマンドの場合と多少異なります。インタラクティブモードでは、出力は読み取りやすいようフォーマットされて、スペースが挿入されたり、必要に応じて複数の行にわたったりすることもあります。スクリプトモードでは、各コマンドの出力は 1 行に表示され、スペースの挿入やフォーマットは行われません。

出力フォーマットは JSON フォーマットのため、他のプログラムでのインポートや解析を簡単に行えます。複数の入力(複数のプール名など)を取るコマンドの出力はテーブル形式で表示されないため、各項目がどの結果に対応しているのかが分かりにくくなります。

例えば、同じコマンドをインタラクティブセッションとスクリプトセッションで実行したときの出力を比較します。

インタラクティブモード:

```
admin@127.0.0.1 > Pool.getNodes [ pool1 pool2 ]
{ pool1: [ "serverA:80", "serverB:80" ],
  pool2: [ "serverC:80", "serverD:80" ] }
```

スクリプトモード:

```
$ ZEUSHOME/zxtm/bin/zcli --user username --passfile
/home/user/password
/home/user/getnodes
[["serverA:80","serverB:80"],["serverC:80","serverD:80"]]
]
```

読み取りやすい出力にするには、`--formatoutput` 引数を指定して `zcli` を実行します。

CHAPTER 27 zconf を使用したきめ細かい設定のインポート/エクスポート

はじめに

非常に多い数のトラフィックマネージャを定期的にプロビジョニングする場合、このプロセス中に行う作業の重複を最小限に抑えることにより、配備に要する時間的なコストを削減することができます。このプロセスでは、ポリシーの標準セットまたは設定の共通要素を多くのサービスに展開することになる可能性が高く、また多くのトラフィックマネージャに対して同じ設定の変更を行わなければならない場合もあります。

ポリシーや設定ファイルは、何らかの変更管理データベースを使用して保存することができます。ソースコードだけでなく、このような設定ファイルに対する変更を保存し、管理することのできるツールの一例として、**Subversion (SVN)**があります。トラフィックマネージャには、詳細レベルの設定のエクスポートおよびインポートを行ったり、これらの(部分的な)設定を **tar** アーカイブに保存したりできる、**zconf** というツールが用意されています。**tar** アーカイブは、必要に応じて抽出し、いずれかの変更管理データベースにチェックインすることができます。

zconf の使用

zconf ユーティリティは `$ZEUSHOME/zxtm/bin` にあります(`$ZEUSHOME` はトラフィックマネージャのインストール先ディレクトリ)。`$ZEUSHOME` が正しく設定されていないと、**zconf** ユーティリティが正常に機能しないことに注意してください。このコマンドラインユーティリティには、ヘルプシステムが内蔵されています。引数を指定せずにこのプログラムを実行すると、以下のような出力が得られます(以下の例はトラフィックマネージャ仮想アプライアンスのものです)。

```
root@ubuntul1:~# $ZEUSHOME/zxtm/bin/zconf

ERROR: No command specified.

Usage: /opt/zeus/zxtm/bin/zconf [COMMAND] [OPTIONS]
[ARGUMENTS]
Commands:

    copy
    diff
    export
    getdefaultfilter
    help
```

```
import
info
list
replicate
version

For more information on a given comment execute:

/opt/zeus/zxtm/bin/zconf help <COMMAND>
```

この出力には、コマンドのリストが示されます。ユーティリティの動作については、各コマンドのヘルプをお読みください。

完全バックアップのエクスポート

トラフィックマネージャのクラスタ設定の完全バックアップを行う場合、zconf の **export** コマンドを使用します。デフォルトでは、**export** コマンドの出力は **STDOUT** に出力されます。出力をファイルに保存するには、**--file** 引数を使用するか、出力を **STDOUT** からファイルにリダイレクトします(以下の例を参照)。多くの Linux または Solaris システムでは、その後 **tar** コマンドを使用して **tar** アーカイブのコンテンツを抽出することができます(subversion リポジトリにチェックインするためなど)。

```
root@ubuntul1:~# $ZEUSHOME/zxtm/bin/zconf export >
backup.tar
root@ubuntul1:~# ls
backup.tar
root@ubuntul1:~# tar xf backup.tar
root@ubuntul1:~# ls
backup.tar  conf
root@ubuntul1:~# ls conf
actions          groups           monitors        security
users
activitymonitor  licensekeys     PARTIAL        services
VERSION_8.0
commkey          locations        rules           settings.cfg
zxtms
events          locations.cfg    scripts        TIMESTAMP
root@ubuntul1:~#
```

設定のリスト表示

設定のバックアップはプレーンテキストファイルに保存され、ディレクトリツリーとして整理されます。zconf の `list` コマンドを使用すると、現在稼働中のトラフィックマネージャの設定ファイルと、設定のバックアップまたはバックアップアーカイブのリストを表示することができます。例えば、現在のトラフィックマネージャに設定されている仮想サーバのリストを取得する場合、以下のように入力します。

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf list vservers
root@ubuntu1:~#
```

この出力には、`vservers` ディレクトリが空であること、すなわち設定されている仮想サーバがないことが示されています。設定アーカイブ内の仮想サーバのリストを表示する場合は、以下のように入力します。

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf list vservers
                                --conf-path backup.tar
monitors/Full HTTP
pools/Bar
vservers/Foo
```

`list` コマンドの `glob` 引数として `vservers` を指定しましたが、プールおよびモニタのリストも出力に含まれています。これは、その仮想サーバを想定どおりに動作させるために必要な全ての依存オブジェクトが出力されるためです。

部分インポート

現在のディレクトリに設定アーカイブ `tar` アーカイブがあるとします。(例えば「`Foo`」という名前の)仮想サーバとその全ての依存オブジェクトをインポートするには、以下のコマンドを使用します。

```
zconf import --include vservers/Foo backup.tar
```

このインポートにより、設定がクラスタの他のメンバに複製されます。

注: 同様の機能は、管理 UI の **System** (システム) > **Backups** (バックアップ) > **Partial Backups** (部分バックアップ) ページからも利用できます。詳細については、CHAPTER 7 の「*System* (システム) > *Backups* (バックアップ) > *Partial Backups* (部分バックアップ)」セクションを参照してください。

CHAPTER 28 マルチサイトクラスタの管理

はじめに

本章で説明するマルチクラスタ管理機能は、トラフィックマネージャが備える高度な機能をベースに、分散した複数の物理、仮想、またはクラウドベースのトラフィックマネージャを管理するためのサポート機能を追加して作成されています。これらのクラスタの地理的な場所、稼働しているプラットフォーム、トラフィックマネージャインスタンスの数は様々です。マルチクラスタ管理機能では、これらの各トラフィックマネージャでホストされている特定のサービスを1箇所から集中管理することができます。

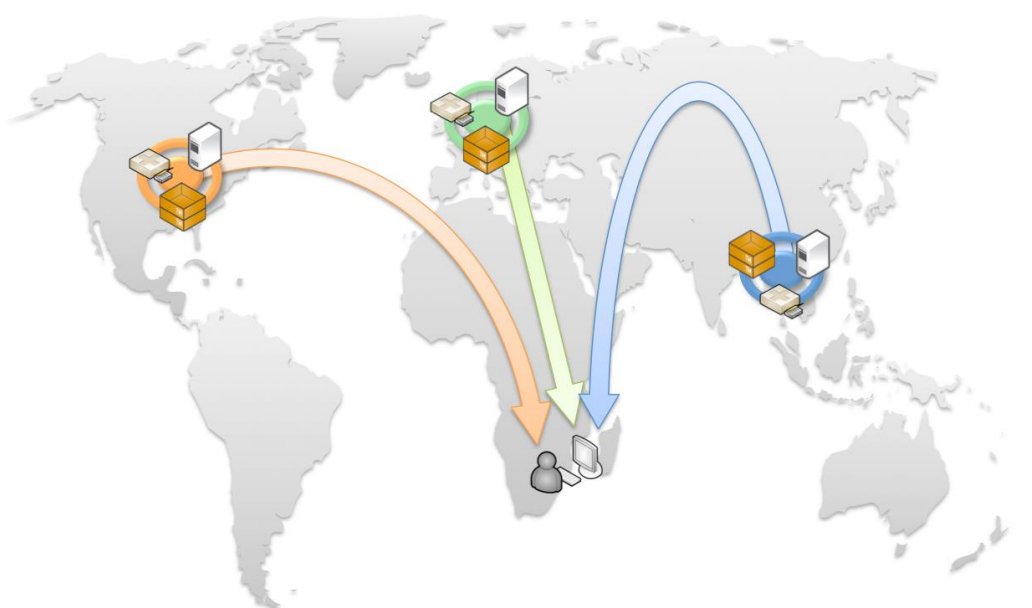


Fig 71. 世界規模のトラフィックマネージャクラスタインフラストラクチャの集中管理

本章では、トラフィックマネージャシステムのロケーション固有の設定を維持しながら、世界各地に分散したトラフィックマネージャインフラストラクチャを単一クラスタとして集中管理する方法について説明します。

世界の複数の拠点でホストされるサービスにわたって地理を意識した負荷分散を行うグローバルサーバ負荷分散(GSLB)とこの機能を混同しないでください。トラフィックマネージャへの GSLB の実装については、下の CHAPTER 29 で説明します。

本章では、本マニュアルの他のセクションで説明しているトラフィックマネージャのユーザインターフェースおよび一般的な設定コンセプトに関する知識があることを前提としています。先に進む前に、CHAPTER 7 を参照することを強くお奨めします。

アクティブ化と非アクティブ化

マルチサイト管理機能は、管理 UI の **System** (システム) > **Traffic Managers** (トラフィックマネージャ) ページから有効化できます。 *Multi-Site Management* (マルチサイト管理) セクションで **Confirm** (確認) チェックボックスを選択して **Enable** (有効化) をクリックすると、トラフィックマネージャがこのモードでアクティブになります。

同様に、同じ方法でマルチサイト機能を非アクティブにすることにより、トラフィックマネージャを元の状態に戻すことができます。この場合、まず定義されているロケーション設定のいずれかを全てのトラフィックマネージャのマスタ設定として定義する必要があります。その他全てのロケーション設定は、それぞれに保存されているロケーション固有の設定値とともに削除されます。マスタにするロケーションをドロップダウンリストから選択し、**Confirm** (確認) チェックボックスを選択して **Disable** (無効化) をクリックします。

注: ロケーション設定が削除されても、GLB ロケーションは非アクティブな状態で残ります。

主なコンセプト

ロケーション設定

基本的に、ロケーション設定はトラフィックマネージャのグループです。これらのトラフィックマネージャは、場合によっては特定の物理的な場所に存在し、クラスタ全体において同じサービスをホストすることが求められます。ロケーション設定は、**Catalogs** (カタログ) > **Locations** (ロケーション) ページで定義し、特定ロケーションに存在するものとして各トラフィックマネージャを指定します。

管理 UI 全体のサービスおよびその他の設定要素をロケーション固有にすることができます。ロケーション固有にしたサービスおよび設定は、そのロケーションに存在するものとして指定されたトラフィックマネージャでのみ使用されます。

このコンセプトにより、世界各地に存在する全てのトラフィックマネージャアプリケーション配信リクエストを単一の集中管理システムに集めることができるようになります。この環境では、構成の複雑さに関わらず、単一の管理 UI から全てのサービスを設定し、モニタリングすることができます。

注: トラフィックマネージャは必ずしも物理的に特定の場所に存在する必要はありません。GLB ロケーションとは対照的に、ロケーション設定のコンセプトは、主にトラフィックマネージャクラスタ内での仮想的なグループ化を可能にするためのメカニズムです。

クラスタ

マルチサイト構成におけるクラスタは、世界中で 1 箇所以上の場所に配備され、集中管理されるトラフィックマネージャのグループです。これによってフォールトトレランスを実現しながらアプリケーションサービスが提供されます。

トラフィックマネージャサービスの設定は、クラスタ内の全てのマシンに複製されます。マルチサイト管理では、非マルチサイト環境と同様に設定の多くが共有されますが、さらにこの機能が拡張され、特定のロケーションに存在するものとして指定されたトラフィックマネージャでのみアクティブになるロケーション固有の設定を行うことができます。これにより、一種のサブクラスタ化、すなわちローカルサービスの提供が可能になります。クラスタでは、トラフィックマネージャの管理 UI を使用して設定全体を編集でき、編集した内容はその他全てのトラフィックマネージャに複製されます。

各ロケーションに固有の設定(ローカル DNS サーバなど)も特定のマシンに固有の設定(SNMP 設定など)も、システム設定のうちのほんの一部です。ロケーション固有のシステム情報を設定するには、そのロケーションに含まれるいずれかのトラフィックマネージャの UI を使用します。この方法で設定した情報は、同じロケーションに定義されている他のトラフィックマネージャに複製されます。マシン固有のシステム情報を設定するには、そのマシンの UI を使用する必要があります。

重要: リモートのトラフィックマネージャをクラスタに組み込む場合、最初にそのトラフィックマネージャに**外部 IP アドレス**を指定する必要があります。また、新しいマシンを追加する前に、このアドレスと、特にポート番号 **9080** と **9090** をクラスタ全体で使用できるか確認することをお奨めします。これを怠ると、クラスタ内での設定の複製時に問題が生じる可能性があります。外部 IP の設定方法の詳細については、**424** ページの「トラフィックマネージャのロケーションの設定」を参照してください。

デプロイシナリオ

注: このセクションで使用されている「ロケーション」という言葉は**ロケーション設定**を意味します。さらに、全てのシナリオは、外部ロケーションへの適切な IP/ポート接続性を引き受けます(上のパラグラフを参照)。

第 2 のトラフィックマネージャロケーションの作成と管理

サービスの利用者増加や高度化に伴い、世界の他の場所に並行して稼働する第 2 のトラフィックマネージャを配備する必要性が生じる場合があります。

- まず、新しいロケーションの新たな各トラフィックマネージャに必要なインストールと初期設定を行います。ただし、既存のトラフィックマネージャクラスタに

組み込むと、追加する各マシンにクラスタ設定が複製され、システム固有でない設定が上書きされることに注意してください。

- 各トラフィックマネージャの管理 UI から、**Join a Cluster** (クラスタに組み込む) ウィザードを実行して、既存のトラフィックマネージャクラスタに接続します。
- あらかじめ既存のクラスタに新しい第 2 のロケーション設定を定義しておくか、ウィザードの手順 4 で「**new location**」オプションを指定します。ウィザードで新しいロケーションを指定しても、ロケーション定義の *position* 設定に値は適用されません。この設定は手動で行う必要があります。
- 以降追加する各トラフィックマネージャについては、このウィザードの手順 4 で、新たに作成したロケーションを選択します。

新しい各トラフィックマネージャがクラスタに追加され、全ての設定がコピーされますが、第 2 のロケーション設定に存在するものとして指定されます。ロケーション固有としてこれまでに行われた、または今後行われる設定の変更は、クラスタ全体に適用されます。

マルチサイトクラスタへの新しいトラフィックマネージャの追加

トラフィックマネージャが管理する任意のロケーションにトラフィックマネージャを追加するには、追加するマシンの管理 UI から **Join a Cluster** (クラスタに組み込む) ウィザードを実行します。これにより、クラスタの設定が新しいトラフィックマネージャにインポートされ、管理 UI には、追加したトラフィックマネージャが管理対象マシンとして表示されます。

複数の既存トラフィックマネージャクラスタのマージ

通常のクラスタ組み込み操作では、組み込まれるマシンにクラスタの設定が継承され、既存の設定は全て削除されます。このシナリオでは、複数の設定をマージするため、全てのマシンの設定が維持されるよう十分に注意する必要があります。

トラフィックマネージャには、きめ細かい設定のインポート/エクスポート操作を実行する「**zconf**」コマンドラインツールが用意されています。このツールを使用すると、既存のサービスに悪影響を与えることなく、セカンダリクラスタの個々のサービスをマスタクラスタの設定にインポートすることができます。また、**zconf** 内蔵の「**dry-run**」モードでインポート手順を実行すると、あらかじめ予測される結果を全て確認できるというメリットもあります。完全な使用構文および詳細については、**zconf** から直接利用可能なオンラインヘルプを参照してください。

このモードの目的は、サービスのダウンタイムを最小限に抑えることです。ただし、実際にはトラフィック IP アドレスが転送されるときにサービスが中断される可能性があるため、インポート操作は Web トラフィックが最小になる時間帯に行うのが最善です。

いかなる場合も、この手順を実行する前に、全てのクラスタの設定をバックアップすることを強くお奨めします。また、期待どおりの結果を得るために、以下の操作を最初に重要性の低いシステムで試すことも役立ちます。

以下の手順に従います。

- 全てのクラスタ内の全トラフィックマネージャで最新のトラフィックマネージャソフトウェアが実行されていることを確認します。いずれかのクラスタをマスタクラスタとして指定します。
- マスタクラスタで新しいセカンダリロケーションを設定します。
- ファイル名の競合を最小限に抑えるために、全てのクラスタの仮想サーバ、ルール、プールなどの名前を、先頭にロケーション名を付けるなどしてユニークに識別できる名前に変更します。例えば、「Intranet」という名前を「london-Intranet」に変更します。これは必須ではありませんが、設定が競合する可能性を減らすことができます。zconfではデフォルトで、インポートの実行時に同じ名前の既存の設定ファイルが上書きされますが、これは切り替え可能です。
- **System (システム) > Backups (バックアップ)** ページを使用してリモートクラスタの設定をエクスポートし、結果の **tar** アーカイブをマスタクラスタにコピーします。
- **zconf** を使用して、設定のバックアップから仮想サーバ(/vservers)ディレクトリをインポートします。zconfは、自動的に各仮想サーバファイル内の依存関係に従い、関連するサポートファイルをインポートします。このときの唯一の注意点は **Java Extentions** の使用であり、手動で/jars および/servlets ディレクトリにインポートする必要があります。
- ここで、インポートしたこれらのサービスについて、新たに定義したリモートロケーションに対してのみ有効化のマークを付ける必要があります。これを行うには、**Virtual Server (仮想サーバ) > Edit (編集)** ページで、**Enabled: (有効化:)** 設定の横にある **Edit setting by location (ロケーション別の設定の編集)** アイコンをクリックします。この仮想サーバがリモートロケーションで有効化され、その他全てのロケーションでは無効化されていることを確認します。
- トラフィック IP グループが使用されている場合、2つのクラスタに同じ IP アドレスが設定されていると、サービスが競合する可能性があります。マスタクラスタで同じ IP アドレスを設定するために、既存のリモートクラスタの IP アドレスをオフラインにすることが必要になる場合があります。リモートトラフィックマネージャをマスタクラスタに追加するときは、まずそのトラフィックマネージャをリモートクラスタから削除する必要があります。これにより、リモート IP グループからも削除されます。
- 各クラスタが複数のトラフィックマネージャマシンで構成されている場合、各クラスタから1つのトラフィックマネージャを切り離してマスタクラスタに組み込みます。このとき、該当する定義済みのリモートロケーションに存在するものと

してそのトラフィックマネージャを指定します。システム固有の設定は、その設定の対象となるマシンの管理 UI から直接適用する必要があることに注意してください。

- このトラフィックマネージャを新たに作成したトラフィック IP グループに追加できます。その後、サービスを再開します。
- 次に、残りの各トラフィックマネージャをリモートクラスタから切り離してマスタクラスタに組み込むことができます。このとき、該当するロケーションに存在するものとしてそのトラフィックマネージャを指定し、適切なトラフィック IP グループに追加します。各トラフィックマネージャは、完全にマージされた設定を継承しますが、指定されたロケーションのサービスにのみレスポンスを返します。

設定

ロケーションの設定

ロケーションは **Catalog** (カタログ) > **Locations** (ロケーション) ページに表示され、このページで変更を行います。また、**System** (システム) > **Traffic Managers** (トラフィックマネージャ) ページの *Manage Locations* (ロケーションの管理) リンクをクリックしても表示されます。このページには、定義されているロケーションのリストが表示され、ロケーションをリストに追加またはリストから削除することができます。

注: グローバル負荷分散機能を持つ製品では、「GLB Locations」という追加のセクションが表示されます。詳細については、下の **CHAPTER 29** を参照してください。

各ロケーションは、単純に名前で定義されます。ロケーション名または関連する **Edit** (編集) リンクをクリックすると、ロケーションの設定を変更したり、ロケーションを完全に削除したりできます。

新しいロケーションを追加するには、メインリストの下 *Add new Location* (新規ロケーションの追加) セクションを使用します。新しいロケーションを作成するときは、既存のロケーションをテンプレートとして使用します(既存のロケーションがない場合を除く)。この既存ロケーションに「基づく」メカニズムにより、テンプレートとなるロケーションの全てのロケーション固有の設定が新しいロケーションにコピーされます。

ロケーションを削除するには、関連する **Edit** (編集) ページの下部にあるセクションを使用します。ただし、1 台以上のトラフィックマネージャで使用中のロケーションを名前変更したり削除したりすることはできません。このような操作を行う前に、まずトラフィックマネージャとこのロケーションとの関連付けを解除する必要があります。

トラフィックマネージャのロケーションの設定

System (システム) > **Traffic Managers** (トラフィックマネージャ) ページでは、クラスタ内の各トラフィックマネージャに対して、新たに以下の 2 つの設定を行えます。

- **Location:** このトラフィックマネージャのロケーション設定をドロップダウンリストから選択します。
- **External IP:** トラフィックマネージャのホスト名がグローバルに名前解決出来ない場合、クラスタメンバ間の通信に使用する、外部から利用可能な IP アドレスを指定する必要があります。

これらの設定を変更する場合は、まず **Update** (更新) ボタンの横にあるチェックボックスをクリックして確認する必要があります。

ロケーション固有の設定

マルチサイトモードでのトラフィックマネージャによる設定の扱いが大きく異なることは、各設定の横に新しいアイコンが追加されることから分かります。



Fig 72. Location アイコン

このアイコンをクリックすると、設定が、ロケーション設定ごとの複数のインスタンスに分割されます。このアイコンを再度クリックすると、全てのロケーションで使用される単一のインスタンスに戻ります。

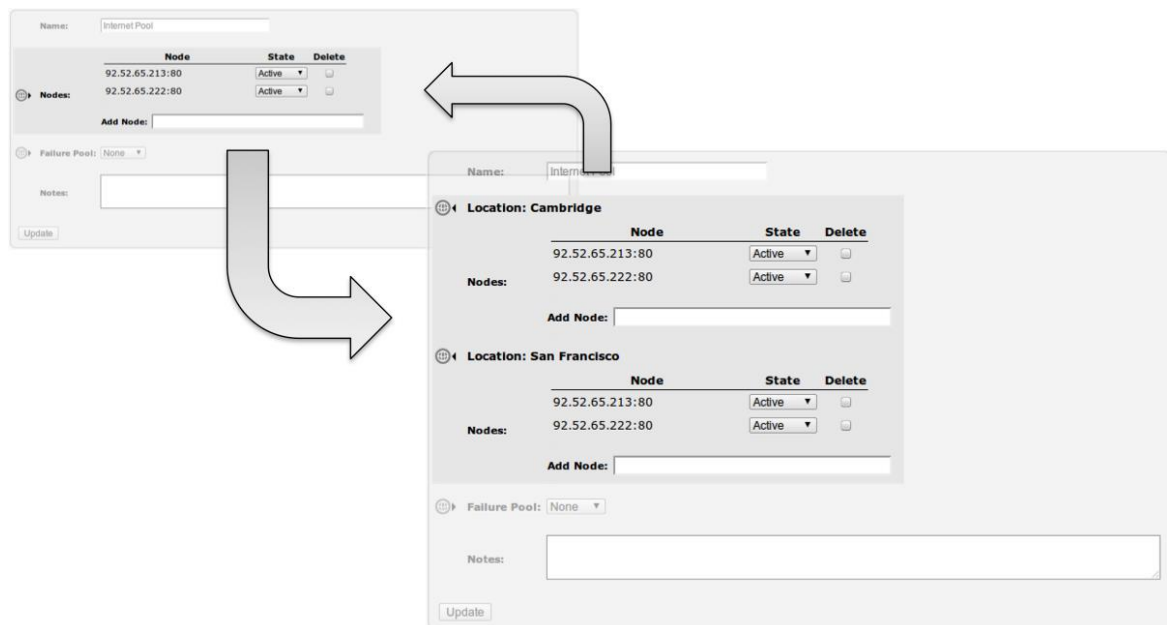


Fig 73. ノードリストのロケーションモードと通常モードの切り替え

このメカニズムにより、全てのクラスタメンバに適用されるグローバル設定を維持しながら、個々の設定をロケーションごとに指定することができます。

ホームページの変更点

マルチサイトクラスタ機能が有効化されている場合、通常とは異なる UI ホームページが表示されます。新しい方式は、トラフィックマネージャの想定運用環境に適したものとなっており、以下に示すいくつかのセクションが表示されます。

- **Locations:** ここには、トラフィックマネージャのロケーション設定が表示されません。定義されていても使用されていないロケーションは表示されません。ロケーション名をクリックすると、そのロケーションの **Catalogs** (カタログ) > **Locations** (ロケーション) > **Edit** (編集) ページが表示されます。
- **Services:** ここには、*enabled* (有効化) のマークが付けられ、このトラフィックマネージャによって管理されているサービスのリストが表示されます。サービスは、仮想サーバ、ルール、プール、およびその他の関連オブジェクトを組み合わせたものとして考えることができます。サービス名をクリックすると、そのサービスの **Virtual Servers** (仮想サーバ) > **Edit** (編集) ページが表示されます。
- **Event Log:** これは、イベントログの簡単な概要です。イベントログは全てのトラフィックマネージャから書き込まれ、各エントリの横には対象のマシンの名前が表示されます。ログ全体を表示するには、*Examine Logs* (ログ確認) リンクをクリックします。

それぞれのロケーションおよびサービスは、以下の適切なステータスインジケータとともに表示されます。



正常に動作している
ことを示します。



警告を示します。



重大なエラーを
示します。

Fig 74. ステータスインジケータのアイコン

警告は重大な問題ではありません。すなわち、ロケーションまたはサービスは、性能が低下した状態でまだ動作しています。エラーには、ロケーションまたはサービスの動作を妨げている問題を解決するためのアクションが必要です。これらの状態は、ステータスアプレットに反映されます。警告またはエラーの場合、ロケーション/サービス名をクリックすると、**Diagnose** (診断) ページに移動し、ここで問題の詳細な説明を確認することができます。

ワールドマップ



Fig 75. ワールドマップ

Activity (アクティビティ) > **Map** (マップ) をクリックすると、ロケーションとサービスが分かりやすく視覚化されたワールドマップが表示されます。このマップは、**Virtual Servers** (仮想サーバ) > **Edit** (編集) ページの *View traffic on World Map* (ワールドマップにトラフィックを表示) リンクをクリックしても表示できます。ここには、システムに定義されているロケーションの世界規模の概要が示されます。

マップはインタラクティブに設計されており、マップ上を移動したり、インフラストラクチャに必要な詳細レベルにズームインしたりするためのツールが用意されています。マウスの左ボタンを押しながらマップを所要の目的地にドラッグすることにより、任意の地理的位置を中央に表示することができます。下部にあるドロップダウンコントロールを使用すると、特定の仮想サーバのトラフィックを表示できます。デフォルトでは、全ての仮想サーバのトラフィックが表示されます。

色分けされた標的マークはロケーションを表します。また、マップ上に表示される小さいドットは特定のサービスに対するリクエストを送信したユーザを表し、リクエストしたサービスがホストされているロケーション別に色分けされて表示されます。役立つ情報を表示するには、**Connection Archiving** を有効化する必要があります(高度なコネクション解析を表示する場合にも必要です)。**Connection Archiving** が無効化されている場合、下部にある *connection archiving is disabled* (**Connection Archiving** が無効) リンクをクリックすると、有効化できます。このリンクをクリックすると、**System** (システム) > **Global Settings** (グローバル設定) ページに移動し、このページで **recent_conns** 設定に適切な値(500 など)を指定する必要があります(デフォルト値 0 の場合、コネクションアーカイブが無効化される)。

トラフィックの可視化

グローバル負荷分散と同様に、システムでマルチクラスタ管理機能をアクティブにすると、トラフィックマネージャの既存の診断および可視化ツールで追加のオプシ

ョンが使用可能になります。詳細については、CHAPTER 29 の「トラフィックの可視化」セクションを参照してください。

CHAPTER 29 トラフィックマネージャの DNS サーバ

トラフィックマネージャには、権威ドメインネームシステム(DNS)サーバ機能が内蔵されています。この機能により、トラフィックマネージャはゾーンファイルをホストし、DNS クエリに応答することができます。

グローバル負荷分散(GLB)のデプロイの場合、内部のトラフィックマネージャの DNS サーバを実装するか、トラフィックを外部 DNS サーバノードのプールに分散するかを選択できます。

DNS 用語に関しては、428 ページの「DNS の概要」で紹介しています。トラフィックマネージャクラスタを DNS サーバとして設定する方法については、437 ページの「DNS サーバの設定」を参照してください。

DNS の概要

はじめに

DNS は、インターネットの電話帳と考えることができます。ドメイン名に関する様々な情報の断片を検索するために使用します。例えば、Web ブラウザは、Web サーバに Web サイトをリクエストするために、どの IP アドレスに接続すればよいかわかる必要があります。同様に、電子メールを送信するときは、電子メールサーバは、メッセージを受信者に配信する役割を担っているメールサーバのドメイン名と IP アドレスの両方を知らなければなりません。この情報を見つける手段を提供するのが DNS です。

DNS のレイアウト

DNS のレイアウトはトップダウンツリー構造に似ています。この構造のルートは、ルートネームサーバのグループから成ります。ルートの下にはトップレベルドメインサーバがあり、「com」(「www.example.com」など)や「net」はどちらもトップレベルドメインの例です。トップレベルより下位では、DNS に「ネームサーバ」以外の明確な分類はありません。

権威の委任

DNS における「権威」は、特定ドメインに対するリクエストを処理するネームサーバの責任を表します。ネームサーバが、example.com の下位にある全てに対する DNS リクエストの処理に権威を持っている場合、このネームサーバには、上位のドメインサーバからこのような処理を行う権威が委任されています。トラフィックマネージャの内蔵 DNS 機能は、権威 DNS サーバの一例です。

名前解決

DNS を使用して「情報の断片を検索する」プロセスを「名前解決」と言います。例えば、Web ブラウザが特定 Web サイトの IP アドレスを見つける必要がある場合、ドメイン名を名前解決して IP アドレスを取得しようとします。電子メールサーバが特定ドメインへのメッセージ配信を担っているメールサーバを知る必要がある場合、「メール交換(MX)名」とその IP アドレスを名前解決しようとします。

ネットワーク設定の過程で、通常、コンピュータは少なくとも 1 つの「ネームサーバ」を認識し、DNS リクエストに対するレスポンスはこのネームサーバによって与えられます。コンピュータが DNS リクエストを別のネームサーバに送信するのは、特殊な状況¹²のみです。

リソースレコード

ドメイン名に関する各情報は、リソースレコードに保持されます。通常、リソースレコードはタイプによって識別されます。例えば、Web ブラウザが `www.example.com` の IP アドレスを知ろうとしている場合、`www.example.com` に一致する「A レコード」を問い合わせるクエリを送信します。メールクライアントが `john.smith@example.com` 宛てのメッセージをどこに送信すればよいかを知る必要がある場合は、`example.com` に一致する「MX レコード」を問い合わせるクエリを送信します。一般的に使用およびサポートされているリソースレコードタイプには、以下のものがあります。

- **SOA (権限の開始)** - 特定のドメインに関する役立つ情報を与えます。
- **NS (ネームサーバ)** - 特定ドメインに対して権威を持つメールサーバのドメイン名を示します。このリソースは、権威がどのように上位ドメインから下位ドメインへと委任されるかを決定します。
- **MX (メール交換)** - 特定ドメイン宛てのメールを受け入れるメールサーバのドメイン名を示します。
- **A (アドレス)** - ドメイン名の IPv4 アドレスが保持されます。
- **AAAA (アドレス)** - ドメイン名の IPv6 アドレスが保持されます。
- **CNAME (カノニカル名)** - ドメイン名のエイリアスが保持されます。
- **PTR (ドメインポインタ)** - IP から DNS 名への逆引きに使用されます。
- **SRV (サービスレコード)** - 指定されたサービスに使用されるサーバのホスト名とポート番号を定義します。

¹² このような状況として、テストを実施する DNS 管理者が意図的に行う場合などがあります。

- **TXT (テキストメッセージ)** - 任意の人間が読み取れるテキストメッセージが保持され、様々な目的で使用されます。

ゾーンファイル

通常、リソースレコードは、権威 DNS サーバ¹³上のテキストファイルに保存されます。ゾーンファイルには、ドメインに固有の一連のレコードが保持され、少なくとも 1 つの SOA レコードと NS レコードが含まれている必要があります。

トラフィックマネージャは、以下のフォーマット例に則ったゾーンファイルをサポートしています。

```
$TTL      604800      ; This TTL declaration applies to all
                        ; records defined below that do not define
                        ; their own TTL

; In this file, @ means the "current origin domain"

@          IN          SOA      ns.example.com. root.example.com. (
                                1          ; Serial
                                604800     ; Refresh
                                86400      ; Retry
                                2419200    ; Expire
                                604800     ; Negative Cache TTL
                                );

@          NS          ns
ns         A           192.168.0.1
@          NS          ns.example.com.
ns         A           192.168.0.254

@          IN          MX       1 host
@          IN          MX       2 mail2.example.net. ; note this is
                                                ; in another
                                                ; zone/domain

host       IN          A         192.168.0.2
                        AAAA      2001:db8:85a3:0:0:8a2e:370:7334

$TTL 1w2d3h4m5s      ; The TTL defined here overwrites the
                        ; setting above and applies to all the
```

¹³ DNS の実装によっては、それに代えてリソースレコードがデータベースに保存されることがありますが、トラフィックマネージャの DNS サーバはここで説明するゾーンファイル手法を使用します。

```

; records below it that do not define their
; own TTLs.

foo.bar                A      192.168.0.3 ; An example of a
                                ; relative address:
                                ; foo.bar.example.com.
foo.bar.example.com.  A      192.168.0.4 ; An example of an
                                ; absolute address

alias                  CNAME  host

_sip._udp              SRV    0 5 5060 sipserver
sipserver              3600   A      192.168.0.5 ; This record has
                                ; its own TTL (3600)

text1                  TXT    "This is a TXT record"
*.wildcards            TXT    "This is a wildcard TXT record"

```

トラフィックマネージャは、トップダウンのアプローチによってゾーンファイルを解析し、インデントが生じた先行ラインから継承します。

当初のデプロイからゾーンファイルを変更または更新した場合は、DNS サービスが変化を認識するように、「シリアル」カウンタを増加させる必要があります。

通常、ゾーンファイルには少なくとも 1 つの TTL(存続時間)定義が、具体的にはファイルの最上部に保持されています。トラフィックマネージャは、この値を使用して、DNS 関連情報が更新されるまでクライアントを待機させる時間の長さを決定します。ゾーンファイルの TTL の値には、以下のルールが適用され、監視されます。

- ゾーンファイルに TTL の値を定義しない場合、トラフィックマネージャは代わりに SOA 定義からネガティブキャッシュ TTL の値を使用します。ネガティブキャッシュ TTL は、通常、「否定的なレスポンス」の TTL として使用されます。この否定的なレスポンスとは、NXDOMAIN (ドメイン名が不明)および NODATA (ドメイン名は判明したが、特定タイプのレコードが存在しない) です。
- 同じドメイン名を持つレコードには、全て同じ値の TTL が定義されています。
- TTL は、秒数を表す一つの数字で指定することも、1w2d3h4m5s (この場合、1 週間 2 日 3 時間 4 分 5 秒を表す)の代替フォーマットで指定することもできます。このフォーマットのアルファベット要素には大文字と小文字の区別はなく、大文字と小文字を混合させて指定することもできます(例: 3W1h5M4s)。

名前解決プロセス

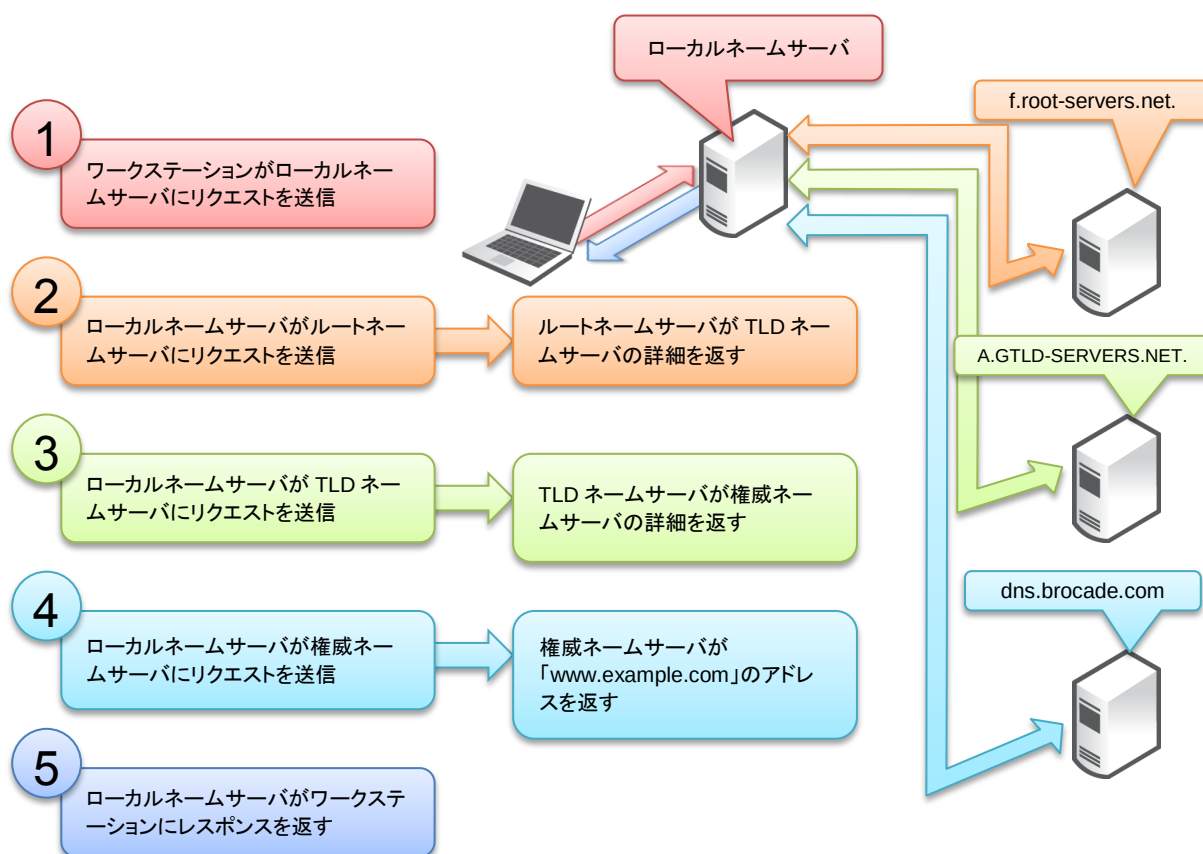


Fig 76. 名前解決プロセス

ここからは、ユーザのクライアントワークステーションが Web サイトの URL を正しい IP アドレスに名前解決する方法について、手順を追って説明します。

注: トラフィックマネージャの内蔵 DNS サーバ機能を使用している場合、このシナリオの「dns.example.com」が権威 DNS サーバとしてのトラフィックマネージャに相当することがあります。

1. ユーザがWebブラウザを起動し、アドレスバーにwww.example.comと入力した場合、ワークステーションはHTTPリクエストをWebサーバに送信する必要があります。ただし、その前に、Webサーバのドメイン名を名前解決しなければなりません。そこで、ワークステーションはIPアドレスを特定するために、DNSリクエストをローカルネームサーバに送信します。
2. ローカルネームサーバはルートヒントゾーンファイルを参照します。ルートヒントゾーンファイルには、ルートネームサーバのドメイン名とIPアドレスのリストが保持されています。次に、ローカルネームサーバはwww.example.comに対するDNSリクエストを、ルートヒントゾーンファイルに登録されているいずれかのルートネームサーバに送信します。「f.root-servers.net」は、DNSリクエストと、このルートネームサーバが権威を持っているゾーンのリストを比較します。この場合のルートネームサーバは「.」ドメインに権威を持っています。「.」のゾー

ンファイル内には、全てのトップレベルドメインの委任レコードがあります。
www.example.comのトップレベルドメインは「.com.」であるため、「f.root-servers.net.」は、「.com.」に権威を持っているネームサーバのドメイン名とIPアドレスを記述したDNSレスポンスをローカルネームサーバに返します。

3. A.GTLD-SERVERS.NET¹⁴は、(いくつかあるトップレベルドメインサーバのうち)「.com.」ドメインに権威を持つトップレベルドメインサーバとして登録されているため、ローカルネームサーバは同じリクエストをこのトップレベルドメインサーバに送信します。これに対し、A.GTLD-SERVERS.NETは、「example.com.」に権威を持つネームサーバのリストをレスポンスとして返します。
4. 「dns.example.com.」は「example.com.」の権威ネームサーバであるため、ローカルネームサーバはこの権威ネームサーバにリクエストを送信します。
「dns.example.com」のゾーンファイルには、「example.com.」の全てのリソースレコードが保持されているため、この権威ネームサーバから(1つ以上の「A」レコードの形で)IPアドレスが与えられます。
5. これで、ローカルネームサーバはwww.example.comの(1つまたは複数の)IPアドレスを取得できたため、DNSリクエストの最初の送信元であるワークステーションにこの情報を渡します。再帰的検索を行ったマシンはローカルネームサーバです。

クライアントが行うこのタイプのクエリは、ローカルネームサーバがDNSツリーを最上位から下位へと検索することから「再帰的」クエリと呼ばれます。「非再帰的」クエリを送信することも可能です。この場合、問い合わせられたドメイン名に権威を持っているネームサーバのみがそのドメイン名のIPアドレスを返し、クエリを受信したそれ以外のネームサーバは、そのクエリに対してレスポンスを返すことを拒否します。

通常、ローカルネームサーバは、エンドユーザに最終的なDNSの結果を返すだけでなく、中間の検索結果のキャッシングも行います。結果的に、全てのリクエストが前述のフルセットのDNSリクエストを必要とするわけではありません。このため、ローカルネームサーバは、「再帰的ネームサーバ」または「キャッシングネームサーバ」と呼ばれることがあります。

サポートされる DNS 機能

トラフィックマネージャのDNSサーバは、RFC 1034およびRFC 1035で定義されているDNSの技術仕様の大部分に適合するように設計されています。このセクションでは、実装されている機能と除外事項を説明します。

¹⁴ 「GTLD」は「Global Top Level Domain (グローバルトップレベルドメイン)」の頭字語です。

RFC 1034 準拠の実装機能

トラフィックマネージャの DNS サーバは以下の機能を実行します。

- 権威サーバの動作
- ドメイン名空間のデータベース
- 肯定的および否定的応答(NODATA、NXDOMAIN、SERVFAIL、FORMAT ERROR、REFUSED、NOT_IMPLEMENTED)
- 委任(referral とも言う)
- DNS データベースのクラス IN
- レコードタイプ ANY
- ワイルドカード DNS レコード(RFC 4592 も参照)

RFC 1035 準拠の実装機能

トラフィックマネージャの DNS サーバは以下の機能を実行します。

- サポートされているレコードタイプ
 - A
 - CNAME
 - MX
 - NS
 - PTR
 - SOA
 - TXT
- その他のサポートされているレコードタイプ
 - AAAA (RFC 3596 も参照)
 - SRV (RFC 2782 も参照)
- サポートされている DNSSEC レコードタイプ(RFC 4034 も参照)
 - DNSKEY
 - RRSIG
 - DS

- NSEC
- NSEC3
- NSEC3PARAM
- DNS の応答にはメッセージ圧縮が使用されます(RFC 1035 のセクション 4.4.4 も参照)。
- UDP および TCP は、IPv4 および IPv6 による転送プロトコルとして完全サポートされています。
- RFC 1035 のセクション 5 で定義されているとおり、ゾーンテキストファイル (「マスタファイル」とも言う) がサポートされています。トラフィックマネージャは、ゾーンファイルの以下の要素を処理することができます。
 - このセクションで説明されているサポート対象タイプの、リソースレコードの定義
 - \$TTL 指示文
 - \$ORIGIN 指示文
 - 丸括弧の使用および複数行レコードの定義
 - 「@」記号
 - セミコロン(;)に続くコメント
- RFC 1035 のセクション 3.5 のとおり、in-addr.arpa ドメインを使用した逆引きがサポートされています。

RFC 1034 に関する除外事項

以下の機能は実装またはサポートされていません。

- CS、CH、および HS のリソースレコードクラス
- 逆クエリ(逆クエリは RFC 3425 で廃止)
- ゾーン転送
- ステータスクエリ(実験)
- 完了クエリ(廃止)

RFC 1035 に関する除外事項

以下の機能は実装またはサポートされていません。

- \$INCLUDE 指示文

- リテラル文字または制御文字を定義するためのバックスラッシュ(\)演算子の使用。例えば、「\.」によるラベル内へのドット文字配置
- それぞれの D が、DDD と記述される 10 進数に相当するオクテット数字である \DDD。トラフィックマネージャは、結果として生じるオクテットをテキストと見なし、特別な意味ではチェックしません

その他の実装機能

トラフィックマネージャの DNS サーバは以下の機能を実行します。

- 大文字と小文字の違いは完全サポートされています。
- RFC 2308 の「Negative Caching of DNS Queries (DNS クエリのネガティブキャッシング)」は完全実装されています。
- EDNS(0) は RFC 6891 の「Extension Mechanisms for DNS (DNS 用拡張機構)」に準じて実装されています。トラフィックマネージャは、EDNS を使用して、512 バイトを超える UDP レスポンスサイズを処理します。さらにトラフィックマネージャは、EDNS の「DO」フラグを使用して、受信する DNS クエスチョンが DNSSEC サポートを示しているかを、RFC 3225 に準じて判断します。その他の EDNS 機能はサポートされていません。
- DNSSEC プロトコルは、以下の RFC での定義に準じてサポートされています。
 - RFC 4033、「DNS Security Introduction and Requirements (DNS セキュリティ拡張の紹介とその要件)」
 - RFC 4034、「Resource Records for the DNS Security Extensions (DNS セキュリティ拡張で使用するリソースレコード)」
 - RFC 4035、「Protocol Modifications for the DNS Security Extensions (DNS セキュリティ拡張のためのプロトコル修正)」
 - RFC 5155、「DNS Security (DNSSEC) Hashed Authenticated Denial of Existence (ハッシュ化された認証による存在否定 (NSEC3))」
- DNSSEC のサポートについては、以下の点に注意してください。
 - トラフィックマネージャは、署名済みの DNSSEC ゾーンファイルを解析することができます。
 - トラフィックマネージャは、規格に準拠した方法により、DNSSEC の署名済みレコードを使用して応答することができます。
 - トラフィックマネージャは、ゾーンが仮想サーバに関連付けられている場合に、ゾーンファイルで DNSSEC 署名の有効期限をチェックし、その後も 1 時間ごとに定期的にチェックします。署名の有効期限が切れているか、7 日以

内に有効期限が切れる場合は、アラートシステムを通じてイベントが発生します(CHAPTER 21 の「イベント処理およびアラート」を参照してください)。

その他の除外機能

以下の機能は実装またはサポートされていません。

- RFC 2317 に準じたクラスレスな委任はサポートされていません。
- RFC 3597 の「Handling of Unknown DNS Resource Record (RR) Types (未知の DNS リソースレコード (RR) タイプの取り扱い)」はサポートされていません。
- DNSSEC のサポートについては、以下の点に注意してください。
 - トラフィックマネージャには、ゾーンファイルに署名するツールは用意されていません。
 - トラフィックマネージャには、暗号化キーのライフサイクルを管理するツールは用意されていません。

DNS サーバの設定

このセクションでは、トラフィックマネージャを権威 DNS サーバとして設定する方法について説明します。

設定概要

以下のタスクを実行するには、管理 UI、REST API、SOAP API、またはコマンドラインインターフェースを使用します。それぞれについては、後で詳しく説明します。

- DNS ゾーンファイルをゾーンファイルカタログにアップロードする
- 「ゾーン」設定を作成し、「オリジンドメイン」を定義するとともに、アップロードした DNS ゾーンファイルをカプセル化する
- DNS を設定した仮想サーバを作成し、そのサーバに 1 つ以上のゾーンを割り当て、受信する DNS リクエストに対処する

ゾーンファイルは再利用したり、複数のゾーン設定に関連付けることができます。そうすることによって、複数のゾーンが異なるオリジンドメイン下の同じホストにアドバタイズできるようになります。さらに DNS 仮想サーバは、あらゆる数のゾーンを利用して、多くのドメインをカバーする権威 DNS サービスを提供することもできます。

注: 1 つのゾーンファイルを複数のゾーン設定に関連付ける場合は、ゾーンファイル内部のドメインオリジンを(\$ORIGIN 宣言を使用して)再定義しないことをお奨めします。

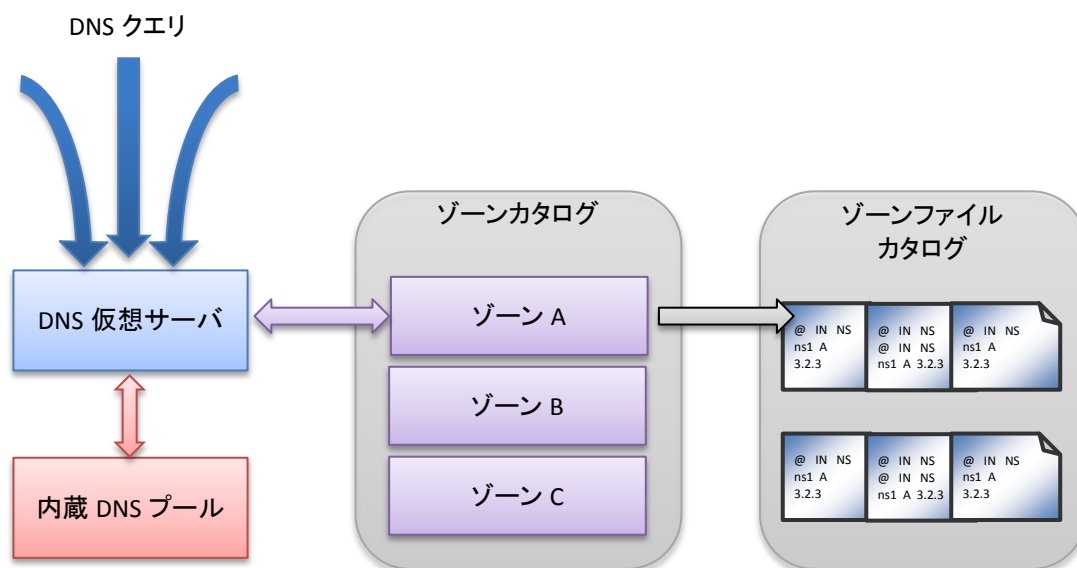


Fig 77. DNS サーバとして設定されたトラフィックマネージャ

この仕組みを利用して、トラフィックマネージャは、外部 DNS サーバノードのプールにリクエストを渡すのではなく、内蔵の DNS 機能を使用して DNS リクエストに応答します。

トラフィックマネージャへの DNS ゾーンファイルのアップロード

ゾーンファイルは、トラフィックマネージャがドメインに対する DNS クエリを受信したときに、トラフィックマネージャにどのような応答を返すかを指示するリソースコードを保持したテキストファイルです。各ゾーンファイルは、RFC 1034 および RFC 1035 で規定されたフォーマットに準じている必要があります。

トラフィックマネージャは、DNS ゾーンファイルカタログにゾーンファイルを保存します。新しいゾーンファイルをアップロードするには、**Catalogs (カタログ) > DNS Server (DNS サーバ) > Zone Files Catalog (ゾーンファイルカタログ)** を選択します。

注: ゾーンファイルがゾーン設定と関連付けられていない限り、トラフィックマネージャはゾーンファイルの内容を検証しません。検出された異常は、全てイベントログに追加されます。

トラフィックマネージャのゾーンの設定

トラフィックマネージャが、アップロードされたゾーンファイルに直接アクセスしたり、解析したりすることはありません。ゾーンファイルを使用するには、まずカタログ内のゾーンファイルを参照するゾーン設定を作成します。

各ゾーンには、関連付けられたゾーンファイル内のリソースレコードと併せてトラフィックマネージャが使用する、ドメイン「オリジン」も保持されています。つまり、ゾーンファイルにリストされた全てのドメインはこのオリジンに分岐し、`<domain>.<origin>`という形式を取ります。

例えば、オリジンが「`example.com`」の場合、ゾーンファイルで「`www`」および「`www.support`」のドメインを指定し、「`www.example.com`」および「`www.support.example.com`」に対するドメイン検索を作成することができます。

ゾーン設定を作成するには、**Catalogs** (カタログ) > **DNS Server** (DNS サーバ) > **Zones Catalog** (ゾーンカタログ)を選択します。ゾーンに対する名前とドメインオリジンを入力し、アップロードしておいた使用するゾーンファイルを選択します。**Create Zone** (ゾーンの作成)をクリックし、ゾーン設定を保存します。

ゾーンファイルを関連付けてゾーン設定を作成するときに、ゾーンファイルの内容に構文上の問題があると、トラフィックマネージャはイベントログにレポートを残します。

DNS 仮想サーバの設定

一般に、仮想サーバは、受信する DNS リクエストを処理し、そのリクエストをバックエンド DNS サーバノードのプールに渡すために作成します。その代わりに、内部 DNS サーバ機能を使用すると、DNS 仮想サーバを設定して内蔵 DNS プール「`builtin_dns`」を使用したり、DNS 検索に使用する所要のゾーン設定をリンクさせたりすることができます。

注:十分なサービス範囲を確保するためには、DNS (UDP)および DNS (TCP)の両方の仮想サーバをデプロイすることをお奨めします。

DNS 仮想サーバの作成

1. **Services** (サービス) > **Virtual Servers** (仮想サーバ)を選択します。
2. 「Create a new Virtual Server (新しい仮想サーバの作成)」セクションで、以下を入力します。
 - **Virtual Server Name** (仮想サーバ名):作成する仮想サーバの識別名です。
 - **Protocol** (プロトコル):ドロップダウンリストから「DNS (UDP)」または「DNS (TCP)」を選択します。

- **Port (ポート):** この仮想サーバが監視する受信ポート番号を選択します(通常は「53」)。
 - **Default Traffic Pool (デフォルトトラフィックプール):** トラフィックマネージャに内蔵された DNS 機能を使用するよう指示するには、「**builtin_dns**」を選択します。
3. **Create Virtual Server (仮想サーバの作成)**をクリックし、新しい仮想サーバを作成します。
 4. **Virtual Server edit (仮想サーバ編集)**ページで **DNS Server (DNS サーバ)**をクリックし、この仮想サーバによる DNS 検索に使用するゾーン設定を 1 つ以上選択します。
 5. 仮想サーバを有効化するには、**Virtual Server edit (仮想サーバ編集)**ページで **Enabled (有効化)**を「Yes (はい)」に設定するか、トラフィックマネージャのホームページの仮想サーバ名の隣にある **play(作動)**アイコンをクリックします。

注: デフォルトのトラフィックプール「**builtin_dns**」は、DNS (TCP)または DNS (UDP)の内部プロトコルを使用する仮想サーバでのみ利用でき、通常のプールリストには記載されないか、またはリストから選択することはできません。

Virtual Server (仮想サーバ) > Edit (編集) > DNS Server (DNS サーバ)ページでは、以下の設定も変更できます。

- **dns!edns_udpsize:** EDNS の UDP レスポンスでアダプタイズされるメッセージのサイズです。
- **dns!max_udpsize:** UDP レスポンスに許容される最大サイズです。
- **dns!verbose:** デバッグの効果を高めるには、この設定を有効化し、トラフィックマネージャに対してイベントログにより詳細な DNS 情報を含めるように指示します。製品を通常どおり使用する場合、この設定は無効化しておきます。
- **dns!rrset_order:** DNS レコードのレスポンス順序です。ゾーンファイル内で定義された順序で、同じ名前とタイプの DNS レコードを返すようにトラフィックマネージャに指示するには「**Fixed (固定)**」に設定します。ローテーションするラウンドロビンシステムを使用するようにトラフィックマネージャに指示するには、「**Cyclic (循環)**」に設定します。一連のレスポンスに対して、同じ名前とタイプの DNS レコードは、循環方式で返されます(レスポンスのたびに、各レコードが 1 つ進みます)。この方法によって、効率的な負荷分散が可能になります。

注: 場合によっては、キャッシングリゾルバによって、権威サーバから受信するレスポンスの DNS レコードの順序が不変となることがあります。

CHAPTER 30 グローバル負荷分散

この章では、トラフィックマネージャのグローバル負荷分散(GLB)について説明します。

注: トラフィックマネージャ製品によっては、この機能を利用できない場合があります。詳細については、サポート窓口にお問い合わせください。

概要と前提条件

GLB は、ビジネス継続性と世界各地のロケーションにわたる性能向上を実現するための機能です。この機能は、グローバルサーバ負荷分散(GSLB)として知られている技術をベースにしています。トラフィックマネージャは地理的負荷分散およびフェールオーバー機能を使用し、可用性、性能、およびユーザ定義のルールに基づいてこれらのロケーションへのトラフィックの分配を可能にします。

本章では、トラフィックマネージャのデプロイで GLB を計画および設定する方法について説明します。ここで説明する設定を行うためには、ネットワークングや DNS 管理を含め、標準的なシステム管理タスクを十分に理解する必要があります。DNS の基礎知識に関しては、428 ページの「DNS の概要」を参照してください。

GSLB の詳細な原理については、442 ページの「グローバルサーバ負荷分散について」を参照してください。 www.brocade.com および Brocade が提供するコミュニティ Web サイト(<http://community.brocade.com>)で公開されている基礎知識のホワイトペーパーおよびドキュメントも参照することをお奨めします。

グローバルサーバ負荷分散について

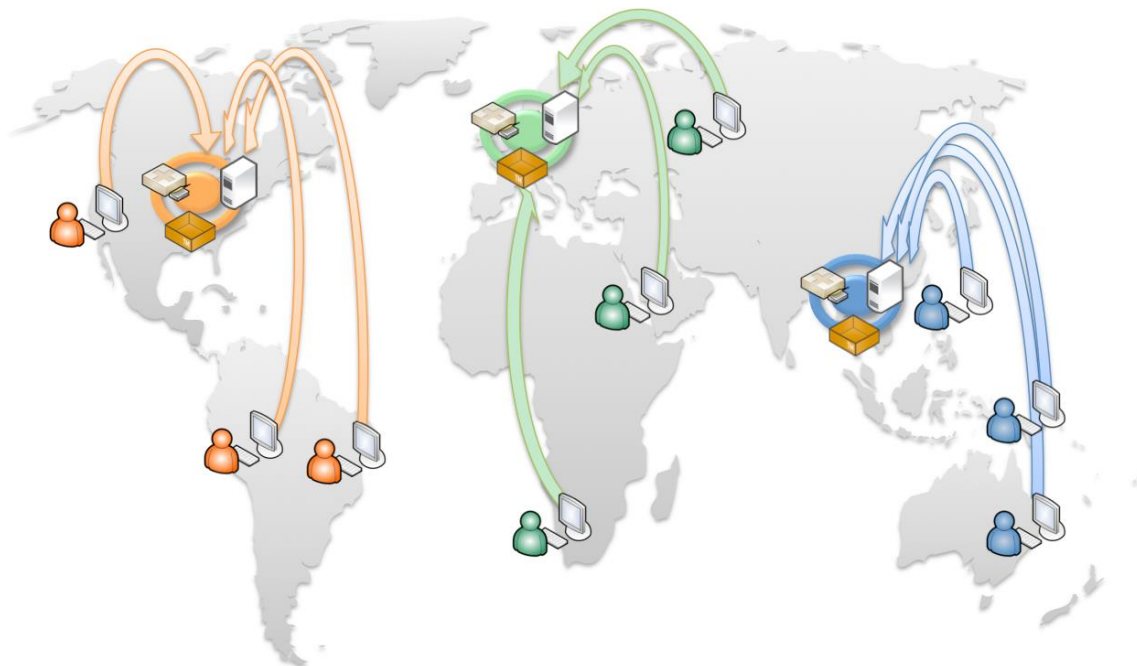


Fig 78. グローバルサーバ負荷分散

グローバルサーバ負荷分散(GSLB)は、サービスが複数のロケーションでホストされている場合に、クライアントの接続先となる地理的ロケーションをどのように決定するかを管理する手法です。

- 「アクティブ-パッシブ」(またはプライマリ-バックアップ)構成では、サービスごとに1つのロケーションがアクティブなロケーションとして指定されます。他のロケーションは、そのサービスについて待機状態になります。アクティブなロケーションが使用不可になった場合、いずれかのパッシブロケーションがアクティブになり、全てのクライアントのトラフィックがこのロケーションに送信されます。
- 「アクティブ-アクティブ」(またはマルチプライマリ)構成では、全てのロケーションが使用され、ロケーションの性能と近接性に基づいてクライアントのトラフィックがロケーションに負荷分散されます。

GSLB システムの主な目的は、ビジネス継続性、すなわち1つ以上のサービスロケーションが使用不可になっても、常にサービスを利用できる状態に保つことです。

GSLB の第2の目的は、より良いカスタマ経験を提供すること、すなわち各ユーザのトラフィックを、いくつかあるロケーションのうち最適なロケーションに負荷分散することです。ロケーションの選択は、ロケーションの性能と近接性に基づいて行うことができるため、クライアントのトラフィックを最も近くて最も性能の高いロケーションに送信することができます。この機能により、クライアントには可能な限り最高のサービスレベルが提供されます。

トラフィックマネージャ内での GSLB

トラフィックマネージャは、グローバル負荷分散(GLB)機能のような、GSLB 技術を実装しています。GLB 機能により、トラフィックマネージャは、エンドユーザが行う DNS 検索にロケーション認識を加えることができ、DNS 検索の結果がソースに依存するようになります。

GLB サービスは、DNS プロトコルを使用している任意の仮想サーバに対して追加することができます。この仮想サーバに DNS リクエストが届くと、トラフィックマネージャは、関連付けられたプールにそのリクエストを転送します。このプールは、内蔵された内部 DNS サーバプール(CHAPTER 29 の「トラフィックマネージャの DNS サーバ」を参照)であることもあれば、外部 DNS サーバのプールであることもあります。いずれのオプションを使用する場合も、複数の「A」レコードを返すためのプールバックエンドを、名前解決されるドメイン名に対する全ての IP アドレスを含め、全てのロケーションにわたって設定する必要があります。

名前解決されたドメイン名が、仮想サーバに設定されている GLB サービスのドメイン名と一致する場合、トラフィックマネージャは、GLB サービスの設定に基づいてリストをフィルタしてから、エンドユーザのロケーションに適した(1つまたは複数の)IP アドレスのみをエンドユーザに返します。

必須ではありませんが、ローカルの性能と可用性をモニタリングするためにトラフィックマネージャを各 GLB ロケーションにデプロイすることができます。また、他の場所にトラフィックマネージャをデプロイしなければならない場合でも、全てのモニタリングをリモートから行うことができます。トラフィックマネージャは、実際にデプロイされている場所とは関係なくピアとの間でモニタリング情報を交換するため、全てのトラフィックマネージャが連携して処理を調整することができます。

注: GLB サービスは「A」レコードのみをフィルタし、「AAAA」レコードはフィルタしません。DNS レスポンスに「AAAA」レコードが保持されている場合、トラフィックマネージャはそのレスポンスをそのまま通過させます。

デプロイ計画

トラフィックマネージャ の位置決め

トラフィックマネージャは、パブリック IPv4 アドレス空間の地理的ロケーションに関するデータベースを保持しており、権威 DNS サーバ、または別のバックエンド DNS サービス用のプロキシのいずれかとして機能します。トラフィックマネージャは、現在の DNS インフラストラクチャと並行して、またはその前に、またはそれに代えてデプロイすることができます。トラフィックマネージャは、クライアントから DNS リクエストを受け取ると、トラフィックマネージャが管理している内部または外部 DNS サービスのいずれかからレスポンスを特定します。その後、トラフィッ

クマネージャは多数の設定メトリクスに基づいて(地理的ロケーションも含め)レスポンスを変更してから、クライアントにそのレスポンスを返します。

トラフィックマネージャの内蔵 DNS 機能の詳細については、CHAPTER 29 の「トラフィックマネージャの DNS サーバ」を参照してください。

デプロイ手法

DNS ドメインは、親ドメインがネームサーバによって権威を持っていると指定された、一連のネームサーバを必要とします。グローバルトップレベルドメインなど、親ドメインがパブリックドメインの場合は、DNS レジストラを通じてご使用のドメイン用に権威ネームサーバを変更します。

GLB をホスト名で使用するには、以下のいずれかの方法でトラフィックマネージャをデプロイすることができます。

- インラインデプロイの場合、トラフィックマネージャは、そのホスト名を保持するドメインに対する権威 DNS サーバになります。親ドメインの委任レコードを変更することによって、そのドメインに対する全ての DNS トラフィックが、ご使用のトラフィックマネージャに送信されるようになります。トラフィックマネージャは、内蔵の DNS サーバ機能を使用してリクエストを処理するか、ご使用の既存 DNS サーバにリクエストを転送してから、GLB が設定されたホスト名でレスポンスを操作します。
- 並列デプロイでは、トラフィックマネージャにホスト名を保持するドメインに対する権威はなく、その代わりに、トラフィックマネージャが権威を持つ別の DNS ドメインで並列のホスト名を作成します。既存ドメインでは、他のドメインの並列のホスト名に対する CNAME を作成し、その並列のホスト名で GLB サービスを設定します。

ご使用の(権威のある)ネームサーバがホスト名に対する DNS リクエストを受信すると、ネームサーバは CNAME を使ってレスポンスを返し、それによりクライアントが、並列のホスト名に対する DNS リクエストをトラフィックマネージャから行います。インラインデプロイと同様に、トラフィックマネージャは内蔵の DNS サーバを使用してリクエストを処理するか、別の権威 DNS サーバにリクエストを転送します。その後、レスポンスは GLB サービスの設定に従って操作されます。

並列デプロイは、設定が複雑になりますが、多数のドメインを管理する場合や頻繁に変更を行うことが予測される場合は、このデプロイの方が適しています。

インラインデプロイ

インラインデプロイの場合、トラフィックマネージャは、ホスト名を保持するドメイン全体に対する権威 DNS サーバとして機能します。DNS リクエストは DNS 仮想サーバに受信され、内蔵の DNS サーバまたは DNS サーバの外部プールのいずれかである、関連付けられたプールによって処理されます。

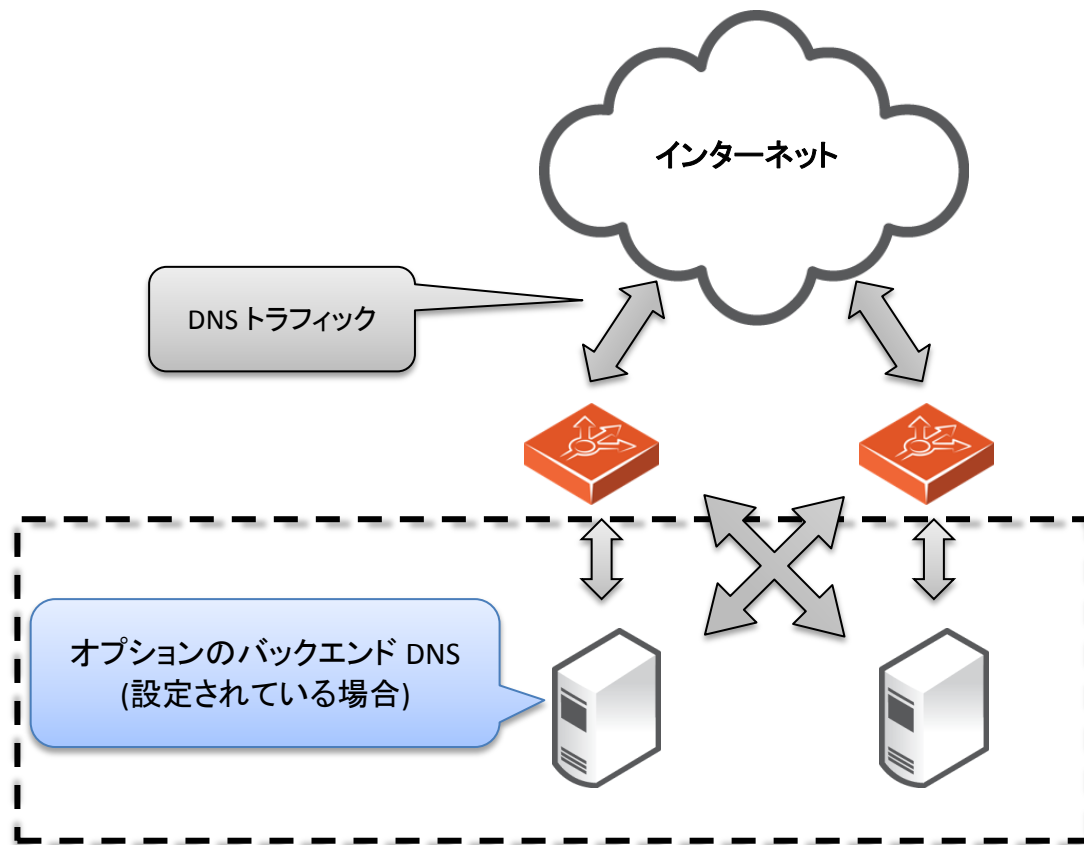


Fig 79. インラインデプロイの概略図

トラフィックマネージャをインラインの手法でデプロイするには、グローバルに分散させたい各ホスト名に対して複数の「A」レコードを返すように、内蔵の DNS サーバまたは外部 DNS サーバを設定します。内蔵の DNS サーバの設定方法については、CHAPTER 29 の「トラフィックマネージャの DNS サーバ」を参照してください。

次に、トラフィックマネージャ上の DNS 仮想サーバを作成します。範囲を最大にするには、DNS (TCP) および DNS (UDP) の両方の仮想サーバを作成することをお奨めします。必要に応じて、内蔵の DNS サーバプールまたはご使用の外部 DNS サーバのいずれかにデフォルトプールを設定します。外部 DNS プールを使用する場合、トラフィックマネージャは全ての DNS リクエストをこれらのネームサーバにプロキシします。

グローバルに分散させたいホスト名に対する DNS 仮想サーバに GLB サービスを追加し、親ドメインの委任レコードを更新して、トラフィックマネージャにそのドメインに対する権威を持たせます。

親ドメインには、多数のレコードタイプが保持されています。

- **NS (委任)レコード:** ドメインに権威を持つ DNS サーバのホスト名です。通常、複数の NS レコードを用意します。

- **A および AAAA (グルー)レコード:** 全ての NS レコードに対して、A レコードまたは AAAA レコードのいずれかを用意する必要があります。特に、NS レコードのホスト名がご使用のドメインに保持されている場合、親ゾーンには、そのホスト名の IP アドレスを提供するグルー A または AAAA レコードが保持されている必要があります。

親ドメインを更新するには、その親ドメインを管理する組織を最初に特定します。組織とは、外部の DNS レジストラ、ご自身の組織の別の部署、ご自身を管理するドメインなどです。

DNS レジストラによる委任レコードの更新

1. 通常、ご利用の DNS レジストラがコントロールする DNS 設定は、Web ベースのインターフェースから設定することができます。設定インターフェースの詳細については、ご利用の DNS レジストラにお問い合わせください。
2. ドメイン設定ページにログインし、ネームサーバのリストを見つけます。
3. 各レコードは、完全修飾ドメイン名(FQDN)と、それに対応する IP アドレス(オプション)で構成されています。これらのレコードを更新するには、現在の DNS サーバの FQDN と IP アドレスをトラフィックマネージャのものに置き換えます。

この手順の詳細については、ご利用の DNS プロバイダまたは DNS レジストラにお問い合わせください。

親ドメインの委任レコードが更新されると、トラフィックマネージャは、負荷分散を行うホスト名の全ての DNS 検索を受信ようになります。リクエストを受信すると、トラフィックマネージャは適切な DNS バックエンドにそのリクエストを転送し、IP アドレスのリストをレスポンスとして受信します。トラフィックマネージャはこのリストを処理して、エンドユーザに返す IP アドレスを正確に選択します。

並列デプロイ

並列デプロイでは、DNS エントリの CNAME を、別のドメイン(ほとんどの場合、ご使用のドメインのサブドメイン)のホスト名を指すよう設定します。次に、並列のホスト名に対する GLB を、インラインデプロイの手法に従って設定します。

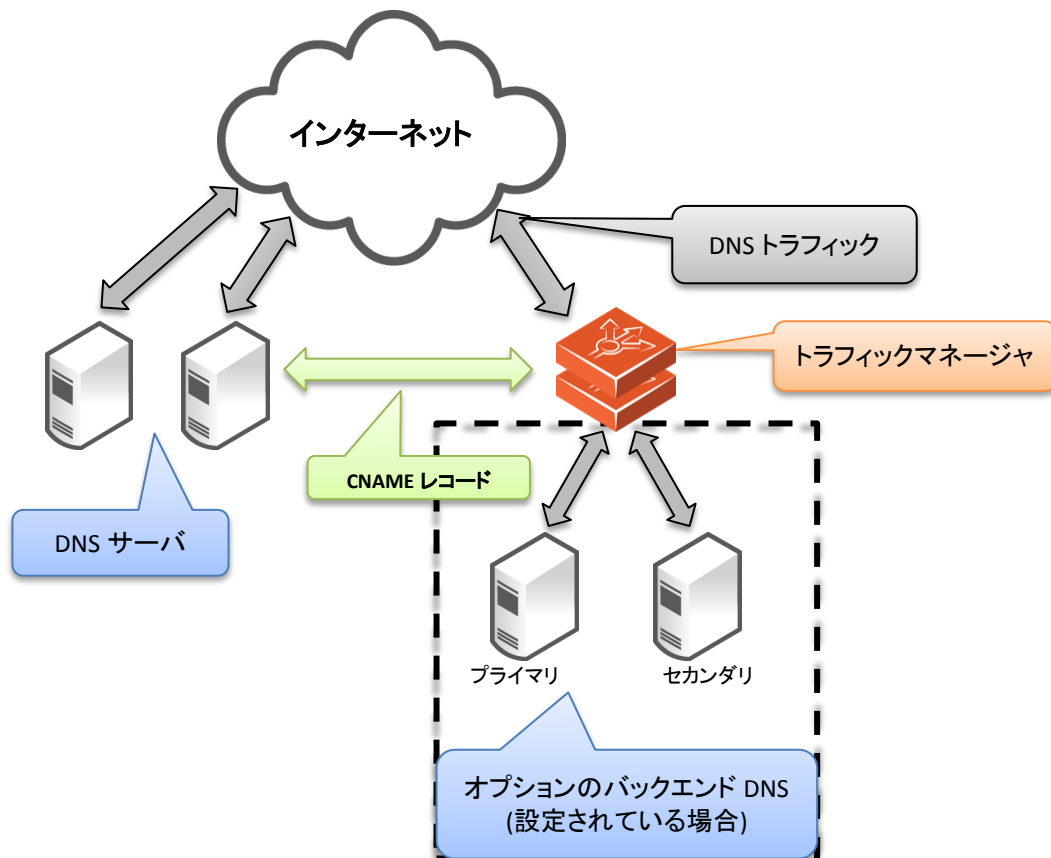


Fig 80. 並列デプロイの概略図

例えば、「example.com」のドメインを管理し、代替ホスト名の「www.gslb.example.com」を備える並列デプロイを使用して、「www.example.com」で GLB を有効化するとします。

この場合、「example.com」のネームサーバを、以下の 2 つのルールを満たすように設定します。

- 「www.example.com」から「www.gslb.example.com」へのリダイレクトを行う
- トラフィックマネージャに対して「gslb.example.com」の権威を委任する

次に、内部のトラフィックマネージャ DNS サーバ機能を使用するか、またはバックエンド DNS サーバの別のプールを介して、トラフィックマネージャで

「gslb.example.com」ドメインをホストします。「example.com」のゾーンファイルでは、以下のエントリを追加します。

```
; Delegation for gslb.example.com using the Traffic Manager.
gslb.example.com.      IN NS   stm1.gslb.example.com.
                       IN NS   stm2.gslb.example.com.

; Glue records for gslb.example.com
```

```
stm1.gslb.example.com.    IN A    192.0.2.11
stm2.gslb.example.com.    IN A    192.0.2.12

; CNAME for www.example.com to parallel GLB-enabled name.
www.example.com.          CNAME    www.gslb.example.com.
```

DNS クライアントは、ドメイン名 `www.example.com` を参照するときに、「example.com」ドメインのネームサーバと通信します。このネームサーバは、「代わりに `www.gslb.example.com` を試す」に相当する CNAME レスポンスを返します。ネームサーバは「example.com」ドメインに対して権威があるため、「gslb.example.com」に対する一部のネームサーバ(この場合はご使用のトラフィックマネージャ)の名前と IP アドレスに関する補足情報もネームサーバから得られます。

これにより、DNS クライアントは「`www.gslb.example.com`」に対するリクエストをトラフィックマネージャに送信し、トラフィックマネージャはそれを受けて全ての GLB ロケーションの IP アドレスが含まれるリストを取得します(内部 DNS サーバを使用するか、バックエンド DNS サーバにクエリします)。トラフィックマネージャはこのリストを処理し、いずれかの GLB ロケーションの IP アドレス(1 つまたは複数)を保持する 1 つ以上の A レコードを返します。

注: 並列デプロイにおいて、DNS サーバがトラフィックマネージャではない場合、並列のホスト名(`www.gslb.example.com`)に対する DNS サーバは、元のホスト名(`www.example.com`)に対する変更されていない A レコードを返してはなりません。そのため、トップレベルの DNS サーバはバックエンド DNS サーバとは異なるネームサーバにすることを強くお勧めします。

さらに、複雑になりますが、スプリットホライズン DNS を使用することもできます。ただし、この手法を用いる場合は、変更されていないレスポンスが外部に漏れないように注意が必要です。

バックエンド DNS サーバが全てトラフィックマネージャの場合、このような問題は起こりません。代わりに、グローバルに負荷分散されたホスト名をパブリックに提供する全ての DNS 仮想サーバで、GLB サービスを確実に設定する必要があります。

並列デプロイでは、DNS サブドメインを新たに作成および設定する必要があるため、デプロイがより複雑になる可能性があります。ただし、このデプロイはより柔軟性に優れており、DNS 設定を頻繁に変更することが予測される場合や、多数のドメインに同じサブドメインを使用して設定を一元化する場合に最適です。

例えば、多数のドメイン名(`www.example.com`、`www.mysite.com`、および `www.example.org`)のエイリアスを同一の CNAME (`www.gslb.example.com`)に設定することにより、ご使用のトラフィックマネージャを使用してホストできるようになります。そうすることで、トラフィックマネージャのバックエンド DNS サービス

のデプロイを適切に設定することにより、多数のドメインを単一の CNAME から一元的に設定することができます。

存続時間(TTL)フィールド

DNS レコードを変更する場合、その変更が即座にインターネットトラフィックで有効にならない可能性があります。これは、各 DNS レコードに存続時間(TTL)フィールドが含まれているからです。一般に、TTL フィールドは、ダウンストリームの DNS サーバに対してこのレコードを指定秒数だけキャッシングするよう指示するものです。その秒数の経過後、DNS サーバは新しいリクエストを送信しなければなりません。レコードがタイムリにインターネット全体に伝搬されるようにするためには、TTL 値を 30～60 秒に設定することをお奨めします。

トラフィックマネージャ GLB デプロイのコンポーネント

このセクションでは、トラフィックマネージャ GLB のアーキテクチャの概要とともに、有効なグローバル負荷分散の基本コンセプトについて取り上げます。

ご使用の GLB デプロイを設定する前に、このセクションをお読みいただくことをお奨めします。GLB の設定については、452 ページの「GLB の設定」を参照してください。

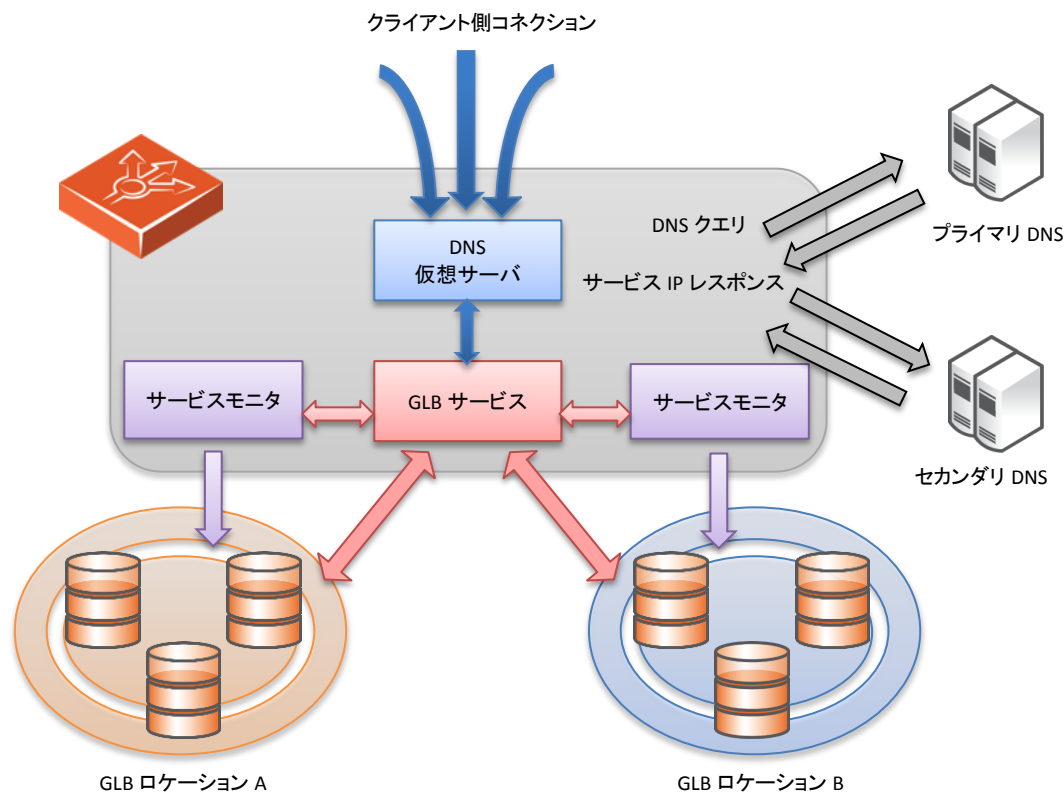


Fig 81. トラフィックマネージャ GLB デプロイのコンポーネント

GLB ロケーション

トラフィックマネージャは、GLB ロケーションを地理的に独立したサービス設備(データセンタと呼ばれることもある)と見なしています。世界の複数の国に2つ以上のGLB ロケーションをデプロイすることにより、各地の必要条件に適したアプリケーション配信インフラストラクチャを提供したり、重要システムの完全な災害復旧を実現したりできます。トラフィックマネージャがGLB ロケーションに物理的に存在する必要はありませんが、物理的に同じロケーションにある場合、性能および可用性上のメリットが得られます。

トラフィックマネージャにおけるGLB ロケーションは、世界の地理的位置によって定義されます。GLB ロケーションは、DNS レスポンスでクライアントのトラフィックをどこに送信するかを判断するために、GLB サービスによって使用されます。各GLB ロケーションは、独自のローカルDNS サーバ(トラフィックマネージャはこのサーバ宛てのDNS リクエストをプロキシする)とヘルスモニタを持っています。

注: GLB ロケーションは、419 ページの「ロケーション設定」セクションで説明しているロケーション設定とは別のコンセプトです。GLB を実装している場合、トラフィックマネージャは、それ自体がロケーション設定に存在するものとして指定されますが、トラフィックマネージャによって管理されるGLB サービスとは関係ありません。

複数のロケーションにあるトラフィックマネージャが同じDNS サーバを使用することもできます。ただし、性能上および信頼性上の理由から、ローカルにあるDNS サーバを使用するようトラフィックマネージャを設定することをお奨めします。これを行うには、GLB ロケーションと、CHAPTER 28 の「マルチサイトクラスタの管理」で説明したロケーション設定機能を組み合わせます。

GLB サービス

GLB サービスは、一連のDNS ドメインに使用するグローバル負荷分散設定を表します。GLB サービスでは、主に以下の情報を設定します。

- GLB サービスの適用先ドメイン
- これらのドメインのサービスをホストしているGLB ロケーション
- そのGLB ロケーションで使用する負荷分散アルゴリズム

さらに、以下の設定を行うことができます。

- DNS クライアントがそのドメインのDNS 情報を確認する頻度を定義するTTL (存続時間)値
- ドレイン: GLB ロケーションをドレインすると、クライアントからそのロケーションにトラフィックが送信されなくなります。

- **モニタ:** 1 つ以上のサービスモニタを設定することにより、各 GLB ロケーションのサービスおよびホストが正常に動作しているかどうかを確認できます。
- **ロギング:** チェックおよびデバッグのために、全ての DNS リクエストおよびレスポンスを記録することができます。

GLB を設定した仮想サーバとプール

各トラフィックマネージャは、GLB サービス用に特別に設定された仮想サーバを通して、受信する DNS リクエストを監視します。この仮想サーバは、DNS (UDP または TCP) を使用して、バックエンド DNS サーバノードで構成されるプールへのトラフィックを負荷分散します。

仮想サーバに割り当てられた複数の GLB サービスがある場合、トラフィックマネージャは、リクエストされたドメインに基づいて適切な GLB サービスを選択します。

DNS サーバ

DNS サーバをノードとして含むプールを作成すると、トラフィックマネージャは、そのプール設定に定義されている負荷分散アルゴリズムに従ってプールを負荷分散します。特定の DNS サーバノードがタイムアウトになると、アラートが報告され、リクエストを処理しているトラフィックマネージャは次の DNS サーバを試します。

性能上の理由から、トラフィックマネージャを DNS サーバと同じ物理的ロケーションにインストールしようとするかもしれませんが、こうすることで必ずしも性能が向上するとは限りません。

サービス IP アドレス

トラフィックマネージャがクライアントに送信するレスポンスを操作するためには、どの IP アドレスがどの GLB ロケーションに属しているかを認識していなければなりません。サービス IP アドレスは、管理対象のドメインについてのクエリに対して DNS サーバが一般にラウンドロビンのメカニズムで返す IP アドレスに一致します。

サービスモニタ

各 GLB サービスに、各 GLB ロケーションで実行するサービスモニタのリストを設定することができます。GLB ロケーションに定義された全てのサービスモニタが成功しないと、トラフィックマネージャによる GLB サービスの負荷分散の決定でその GLB ロケーションが考慮されません。

いずれかのモニタが(そのモニタに定義されている失敗しきい値制限内に)失敗した場合、その GLB ロケーション全体に**障害発生**のマークが付けられ、その失敗を検出したトラフィックマネージャは、クライアントの負荷分散時にそのロケーションを除外するようになります。

注: 環境によっては、地理的に分散された GLB デプロイ内の 1 つ以上のトラフィックマネージャが、隔離されたネットワークの障害などによって、ピアと同じ失敗を検出しないことがあります。そのようなトラフィックマネージャに対してサービスモニタから成功したレスポンスが返されると、トラフィックマネージャは、ピアが利用不可と検出した GLB ロケーションを使用し続けることがあります。

一般に、サービスモニタは、指定された特定のサービス IP アドレスまたはホスト名が正常に動作しているかどうかをテストします。例えば、適切にコンテンツを提供しているかどうかを検証するために、サービスモニタを、その GLB ロケーションにあるプライマリサーバの負荷分散をテストするように設定することがあります。プライマリサーバの負荷分散には、DNS によってその GLB ロケーションにあるそのサービスに対して発行された IP アドレスが割り当てられます。

複数のサービスまたはホストが正常に動作しているかどうかをテストする場合は、1 つの GLB ロケーションに複数のサービスモニタを割り当てることができます。

複数の GLB サービスに複数の GLB ロケーションを設定した場合、それぞれのサービスモニタは独立して動作します。ある GLB ロケーションで 1 つの GLB サービスのモニタが失敗した場合でも、他の GLB サービスのモニタが成功すれば、これらの GLB サービスでは引き続きこの GLB ロケーションが使用されます。

「Load」または「Adaptive」負荷分散手法を使用している場合、サービスモニタはオプションで性能情報を負荷分散アルゴリズムに提供することができます。これにより、負荷分散アルゴリズムでは、最高の性能が報告された GLB ロケーションに多くのトラフィックが送信されます。

性能データは、サービスモニタが完了までに要した時間、すなわちモニタされている GLB ロケーションのレスポンス時間に基づきます。

GLB の設定

トラフィックマネージャは、449 ページの「トラフィックマネージャ GLB デプロイのコンポーネント」で説明されているコンポーネントを使用して、グローバル負荷分散を実行します。このセクションでは、一般的なトラフィックマネージャのデプロイでそれらの各コンポーネントを設定する方法について説明します。

概要

トラフィックマネージャが、地理的に離れたロケーション間で負荷分散を行うには、以下の条件を満たす必要があります。

- GLB ロケーションが提供されるように定義する
- サービスモニタを作成し、各 GLB ロケーションの可用性をモニタリングする

- GLB サービスを作成し、リクエストされるドメインに対する DNS クエリを管理する
- バックエンド DNS サーバのプールを作成する
- DNS 仮想サーバを作成し、DNS リクエストを監視する

GLB ロケーションの定義

GLB ロケーションを表示、変更および新規追加するには、**Catalogs** (カタログ) > **Locations** (ロケーション) ページを使用します。既存の GLB ロケーションの名前または地理的位置を変更するには、ロケーション名をクリックするか、関連付けられた **Edit** (編集) ボタンをクリックします。

注: このページには、ロケーション設定のリストと GLB ロケーションのリストが別々に表示され、各リストで別々にロケーションの追加および削除を行えます。ロケーション設定の詳細については、CHAPTER 28 の「マルチサイトクラスタの管理」を参照してください。

新しい GLB ロケーションの作成

1. **Catalogs** (カタログ) > **Locations** (ロケーション) に移動します。
2. 「Create new GLB Location (新しい GLB ロケーションの作成)」セクションで、**Name** (名前) テキストボックスに識別名を入力します。
3. **Type** (タイプ) では「GLB」を選択します。

注: マルチサイトクラスタの管理が無効化されている場合、このページで作成した全てのロケーションのタイプは GLB となるため、このフィールドは表示されません。

4. **Add Location** (ロケーションの追加) をクリックすると、この GLB ロケーションが追加され、ロケーション編集ページにアクセスします。
5. ロケーション編集ページでは、GLB ロケーションの **Position** (位置) を選択します。国名のドロップダウンリストから選択するか、特定の緯度と経度の座標を選択するか、表示されるマップ上のポイントを選択します。
6. **Update Location** (ロケーションの更新) をクリックして変更を保存します。

GLB ロケーションを削除するには、まずそのロケーションを使用している GLB サービスとの関連付けを解除します。次に、**Confirm** (確認) チェックボックスをクリックし、**Delete Location** (ロケーションの削除) をクリックしてシステムからロケーションを削除します。

サービスモニタの作成

GLBロケーションにサービスモニタを設定するには、新しい**プール/GLB**モニタを作成し、各GLBロケーションでホストされているサービスのIPアドレスまたはホスト名をモニタリングするように指示します。選択するモニタタイプによっては、追加のパラメータが必要なことがあります。詳しくは、CHAPTER 14 の「ヘルスマニタリング」を参照してください。

サービスモニタの作成

1. **Catalogs** (カタログ) > **Monitors** (モニタ)に移動します。
2. 「Create new monitor (新しいモニタの作成)」セクションで、以下を入力します。
 - **Name (名前):** 作成するサービスモニタの識別名です。
 - **Type (タイプ):** 必要条件に最も適したモニタタイプを選択してください。
 - **Scope (スコープ):** **Pool/GLB** (プール/GLB)をクリックし、モニタリングしたいGLBロケーションに存在するサービスのIPアドレスまたはホスト名を入力します。モニタタイプによっては、ポート番号の入力も必要です。
3. **Create Monitor** (モニタの作成)をクリックすると、このサービスモニタが作成され、編集ページにアクセスします。
4. オプションで必要なパラメータを調整し、**Update** (更新)をクリックして変更を保存します。
5. この手順を、モニタリングしたい各GLBロケーションに対して行います。

GLB サービスの作成

GLB サービスは、FQDN と IP アドレスのセットに基づきます。新しい GLB サービスを作成するときは、各ドメイン名についてそれぞれの GLB ロケーションでホストされる IP アドレスをトラフィックマネージャに指定します。これにより、接続されている全てのトラフィックマネージャが各 GLB サービスの DNS 検索を管理するようになります。

GLB サービスの作成

1. **Catalogs** (カタログ) > **GLB Services** (GLB サービス)に移動します。
2. 「Create a new GLB service (新しい GLB サービスの作成)」セクションで、以下を入力します。
 - **Service Name (サービス名):** この GLB サービスの識別名です。

- **Domains (ドメイン):** この GLB サービスにグローバルに負荷分散させたいドメイン名を少なくとも 1 つ入力します。複数のドメインを入力するには、リストをカンマまたはスペースで区切ります。
- **Add locations (ロケーションの追加):** GLB サービスをホストさせたい、あらかじめ定義された GLB ロケーションを 1 つ以上クリックします。GLB ロケーションの作成については、453 ページの「GLB ロケーションの定義」を参照してください。

3. **Create GLB Service** (GLB サービスの作成)をクリックすると、この GLB サービスが作成され、編集ページにアクセスします。

GLB サービスの設定を変更するには、それぞれの GLB サービスの編集ページを使用します。このページには、サービスを作成した際に入力された GLB サービスの基本設定と、サブページに分類されたその他の利用可能な設定が表示されます。

新たに作成された GLB サービスは、それぞれ無効化された状態で追加されます。GLB サービスを有効化するには、**Enabled** (有効化)を「Yes (はい)」に設定します。

GLB サービスが負荷分散するドメインを追加または削除するには、**Domains** (ドメイン)テーブルを使用します。グローバル負荷分散を行う全ての FQDN のエントリはここで作成します。

ワイルドカード文字の指定には「*」を使用します。例えば、全ての「example.com」ドメインの全ての DNS 検索を管理する場合は、「*.example.com」と入力します。

Locations and Monitoring (ロケーションとモニタリング)

この GLB サービスに設定されている GLB ロケーションおよびサービスモニタを設定するには、**Catalogs** (カタログ) > **GLB Services** (GLB サービス) > **Locations and Monitoring** (ロケーションとモニタリング)をクリックします。

トラフィックマネージャには、設定されている各 GLB ロケーションに対する専用の設定サブセクションが用意されています。**Update** (更新)をクリックして変更を保存します。

通常、最初に GLB サービスを作成するときに、使用したい各 GLB ロケーションを選択します。追加の GLB ロケーションを追加するには、「Add Location (ロケーションの追加)」セクションを使用します。GLB ロケーションを削除するには、目的の GLB ロケーション設定サブセクションで、専用の **Remove this location** (このロケーションを削除)リンクを見つけてクリックします。

注: 新たに作成した GLB サービスの場合、選択した GLB ロケーションに関連する警告がトラフィックマネージャによって表示されることがあります。この警告は、各 GLB ロケーションに対して適切なサービス IP アドレスを提示するまで発し続けられます。

設定した各 GLB ロケーションについて、以下の設定を行うことができます。

- **Draining:** この GLB ロケーションへのトラフィックの送信を停止するには、これを「Yes (はい)」に設定します。例えば、装置をあるロケーションから別のロケーションへと物理的に移動する場合や、バックアップ電源システムをアップグレードするときに、UPS、ジェネレータ、インバータ、バッテリーを切断する必要がある場合、このオプションを使用して、トラフィックマネージャにこの GLB ロケーションへのトラフィックの送信停止を指示することができます。
- **Service IPs:** トラフィックマネージャがクライアントに送信するレスポンスを操作するためには、どの IP アドレスがどの GLB ロケーションに属しているかを認識していなければなりません。この設定オプションを使用すると、この GLB ロケーションに対する IPv4 アドレスまたは IPv4 CIDR サブネットのリストまたは範囲を入力することができます。これらの IP アドレスは、管理対象のドメインについてのクエリに対して DNS サーバがラウンドロビンのメカニズムで返す IP アドレスに一致します。

注: 一般に、これらの IP アドレスは、この GLB ロケーションに割り当てるサービスモニタで設定する IP アドレスと同じです。

- **Monitors:** この GLB ロケーションにトラフィックマネージャが使用すべきサービスモニタを選択します。各モニタでは 1 つのテストが実行され、各 GLB ロケーションに複数のモニタを設定することができます。選択できるのは、ホスト全体またはプール全体のスコープを持ったモニタのみです。モニタテストは、サービスモニタで設定されている特定のターゲットに対して、設定された間隔で実行されます。モニタが連続失敗回数の限度を超えた場合、特定のデータセンタのサービスに利用不可のフラグが設定されます。その結果、この GLB ロケーションは、以降の DNS レスポンスから除外されます。

負荷分散

この GLB サービスに対して負荷分散および障害復旧を設定するには、**Catalogs** (カタログ) > **GLB Services** (GLB サービス) > **Load Balancing** (負荷分散) をクリックします。

利用できる負荷分散手法には以下のものがあります。

- **Load:** 各 GLB ロケーションで検出された負荷に基づいてトラフィックを分配します。GLB ロケーションの現在の負荷を監視するために、各 GLB ロケーションで定義したヘルスモニタが使用されます。トラフィックマネージャの SOAP API を通じて各 GLB ロケーションに特定の負荷値をプログラムで設定することにより、このメカニズムをオーバーライドし、負荷分散の決定に影響を及ぼすことができます。各ロケーションに対して負荷値を宣言するには、`GLB.Service.setLoad()` 関数を使用します。トラフィックマネージャは、トラフィックの分配を決定する場合に、代わりにこの値を使用します。SOAP API を無効化し、GLB ロケーションのヘルスモニタに負荷レポートを戻すには、負荷値を 0 に設定します。

- **Geographic:** 各クライアントの地理的位置にのみ基づいてトラフィックを分配します。
- **Adaptive:** 現在の GLB ロケーションの負荷とクライアントの地理的位置の両方に基づいてトラフィックを分配します。特定のロケーションがトラフィックのほとんどを受信している(ほぼ過負荷状態になっている)ことが判明した場合、この負荷分散アルゴリズムを使用することで問題が解消されることがあります。この場合、このアルゴリズムによって、距離による待ち時間と負荷による待ち時間のバランスを最適な状態で取ることができます。
 - **Geo Effect:** このオプションを使用すると、トラフィックマネージャがクライアントのトラフィックの送信先 GLB ロケーションを決定するときの、地理的位置による影響の程度を微調整することができます。
- **Round Robin:** 利用可能な各 GLB ロケーションの現在のシーケンシャルな選択に基づいてトラフィックを分配します。時間が経つにつれ、全ての GLB ロケーションが同数のリクエストを受信します。
- **Weighted Random:** ランダムセレクションジェネレータを使用して、**Location Weights** の設定による加重に従ってトラフィックを分配します。トラフィックマネージャは、より大きな加重が設定された GLB ロケーションにより多くのトラフィックを送信します。
 - **Location Weights:** 各 GLB ロケーションに対して必要な比例加重を設定します。時間が経つにつれ、GLB ロケーションは、この設定を使用して宣言された比率に等しい数のリクエストを受信します。
- **Primary/Backup:** トラフィックマネージャは、プライマリ GLB ロケーションで障害が発生した場合にはバックアップ GLB ロケーションのみを頼りにして、常に 1 つのプライマリ GLB ロケーションに対する DNS レコードを送信します。状態情報に依存するサービス(買い物かご Web サイトなど)を提供し、データベースをリアルタイムにグローバルに複製していない場合、このアルゴリズムが適している場合があります。

GLB ロケーションのリスト内の優先順を決定するには、各 GLB ロケーションのインジケターバーの左にあるドラッグアンドドロップタブを使用します。(トップダウンの優先順で)GLB ロケーションを昇格させるには、リスト内でそのロケーションを目的の位置までドラッグします。

- **Automatic Failback:** 優先順の高い GLB ロケーションが障害から復旧したときに、どのような処理を行うかを定義します。「Yes (はい)」をクリックすると、優先順に高い GLB ロケーションが自動的にアクティブとしてマークされ、トラフィックマネージャは全てのユーザをそのロケーションに送信するようになります。「No (いいえ)」をクリックすると、復旧した GLB ロケーションは非アクティブのままになります。この場合、トラフィックマネージャはユーザをその時点でアクティブな GLB ロケーションに送信し続けます。

- **Activate best available location:** 「Automatic Failback」が「No (いいえ)」に設定されている場合にこのボタンをクリックすると、トラフィックマネージャは、その時点で有効化されているか利用可能な GLB ロケーションのリストから新しいアクティブな GLB ロケーションを選択します。

全ての負荷分散アルゴリズムに障害復旧オプションを設定するには、以下の設定を使用します。

- **disable_on_failure:** GLB ロケーションの復旧が自動的に有効化されないようにするには、「Yes (はい)」をクリックします。その後、特定の無効化された GLB ロケーションを手動で有効化するには、目的の GLB ロケーションインジケータバーの **Enable** (有効化) をクリックします。この設定は、復旧した GLB ロケーションを使用する前に状態またはコンテンツを同期する必要があるデプロイに適しています。
- **autorecovery:** 全ての GLB ロケーションで障害が発生した場合に「Yes (はい)」をクリックすると、トラフィックマネージャは、最後に障害が発生した GLB ロケーションを、そのロケーションが復旧したときに自動的に有効化します。
「Primary/Backup」アルゴリズムを使用している場合、復旧した GLB ロケーションもアクティブになります。「No (いいえ)」をクリックすると、復旧した全てのロケーションは無効化されたままになります。

注: 「autorecovery」の設定は、有効化されている場合、「Automatic Failback」および「disable_on_failure」に優先します。

- **last_resort_response:** この GLB サービスに接続されている全ての GLB ロケーションが利用不可になった場合、トラフィックマネージャは、この設定の内容を最終手段の DNS レスポンスとして全てのクライアントリクエストに返します。トラフィックマネージャが DNS サーバから受信した全ての IP アドレスを返すようにするには、スペースで区切られた IP アドレスのリスト、単一のドメイン名、または「ALL」のいずれかを使用します。

DNS Authentication (DNS 認証) (DNSSEC)

この GLB サービスに設定されているドメインネームシステムセキュリティ拡張 (DNSSEC) を設定するには、**Catalogs (カタログ) > GLB Services (GLB サービス) > DNS Authentication (DNSSEC) (DNS 認証(DNSSEC))** をクリックします。

DNSSEC 仕様は、DNS にセキュリティ強化および認証のための一連の拡張機能を付加したものです。トラフィックマネージャが DNSSEC 認証されたレスポンスを変更できるようにするには、このページで、署名ドメインと DNSSEC 秘密鍵の関連付けを設定します。

関連付けの追加

1. 表示されるボックスに署名ドメインを入力し、プラス記号をクリックします。
2. ドロップダウンボックスからドメインに関連付けられた秘密鍵を選択します。

3. **Update** (更新)をクリックして関連付けを確定します。

複数の鍵を 1 つのドメインに関連付けたり、個々の鍵を使用して多数のドメインを認証(多対多の関係)したりすることができます。

DNSSEC 秘密鍵を追加するには、**Manage DNSSEC private keys** (DNSSEC 秘密鍵の管理)をクリックして **Catalogs** (カタログ) > **SSL** > **DNSSEC Keys** (DNSSEC 鍵) ページにアクセスします。SSL 証明書および鍵のインポートおよびアップロードの詳細については、CHAPTER 13 の「SSL 暗号化」を参照してください。

DNSSEC の詳細については、www.dnssec.net を参照してください。

Rules (ルール)

この GLB サービスに対して TrafficScript ルールを設定するには、**Catalogs** (カタログ) > **GLB Services** (GLB サービス) > **Rules** (ルール)をクリックします。

この GLB サービスに TrafficScript ルールを適用することで、追加的なトラフィック管理機能を利用できるようになります。ただし、DNS プロトコル(http.*など)と互換性のない関数は使用できません。TrafficScript とその機能の詳細については、CHAPTER 8 の「TrafficScript ルール」を参照してください。

Request Logging (リクエストロギング)

この GLB サービスに対してリクエストロギングを設定するには、**Catalogs** (カタログ) > **GLB Services** (GLB サービス) > **Request Logging** (リクエストロギング)をクリックします。

環境によっては、トラフィックマネージャクラスタに送信されてくる DNS リクエストをさらに正確に把握することが必要になる場合があります。リクエストロギングでは、各リクエストを指定のファイルに記録し、後で解析に利用することができます。

以下を設定することができます。

- **log!enabled:** リクエストロギングを有効化するには、「Yes (はい)」に設定します。
- **log!filename:** トラフィックマネージャがこの GLB サービスに対するリクエストログを保存するローカルファイルシステムのフルパスおよびファイル名です。**Macros...**(マクロ)をクリックすると、利用可能なマクロのリストを確認できます。
- **log!format:** リクエストの文字列フォーマットです。各リクエストに関する情報には、「%」でエスケープしたマクロによってアクセスできます。**Macros...**(マクロ)をクリックすると、利用可能なマクロのリストを確認できます。

ご使用の GLB サービスに対するリクエストログを表示するには、**View Request Logs in file** (リクエストログをファイルで表示)をクリックします。

DNS 設定

この GLB サービスに対する DNS 設定を行うには、**Catalogs (カタログ) > GLB Services (GLB サービス) > DNS Settings (DNS 設定)** をクリックします。

以下を設定することができます。

- **Time To Live (TTL):** DNS サーバにより、DNS レスポンスの各リソースレコードの TTL 値が設定されます。TTL 値は、キャッシングリゾルバが DNS レコードを再度名前解決するまでキャッシングしておく時間をコントロールします。GLB ロケーションに障害が発生し、DNS サーバによって設定された TTL 値が長すぎる場合(数時間または数日など)、一般に、GLB サービスでは迅速なフェールオーバーが必要です。トラフィックマネージャによって処理されるリソースレコードに対して TTL をオーバーライドするには、**Custom (カスタム)** をクリックし、新しい値(秒単位)を入力します。

DNS サーバプールの作成

GLB 仮想サーバがグローバル負荷分散を行うためには、DNS サーバノードが含まれるプールを作成する必要があります。

DNS サーバプールの設定

1. **Services (サービス) > Pools (プール)** に移動します。
2. 「Create a new Pool (新しいプールの作成)」セクションで、以下を追加します。
 - **Pool Name (プール名):** 新しいプールの識別名です。
 - **Nodes (ノード):** 1 つ以上の DNS サーバの IP アドレス、または<host>:<port> のフォーマットで記されたホスト名と対応するポート番号のリストです。複数のノードを指定するには、スペースまたはカンマ区切りを使用します。
 - **Use autoscaling (オートスケーリングの使用):** このボックスのチェックを外します。
 - **Monitor (モニタ):** ご使用の DNS サーバのヘルスをチェックするモニタをオプションで指定します。
3. **Create Pool (プールの作成)** をクリックすると、このプールが作成され、編集ページにアクセスします。

DNS 仮想サーバの作成

仮想サーバは、受信する DNS リクエストを処理し、そのリクエストをバックエンド DNS サーバプールに渡します。作成した GLB サービスは、この仮想サーバに関連付けます。

DNS 仮想サーバの作成

1. **Services** (サービス) > **Virtual Servers** (仮想サーバ) に移動します。
2. 「Create a new Virtual Server (新しい仮想サーバの作成)」セクションで、以下を入力します。
 - **Virtual Server Name** (仮想サーバ名): 作成する仮想サーバの識別名です。
 - **Protocol** (プロトコル): ドロップダウンリストから「DNS (UDP)」または「DNS (TCP)」を選択します。
 - **Port** (ポート): この仮想サーバが監視する受信ポート番号を選択します。
 - **Default Traffic Pool** (デフォルトトラフィックプール): ご使用の DNS サーバノードが含まれるプールを選択します。詳細については、460 ページの「DNS サーバプールの作成」を参照してください。
3. **Create Virtual Server** (仮想サーバの作成) をクリックしてこの仮想サーバを作成し、編集ページにアクセスします。
4. **GLB Services** (GLB サービス) をクリックすると、「Edit GLB Services (GLB サービスの編集)」ページにアクセスします。
5. 「Add new GLB Service (新しい GLB サービスの追加)」セクションで、ドロップダウンボックスから目的の GLB サービスを選択し、**Add Service** (サービスを追加) をクリックします。GLB サービスの作成の詳細については、454 ページの「GLB サービスの作成」を参照してください。
6. 仮想サーバを有効化するには、**Virtual Server edit** (仮想サーバ編集) ページで **Enabled** (有効化) を「Yes (はい)」に設定するか、トラフィックマネージャのホームページの仮想サーバ名の隣にある *play*(作動) アイコンをクリックします。

EDNS0 Client Subnet のサポート

一般に、トラフィックマネージャでは、ロケーションに基づいて GLB の決定を下す場合に、受信 DNS リクエストの IP アドレスを使用して、認識されたクライアントの地理的ロケーションが決定されます。しかし、この IP アドレスは、送信元のクライアントではなく、アップストリームサーバまたは再帰的リゾルバに属することがあります。多くの場合、DNS リクエストの IP アドレスは、トポロジ的にリクエスト元に近いと見なすことが安全です。しかし、状況によっては、クライアントとアップストリームリゾルバ間の距離が大きく、トラフィックマネージャであまり効率的でない GLB の決定が下されることがあります。

この状況における問題を軽減するために、トラフィックマネージャは、EDNS0 Client Subnet プロトコル拡張機能の使用をサポートして、DNS リクエストで送信元クライアントのサブネット IP アドレスの識別を容易にします。この機能を有効化す

ると、GLB サービスでは、DNS リクエストの送信元アドレスではなく、この拡張機能で保存されたネットワークアドレスが使用されます。

EDNS0 Client Subnet のサポートを有効化するには、DNS 仮想サーバで **dns!edns_client_subnet** を Yes (はい) に設定します。

EDNS0 Client Subnet オプションの詳細については、<https://datatracker.ietf.org/doc/draft-ietf-dnsop-edns-client-subnet> にある IETF ドキュメントのドラフトを参照してください。これは作業中のドラフトであるため、変更される可能性があることに注意してください。

トラフィックの可視化

GLB が有効化されている場合、トラフィックマネージャでは、管理 UI の既存の可視化および診断ツールに追加情報が表示されます。例えば、ワールドマップには、受信した DNS リクエストと、その後クライアントが接続された GLB ロケーションがリアルタイムに表示されます(詳細については、426 ページの「ワールドマップ」を参照)。

現在のアクティビティグラフ

システム上のアクティビティのリアルタイムのグラフを確認するには、**Activity** (アクティビティ) > **Current Activity** (現在のアクティビティ) をクリックします。

「Settings (設定)」セクションで、データをトラフィックマネージャ別またはロケーション設定別にプロットできます。どちらの場合も、システムに設定されている対応するトラフィックマネージャまたはロケーションの全部または一部を選択したり、結果を統合してプロットするか別々にプロットするかを選択したりできます。

GLB が有効化されていると、トラフィックマネージャは、GLB サービスおよびロケーションに関するメトリクスの新しいセット 2 つを、**Current Activity** (現在のアクティビティ) > **Edit** (編集) ページに表示されるモニタ値ツリーに追加します。

現在のアクティビティグラフの詳細については、CHAPTER 7 の「トラフィックマネージャ管理インターフェースの主要機能」を参照してください。

ヒストリカルアクティビティグラフ

トラフィックマネージャは、過去 90 日間のシステムのアクティビティを記録します。このデータをグラフ形式で表示するには、**Activity** (アクティビティ) > **Historical Activity** (ヒストリカルアクティビティ) をクリックします。データは、線形軸または対数軸にプロットできます。

「Settings (設定)」セクションで、データをトラフィックマネージャ別またはロケーション設定別にプロットできます。どちらの場合も、システムに設定されている対

応するトラフィックマネージャまたはロケーションの全部または一部を選択したり、結果を統合してプロットするか別々にプロットするかを選択したりできます。タイムスケールの範囲は1時間から90日間まででプロットできます。

ヒストリカルアクティビティグラフの詳細については、CHAPTER 7の「トラフィックマネージャ管理インターフェースの主要機能」を参照してください。

Connections (コネクション) ページ

GLB クラスタで行われた最近の負荷分散の決定に関する情報を表示するには、**Activity** (アクティビティ) > **Connections** (コネクション) をクリックします。**Connections** (コネクション) ページの詳細については、CHAPTER 7の「トラフィックマネージャ管理インターフェースの主要機能」を参照してください。

GLB リクエストログ

各 GLB サービスが処理する個々のリクエストを表示するには、**Activity** (アクティビティ) > **View Logs** (ログの表示) をクリックし、確認したい GLB サービスの名前をクリックします。

トラフィックマネージャがログ情報の収集を開始するためには、まず GLB サービスに対してリクエストロギングを有効化する必要があります。詳細については、459 ページの「Request Logging (リクエストロギング)」を参照してください。

Dig による DNS のテスト

「Dig」は、DNS システムのテストおよびトラブルシューティングに推奨されるツールで、Berkley Internet Name Domain (BIND) ソフトウェアの管理も行っている Internet Systems Consortium, Inc. によって管理されています。Dig はオープンソースソフトウェアのため、多くのアーキテクチャおよび OS で利用できます。

Linux の場合、dig は一般に、実行するディストリビューションに応じて、BIND とともにインストールされているか追加パッケージとして提供されます。

詳細については、以下の Internet Systems Consortium 公式 Web サイトをご覧ください。

<http://www.isc.org/downloads/bind>

Dig の使用例は以下のとおりです。

```
$ dig -t any +norec www.example.com @192.0.2.1
```

上記の例では、

- 「-t any」は、全てのタイプのレコード(A、NS、CNAME、PTR など)をリクエストします。
- 「+norec」は、再帰的フラグを「off」に設定してリクエストを送信するよう指定します。
- 「www.example.com」は、テスト対象のレコードです。
- 「@192.0.2.1」は、リクエストの送信先ネームサーバです。

DNS サーバまたはサービスをテストするときに、必ず非再帰的クエリを実行することが非常に重要です。これにより、信頼できる回答を受け取ることができるか、または名前解決しようとしているドメイン名に権威を持つネームサーバを正確に知ることができます。クエリ先のネームサーバが検索した回答を受け取ることはありません(必ずしも情報源を検証できるわけではないため)。

トラフィックマネージャの GeoIP データベースの拡張

トラフィックマネージャには、IP アドレスをロケーション(経緯度、都市、郡および国)にマッピングする GeoIP データベースが含まれています。GLB は、この GeoIP データベースを使用して、リモートユーザとローカルのデータセンタ間の距離を推測します。

データベースには、含まれている geo.* TrafficScript および Java 関数を使用して直接アクセスできます。例えば、サイト訪問者のアクセス元の 2 文字の国コードを検出するには、以下の TrafficScript を使用します。

```
$ip = request.getRemoteIP();  
$countryCode = geo.getCountryCode( $ip );
```

トラフィックマネージャは、MaxMind 社の GeoLite City データベース(<http://dev.maxmind.com/geoip/legacy/geolite/>)から得られるデータベースを使用します。Brocade は、トラフィックマネージャをリリースするごとにこのデータベースを更新しますが、アップグレードパッケージを使用して手動で最新バージョンに更新することもできます。

詳細については、Brocade が提供するコミュニティ Web サイト(<http://community.brocade.com>)を参照してください。

未認識 IP アドレス

データベースにはプライベート IP アドレスの範囲に対するロケーションは含まれていません(RFC1918 を参照)。また、その他の IP アドレスの範囲も、最近割り当てられたか移動された場合は含まれていないか正確でないことがあります。このセクシ

ョンでは、内部の GeoIP データベースを拡張して、IP アドレスの範囲を追加またはオーバーライドする方法について説明します。

トラフィックマネージャの GeoIP データベースの拡張

データベースへの拡張は、\$ZEUSHOME/zxtm/conf/locations.cfg ファイルに保存されます。

拡張ごとに 1 行で指定される以下のフォーマットを使用します。

firstIP	lastIP	lat	lon	CC	RR	city

各マッピングには、以下のルールと定義が適用されます。

- 行内の要素は空白で区切ります。
- IP アドレスの範囲は包括的で、緯度(「lat」)および経度(「lon」)は「-」または 10 進法の度数のいずれかとします。
- CC および RR は「-」または 2 文字の国および地域コード(例: US TX はテキサス州)のいずれかとします。専用ファイルの
\$ZEUSHOME/zxtm/etc/geo/country_codes.txt および
\$ZEUSHOME/zxtm/etc/geo/region_codes.txt に該当コードの一覧を含めます。
- 「San Francisco」のように、都市名にスペースを含めることができますが、ある都市を具体的に言及する必要はありません(説明文であれば許容されます)。
- 冒頭の 2 つのフィールドのみ必要です。

以下にマッピングの例を挙げます。

192.168.0.1	192.168.0.128	52.1234	-0.5678	US	TX	New datacntr
172.16.0.1	172.16.255.255	-	-	-	-	Test VPN
99.98.97.96	99.98.97.99					

IP アドレスマッピングのテスト

以下の TrafficScript リクエストルールの例を使用して、全ての変更をテストすることができます。

<pre> \$text = ""; # Test your own IP addresses with the whereis function here. \$text .= whereis("192.168.35.40"); \$text .= "\n"; \$text .= whereis("192.168.199.199"); \$text .= "\n"; \$text .= whereis("17.18.19.20"); \$text .= "\n"; </pre>

```
http.sendResponse( "200", "text/plain", $text, "" );

sub whereis( $ip ) {
    return $ip . " is in:\n" .
        "    Country:      " . geo.getCountry( $ip ) . "\n" .
        "    CountryCode: " . geo.getCountryCode( $ip ) . "\n" .
        "    Region:       " . geo.getRegion( $ip ) . "\n" .
        "    City:         " . geo.getCity( $ip ) . "\n" .
        "    Long/Lat:     " . geo.getLongitude( $ip ) . '/' . geo.g
etLatitude( $ip ) . "\n";
}
```

トラフィックマネージャのクラスタ設定の更新

locations.cfg ファイルは直接編集することができ、それを受けてローカルのトラフィックマネージャ設定システムは、このファイルが変更されたことを通知し、ファイル内で定義されたロケーションマッピングを自動的に受け入れます。

しかし、ファイルシステム内の設定ファイルを直接編集することによって、ご使用のトラフィックマネージャの設定は他のクラスタメンバに自動的に複製されなくなります。更新された設定を手動で複製するには、更新を行ったトラフィックマネージャで以下のいずれかの手法を使用します。

- 管理 UI の **Diagnose** (診断) > **Cluster Diagnosis** (クラスタ診断) ページで複製オプションを使用する
- ZEUSHOME/zxtm/bin/replicate-config スクリプトをコマンドラインで実行する

CHAPTER 31 トラフィックマネージャの FIPS 認証

FIPS について

本章では、連邦情報処理標準(FIPS)に固有の情報を説明するとともに、トラフィックマネージャに FIPS を実装する方法について詳しく説明します。ご不明な点は、サポート窓口にお問い合わせください。

FIPS は、コンピュータシステムでの使用を目的として、アメリカ合衆国連邦政府によって開発されています。利用可能かつ必要なライセンスを所有している場合、トラフィックマネージャでは、FIPS に準拠した特別動作モードやサードパーティの暗号化動作モジュールを使用することができます。

重要: トラフィックマネージャは、FIPS による認証を受けていません。使用可能な場合に限り、トラフィックマネージャは全ての FIPS 動作において認証済みの暗号化モジュールを使用します。

FIPS モード

FIPS モードをアクティブにすることによって得られる最も重要な機能として、トラフィックを処理するときに **FIPS 140-2 準拠暗号化モジュール**を使用できることが挙げられます。一般に、トラフィックマネージャの FIPS モードは、組織が FIPS 準拠に向けて取り組むうえで役立つ機能です。

さらに、以下の 2 つの理由のため、FIPS モードを設定すると、トラフィックマネージャの様々な設定/動作が制限されます。

1. 暗号化モジュールへの入力が FIPS 140-2 の認証範囲内でなければなりません。
2. FIPS 140-2 の要件と互いに排他的な動作を有効化できてはなりません。制限される動作として、SSL 2.0 や SSL 3.0 が挙げられます(FIPS 140-2 に適合しているのは TLS 1.0 およびそれ以降のみです)。

FIPS 140-2

FIPS 140-2 は、アメリカ国立標準技術研究所(NIST)によって発行された、FIPS 140 発行シリーズの第 2 版です。詳細については、以下のサイトで確認できます。

<http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>

本ドキュメントでは、暗号化モジュールに対するセキュリティ要件について説明します。FIPS 140-2 の要件には、1~4 の範囲で 4 つのセキュリティレベルが定義されています。一般に、ソフトウェアのみのセキュリティモジュールは、最低レベル(1)を達成することしかできないと考えられています。それよりも高いセキュリティレ

ベルには、改ざん保護のような、物理的製品でのみ達成できる要件が設けられています。通常、これらの要件は、暗号化動作に関与するプロセスに関係しています。

さらに、その他にも以下のような NIST の「特別刊行物」も考慮する必要があります。

- 「Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations (トランスポート層セキュリティ(TLS)実装の選択、設定および使用に関するガイドライン)」、SP 800-52r1 (<http://dx.doi.org/10.6028/NIST.SP.800-52r1>)
- 「Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths (遷移: 暗号化アルゴリズムの使用遷移および鍵長に関する推奨事項)」、SP 800-131A (<http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>)
 - 「鍵強度」の構成要素に関する検討事項、SP 800-51 パート 1 (http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf)

注: トラフィックマネージャは、本ドキュメントに記載された全ての推奨事項を実装しているわけではありません。

FIPS 140-2 とトラフィックマネージャ

トラフィックマネージャの Linux 系製品で使用されている FIPS 140-2 準拠暗号化モジュールは、Riverbed®暗号化セキュリティモジュール(RCSM)(証明書番号 2099)です。このモジュールの FIPS 140-2 有効性について、詳しくは以下の NIST CMVP ページに記載されています。

<http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2014.htm#2099>

FIPS モードが有効化されているときにトラフィックマネージャを起動すると、RCSM が読み込まれて初期化されます。初期化(パワーオンセルフテストなど)の最中に発生した障害は、全て致命的なエラーとして報告され、「トラフィックマネージャ FIPS 境界」での全ての暗号化動作が無効化されます(SSL 復号化仮想サーバ、SSL 暗号化プールなど)。

トラフィックマネージャ FIPS 境界

トラフィックマネージャは、SSL/TLS をはじめ、数多くの機能で暗号化を使用しています。ただし、以下に示す専用トラフィックマネージャ FIPS 境界内でこれらの暗号化が全て行われているわけではありません。

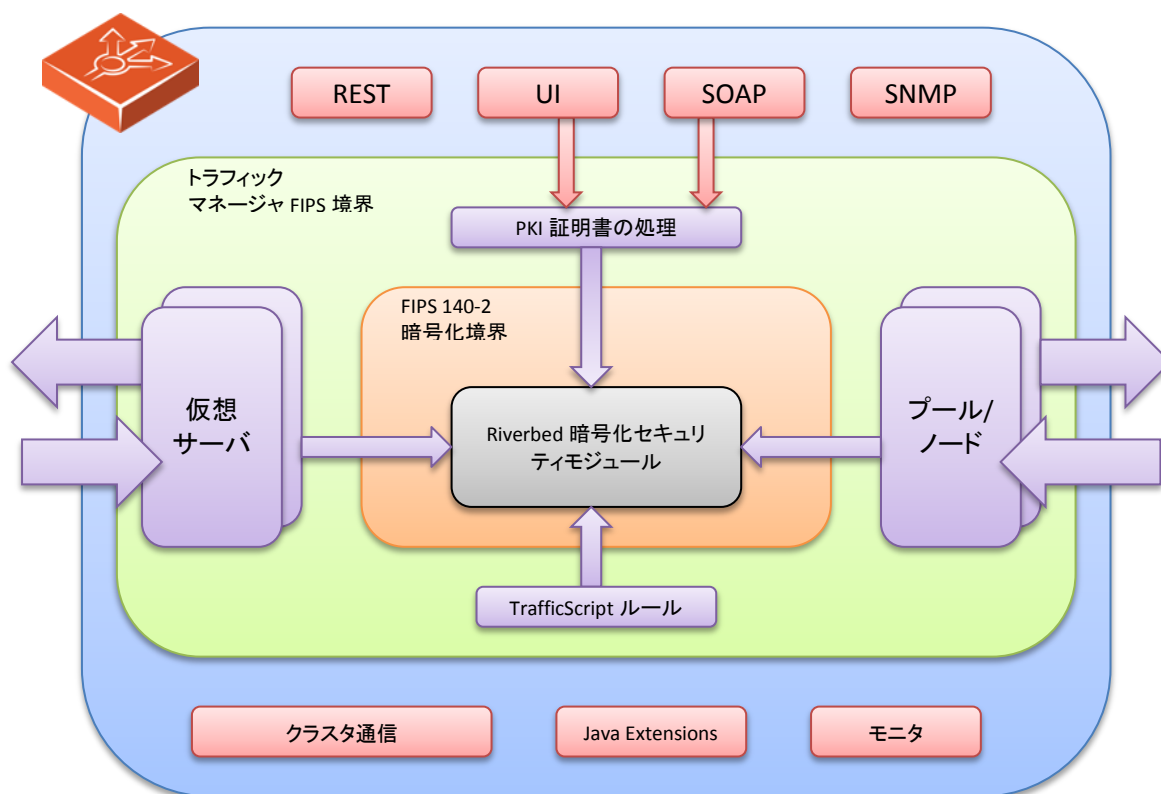


Fig 82. トラフィックマネージャ FIPS 境界

この境界は、以下のように極めて簡単に定義づけられています。

1. 全ての暗号化動作がトラフィックの処理のために行われる。
2. トラフィックを処理するとき、暗号化オブジェクトとの全ての暗号化相互作用が使用される。

FIPS モードが有効であると、以下のとおり動作します。

項目(1)は主として TLS を対象とします。仮想サーバおよびプールに関してトラフィックマネージャが処理する全てのデータは、全ての暗号化において FIPS 140-2 準拠の暗号化モジュールを使用します。これには、OCSP リクエストなど、TLS による下位動作も含まれます。

項目(2)は、保存されている暗号化オブジェクトの操作をトラフィックマネージャ設定の一部と見なします。PKI 証明書や秘密鍵などが主な例に挙げられます。これらのオブジェクトとの暗号化相互作用は、全て FIPS 140-2 準拠暗号化モジュールにより行われます。

トラフィックマネージャ FIPS 境界には、「コントロールプレーン」の暗号化は一切含まれません。具体的には、以下に挙げる機能はトラフィックマネージャ FIPS 境界に含まれません(ただし、完全に網羅しているわけではありません)。

- トラフィックマネージャ SNMP 相互作用

- トラフィックマネージャ SOAP API
- トラフィックマネージャ REST API
- トラフィックマネージャ管理 UI
- トラフィックマネージャアプライアンスへの SSH アクセス

境界において(2)が重要である理由は、例えば SOAP または管理 UI の場合、転送に使用される暗号化に FIPS 140-2 準拠暗号化モジュールが使用されていなくても、トラフィックマネージャ FIPS 境界内での使用のために暗号化オブジェクトを操作する可能性のある全ての要求された動作が、FIPS 140-2 準拠暗号化モジュールによって行われることにあります。

FIPS モードがご使用のデプロイ要件に適しているかどうかを検討するときには、トラフィックマネージャ FIPS 境界の機能的な対象範囲が十分であるか検討する必要があります。

例外としての TLS

前述したように、境界内で行われる全ての暗号化動作が FIPS 140-2 暗号化モジュールを使用して行われるというのがトラフィックマネージャ FIPS 境界の一般的定義です。

ただし、MD5 が TLS 1.0 ハンドシェイクプロトコルの要件の一部である場合、これには例外があります。MD5 を使用しない場合、最新バージョンの TLS のみを要求する TLS クライアントと相互運用できなくなります。このケースは、前述のドキュメント『**Guidelines for the Selection and Use of Transport Layer Security (TLS)**

Implementations (トランスポート層セキュリティ(TLS)実装の選択および使用に関するガイドライン)』に記された考察に該当します。それによると、TLS ハンドシェイクプロトコルのこのような極めて特定の部分における MD5 の特定用途は、安全であると見なされます。技術的な詳細については、このドキュメントを参照してください。

このため、通常 MD5 は FIPS 140-2 準拠暗号化モジュールから利用可能ではありませんが、このような特定のケースにおいてのみ、トラフィックマネージャの SSL 実装では、別の暗号化実装からの MD5 アルゴリズムが使用されます。

同様に、別の用途においてもトラフィックマネージャ FIPS 境界内で MD5 アルゴリズムが使用されますが、それらの用途は暗号化またはセキュリティに関する目標には無関係であると見なされています(例えば、暗号化ダイジェストではなくハッシュ関数として MD5 アルゴリズムが使用されます)。

FIPS モードのデプロイ

準備

FIPS モードの設定は、その他の特定の設定に関する前提条件が満たされている場合に限り行うことができます。その前提条件は、トラフィックを処理するために FIPS 140-2 準拠暗号化モジュールを使用するための適切な環境を整えるために満たす必要があります。

そのため、FIPS モードを有効化するための準備において、以下の設定要素をチェックし、必要に応じて更新してください。

サポートされている SSL/TLS バージョン

SP 800-52r1 のガイドラインでは、FIPS 140-2 準拠暗号化モジュールを使用する環境での動作に適しているのは、TLS 1.0 およびそれ以降の SSL バージョンのみとされています。

したがって、FIPS モードを有効化するためには、トラフィックマネージャ設定で SSLv2 および SSLv3 が無効化されていることを確認してください。これは、**System** (システム) > **Global Settings** (グローバル設定) > **SSL Configuration** (SSL 設定) ページで「ssl!support_ssl2」および「ssl!support_ssl3」を「No (いいえ)」に設定することにより無効化されます。さらに、仮想サーバおよびプールがグローバル SSL をオーバーライドしないことを個々のレベルで確認してください(**Virtual Server** (仮想サーバ) > **Edit** (編集) > **SSL Decryption** (SSL 復号化) および **Pool** (プール) > **Edit** (編集) > **SSL Encryption** (SSL 暗号化) ページの「ssl_support_ssl<version>」設定を使用します)。

重要: ご使用の SSL 仮想サーバクライアントおよびプールノードが TLS 1.0 またはそれ以降をサポートしていない場合、それらは SSL を使用して接続できなくなります。

SSL3 暗号スイートの選択

FIPS 140-2 の要件には、FIPS 140-2 準拠暗号化モジュールに含めることができる承認済みの暗号化手法のリストが提示されています。逆に、承認されていない手法は暗号化モジュールでは許可されていません。

SSL/TLS 暗号スイートの中には、FIPS 140-2 による承認を得ていない暗号化手法が含まれています。例えば、RC4 ストリーム暗号や MD5 ダイジェストアルゴリズムがこれに該当します。FIPS モードにおいて、全ての TLS 暗号化が FIPS 140-2 準拠暗号化モジュールを使用して行われるようにするために、トラフィックマネージャとの使用に向けて設定できる暗号スイートのリストが以下のセットに減らされています。

- SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

- SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_128_GCM_SHA256
- SSL_RSA_WITH_AES_128_CBC_SHA256
- SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- SSL_DHE_RSA_WITH_AES_128_GCM_SHA256
- SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA256
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_AES_256_GCM_SHA384
- SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- SSL_DHE_RSA_WITH_AES_256_GCM_SHA384
- SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- SSL_DHE_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA
- SSL_DHE_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA

重要: これは、SSL 仮想サーバクライアントおよびプールノードとの相互運用性に影響します。SSL 仮想サーバクライアントおよびプールノードが FIPS 140-2 準拠のいずれの暗号スイートもサポートしていない場合、それらは SSL を使用して接続できなくなります。

TrafficScript 関数

TrafficScript を使用することで、トラフィックマネージャで実行されるルール多くの暗号化動作にアクセスできます。FIPS モードで動作しているときは、これらの動作は FIPS 140-2 準拠暗号化モジュールにより行われます。

FIPS 140-2 準拠暗号化モジュール用に承認されていない暗号化手法を使用する TrafficScript 関数の場合、FIPS モードでの動作では、それらの関数は適切に実行されません。以下に挙げる TrafficScript 関数を使用している場合は、FIPS 140-2 準拠暗号化モジュール向けに承認されている別の暗号化ダイジェスト実装に移行することをお奨めします。

FIPS モードで使用するとエラーを発生させる TrafficScript 関数は、MD5 アルゴリズムを使用する TrafficScript 関数です。

FIPS モードで無効な関数	代用に推奨される関数
<code>string.hashMD5()</code>	<code>string.hashSHA1()</code>
<code>ssl.serverCertHash()</code>	<code>ssl.serverCertSHA1()</code>
<code>ssl.clientCertHash()</code>	<code>ssl.clientCertSHA1()</code>

PKI 証明書

公開鍵基盤(PKI)証明書および鍵は、FIPS 140-2 準拠のデプロイの一部と見なす必要がある、暗号化アセットです。

注: トラフィックマネージャは、RSA アルゴリズムを使用する PKI のみをサポートしています。

鍵強度

FIPS 140-2 のドキュメントでは直接言及されていませんが、SP 800-131A では、暗号化動作で使用される鍵の許容強度に関する一連の要件が取り扱われています。一般に、これらの要件は FIPS 140-2 準拠暗号化モジュールによって強化されるため、FIPS モードを有効化する前に検討する必要があります。この文脈における「鍵強度」の実際の定義についての公式の説明は、上の「FIPS 140-2」のセクションで説明した SP 800-51 パート 1 の刊行物に詳述されています。

署名アルゴリズム

証明書によって示される鍵の強度のほかに、証明書自体にも暗号化シグネチャが含まれています。これにより、発行者によって認証された証明書と同じ証明書であることが立証されます。

暗号化シグネチャを作成および認証するために使用されるアルゴリズムに対しては、FIPS モードにおいて FIPS 140-2 準拠暗号化モジュールで使用できることが承認され

ていなければなりません。FIPS モードでは、署名ダイジェストが FIPS 承認済みの SHA-1、SHA256、SHA384、および SHA512 アルゴリズムを使用する証明書のみが使用可能です。

トラフィックマネージャの SSL 設定

FIPS モードを有効化する前に、以下に関して、トラフィックマネージャ SSL カタログ(**Catalogs** (カタログ) > **SSL** から確認)を監査する必要があります。

- 1024 ビット未満のモジュラス値を備えた証明書および鍵
- 未承認のダイジェストアルゴリズムを使用する署名を備えた証明書(通常、MD2 および MD5)

この方法で FIPS モードを有効化しようとする、現在の設定の中で動作の進行の妨げになる全ての証明書が検証エラーによってハイライトされます。

代わりに、インストールされたトラフィックマネージャに含まれているコマンドライン「**cert**」ツールを使用して、個々の証明書の評価することができます。例えば、SSL サーバ証明書の「**example**」証明書ファイルをチェックするには、以下のコマンドを使用します。

```
$ZEUSHOME/admin/bin/cert --format text \  
--in $ZEUSHOME/zxtm/conf/ssl/server_keys/example.public
```

このような「脆弱なアセット」が特定された場合は、トラフィックマネージャデプロイにおけるそのアセットの目的を評価し、本当にそのアセットが必要なのか、またそのアセットは、FIPS モードの有効化を目的にトラフィックマネージャデプロイを適切に動作させるうえで変更/更新/廃止されるその他のシステム/サービスにとって必要なかを判断する必要があります。FIPS モードを有効化するためには、まずこのような「脆弱なアセット」を取り除く必要があります。

重要: 証明書および秘密鍵を削除すると、トラフィックマネージャデプロイとの接続性に影響することがあります。例えば、SSL によってプール内のノードに接続する際に、SSL クライアントが「脆弱な」CA によって署名された証明書の提示を求められたり、トラフィックマネージャ自体が特定の証明書の提示を求められたりすることがあります。このような場合は、「脆弱なアセット」を取り除くことによってクライアントとトラフィックマネージャが SSL コネクションを行わなくなります。

デプロイの検討事項

これまでに説明した事項に加えて、トラフィックマネージャを動作させるデプロイについても検討する必要があります。

FIPS モードで動作中、トラフィックマネージャは暗号化「脆弱アセット」を処理するリクエストを、それがローカルの設定によるものか、リモートシステムから受信したものかを問わず拒否します。

重要: SSL によってトラフィックマネージャに接続する際、またはトラフィックマネージャが SSL によってプールノードに接続しているときに、仮想サーバクライアントに証明書を提示させようとする場合、仮想サーバクライアントおよびプールノードが提示する証明書は FIPS モードでの動作に適したものでなければなりません。

証明書チェーン

証明書チェーンを使用して動作しているとき、トラフィックマネージャがその証明書チェーンを FIPS モードで適切に使用するためには、チェーン内の全ての証明書が「脆弱」であってはなりません。

鍵交換強度

暗号スイートのなかには、ディフィーヘルマン鍵交換アルゴリズムを使用するものもあります。SP 800-131A には鍵の許容強度に関する一連の要件も含まれており、その要件はディフィーヘルマンアルゴリズムにも適用されます。

トラフィックマネージャが使用するディフィーヘルマン鍵のサイズは、`ssl!ssl3_diffie_hellman_key_length` 設定キー(**System (システム) > Global Settings (グローバル設定)** ページの「**SSL Configuration (SSL 設定)**」セクション)を使用して設定することができます。

有効性の確保/維持

FIPS 140-2 準拠暗号化モジュールの有効性は、認定を受けたテスト研究所が実施するテストによって定義されます。ソフトウェア暗号化モジュールに対するテストは、特定のハードウェア/OS プラットフォーム(運用環境)で実施されます。FIPS 140-2 準拠暗号化モジュールが認証済みの運用環境で使用されている場合、そのモジュールは FIPS 140-2 認証済みであると見なすことができます。

RCSM が認証されている運用環境の詳細については、NIST の Web サイトに掲載されている一覧で確認できます。

テストが、限定された数の運用環境において実施された場合(さらに、その環境で使用される BCSM が FIPS 140-2 に対する明示的な有効性を備えている場合)、認証済みの FIPS 140-2 準拠暗号化モジュール(レベル 1)は、(FIPS 140-2 に対して)明示的に認証されていない環境で運用する場合でも、その有効性は失われないと理解されています。

FIPS 140-2 に関する実施要項

(<http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf>)では、G.5 のセクションでこの件が取り扱われています。そこでは、認証済みとして明示的にリストされていないプラットフォームにおいて FIPS 140-2 に対する有効性を維持するために、ベンダー(BCSM の場合は Brocade)またはユーザ(トラフィックマネージャ製品の管理者など)が従わなければならない「移植」ルールが取り扱われています。この場合も、FIPS 140-2 準拠暗号化モジュールが有効であるかどうかを判断する総合的な裁定者は NIST (具体的には CVMP)であることに注意してください。

重要: ご使用のトラフィックマネージャ製品の BCSM で、FIPS 140-2 に対する有効性が維持されているかどうかについてご不明な点があれば、サポート窓口にお問い合わせください。

Linux 以外の OS

BCSM の有効性は、Linux OS のバージョンに対してのみテストされています。Solaris/SunOS などの Linux 以外のシステムに対するサポートおよび有効性は提供されていません。

FIPS モードの有効化

前述の方法に従ってトラフィックマネージャデプロイを準備すると、管理 UI の **System (システム) > Global Settings (グローバル設定)** ページにアクセスし、fips!enabled 設定キーを **Yes (はい)** に設定することで FIPS モードを有効化できます。

注: fips!enabled は、ロケーションごとに変更できないクラスタ全体の設定です。

トラフィックマネージャのリスタート

トラフィックマネージャを完全に FIPS モードで動作させるためには、fips!enabled キーを設定するだけでは十分とは言えません。クラスタ内の各トラフィックマネージャインスタンスをリスタートさせる必要があります。

重要: トラフィックマネージャをリスタートすると、アクティブなコネクションが全て切断されます。デプロイの可用性を高め、サービスの中断を最低限に抑えるために、この操作をスケジュールして実行することをお奨めします。詳細については、ご使用の製品の『**Installation and Getting Started Guide**』を参照してください。

バリデーション

FIPS モードを有効化してから管理 UI または SOAP ベースのコントロール API を使用する場合、トラフィックマネージャはご使用の設定に対するバリデーションを実行し、FIPS モードに対する様々な基準が満たされているか確認します。

FIPS モードと競合する設定要素があれば、全てエラーとして報告されます。そのようなエラーが発生した場合は、上の「準備」のセクションで確認したエラーに関連する記載を参照してください。

FIPS モードでの動作

FIPS モードで動作しているとき、クラスタ内の各トラフィックマネージャは、FIPS モードでの動作に異常がないかについてイベントログで報告します。以下のようなメッセージが記録されます。

```
FIPS 140-2 Cryptographic module loaded: FIPS Mode in operation
```

FIPS モードのエラー

FIPS モードを有効化しようとしているときにエラーが発生した場合、トラフィックマネージャがイベントログでそのエラーを報告します。トラフィックマネージャコンポーネントで FIPS モードの障害が発生した場合、そのコンポーネント内の暗号化動作(およびその動作を使用するサービス)は無効化されます。

例えば、1 台は SSL 復号化あり、もう 1 台は SSL 復号化なしという 2 台の HTTP 仮想サーバを備えたトラフィックマネージャがあるとします。FIPS モードの初期化が正しく行われなかった場合、SSL 復号化仮想サーバは無効化され、一切のトラフィックは処理されなくなりますが、もう一方の仮想サーバは正常に動作し続けます。

重要: FIPS モードが動作した状態でトラフィックマネージャが適切に起動したとしても、まれではありますが、動作中に FIPS 140-2 の要件に対する適合性が失われることがあります。このような場合、初期化の失敗時と同じ動作が想定されます。つまり、初期化に失敗したトラフィックマネージャからエラーが報告され、FIPS モードで障害が発生した場合は、暗号化動作が実行不能になります。

CHAPTER 32 Kerberos 強制委任のサポート

Kerberos プロトコル

Kerberos プロトコルを使用することで、エンティティ(プリンシパルとも呼ばれる)は、広く知られた信頼できる鍵配布センタ(KDC)サービスによって互いに認証し合うことができます。あるレルム用の KDC は、そのアドレス系に関連付けられたプリンシパルを、そのプリンシパルによって提示されたクレデンシャルを検証することによって認証します。その結果、レルムで機能するプリンシパルどうしが交換できるチケットが生成され、認証されたピアツーピアの情報交換を確立することができます。KDC がユーザに対して発行するプリンシパルへのチケットには、プリンシパルと KDC のみによって共有される秘密を使用して暗号化された、そのユーザに対する識別データが保持されています。

注: Microsoft による Kerberos プロトコルの実装では、レルムは **ドメイン**と呼ばれ、KDC サービスは **ドメインコントローラ**と呼ばれています。

Kerberos プロトコルの遷移と制約付き委任

クライアントのプリンシパルが一部を成すレルムの KDC に、クライアントが直接アクセスできる場合、「Kerberos 対応」のアプリケーション間の Kerberos 認証は正常に機能します。しかし、KDC へ直接アクセスできない場合、ユーザは、認証のために Kerberos チケットの交換を要求するサービスに必ずしもアクセスできるとは限りません。例えば、パブリックインターネットに単独で接続されているラップトップコンピュータは、ファイアウォールのルールによって、ラップトップユーザの KDC に直接アクセスできないことがあります。このような事態に対処するために、プロトコル遷移と制約付き委任という、Kerberos プロトコルに対する 2 つの拡張機能が用意されています。

プロトコル遷移

これは、適切に特権が与えられた Kerberos サービスが、特定のレルムの任意のユーザプリンシパル用にそのサービス自体へのチケットを取得できるようにするメカニズムです。KDC は、Kerberos プロトコルでのユーザ用のチケットを生成する前に、その他の手段によって認証を実施し、ユーザの身元を確認することをサービスに対して要求します。すなわち、サービスは 1 つの認証プロトコルから Kerberos への遷移を提供します。

制限付き委任

プロトコル遷移を提供しているサービスでは、ユーザの身元確認はできているため、そのサービスに対する任意のユーザプリンシパル用のチケットを持つことの効果は

限定的です。制限付き委任を使用すると、サービスは、そのサービスに対するチケットを別のサービスに委任することができます。つまり、そのチケットをそのサービスから(元のユーザに関連付けられたリクエストとともに)さらに先の **Kerberized** サービスに転送できるようになります。この新しいサービスは、転送されたリクエストを元のユーザに照らして処理します。

レルム内の任意のユーザになりすませることは、サービスにとって強力すぎる権限に思われるかもしれませんが、この機能には「制限付き」という側面があるため、KDC は、チケットが委任されるサービスの選択において制限を課すことができます。

Kerberos を使用する論理的根拠

トラフィックマネージャとの関連において、**Kerberos** プロトコル遷移は、トラフィックマネージャが内部ネットワークの境界上に位置し、内部サービスへのアクセスを提供している場合に活用できます。**Kerberos** を使用して認証を行っている社内 LAN 上の内部サービスは、KDC へのアクセスが許可されていない、よりパブリックなネットワークで利用することができます。トラフィックマネージャは、代替手段を使用してユーザのプリンシパル名を特定し、プロトコル遷移を行います。

トラフィックマネージャは、TLS クライアント証明書ベースの認証からのプロトコル遷移をサポートしています。

トラフィックマネージャでの Kerberos 設定

トラフィックマネージャが **Kerberos** をサポートしていることによって、TLS クライアント証明書を使用したユーザの認証が可能になります。トラフィックマネージャは、それ自体のサービスプリンシパルを使用してレルムの **KDC** と通信を行い、仮想サーバでリクエストを行うクライアントに代わって、バックエンドノードで実行されているサービスへのチケットを取得します。

トラフィックマネージャのサービスプリンシパル

Kerberos のレルムに参加するには、トラフィックマネージャに、自らを特定するために使用する、適切に権限が与えられたプリンシパルが備わっていなければなりません。ご使用の身元管理システムおよび関連付けられた **KDC** 設定には、トラフィックマネージャ用の **Kerberos** プリンシパルが備わっていることを必ず確認してください。また、トラフィックマネージャが管理しているサービスへのプロトコル遷移および制限付き委任が許可されているように設定する必要もあります。

クラスタ内の各トラフィックマネージャに対して、**KDC** 管理者は、トラフィックマネージャのサービスプリンシパルが他のサービスに委任できるように **KDC** を設定する必要があります。また、サービスプリンシパルの名前は、トラフィックマネー

ャが使用する名前と一致しなければなりません。これは、以下の形式の文字列で記します。

<service>/<Traffic Manager hostname>@<REALM>

この文字列の<service>および<REALM>の値には、Kerberos プリンシパルカタログから使用したい Kerberos プリンシパルレコードの値を使用します。

例えば、トラフィックマネージャのホスト名が「stm1.example.com」で、サービス名が「HTTP」、レルムが「EXAMPLE.COM」の Kerberos プリンシパルレコードを使用して接続する場合、KDC は「HTTP/stm1.example.com@EXAMPLE.COM」と指定されたプリンシパルを信頼し、エンドサービスへの委任を行います。

Microsoft 実装では、このことは、トラフィックマネージャに対するコンピュータアカウントがドメイン上に作成され、トラフィックマネージャ上の関連するサービスにサーバプリンシパル名が設定され、それらのサービスに対してトラフィックマネージャへの委任が許可されたことを意味します。

トラフィックマネージャが、作成されたプリンシパルとしてトラフィックマネージャ自体を認証するためには、適切なクレデンシャルを指定する必要があります。ユーザプリンシパルに対しては、パスワード入力の形式でクレデンシャルを指定します。インタラクティブなパスワード入力サービスプリンシパルとして不便なため、代わりに「Keytab」ファイルをトラフィックマネージャサービスプリンシパルのクレデンシャルに使用しなければなりません。Keytab ファイルには、1 つ以上のプリンシパルに対するクレデンシャルを保持させることができます。

Kerberos のレルムでプロトコル遷移を実施するようにトラフィックマネージャを設定するには、クラスタ内の全てのトラフィックマネージャに対するクレデンシャルを保持する Keytab ファイルをアップロードし、Keytab ファイルを使用するように Kerberos プリンシパルを設定します。

Keytab ファイルはトラフィックマネージャの Keytab カタログに保存されます。Keytab ファイルをカタログにアップロードするには、管理 UI の **Catalog** (カタログ) > **Kerberos** > **Kerberos Keytabs** ページを使用します。

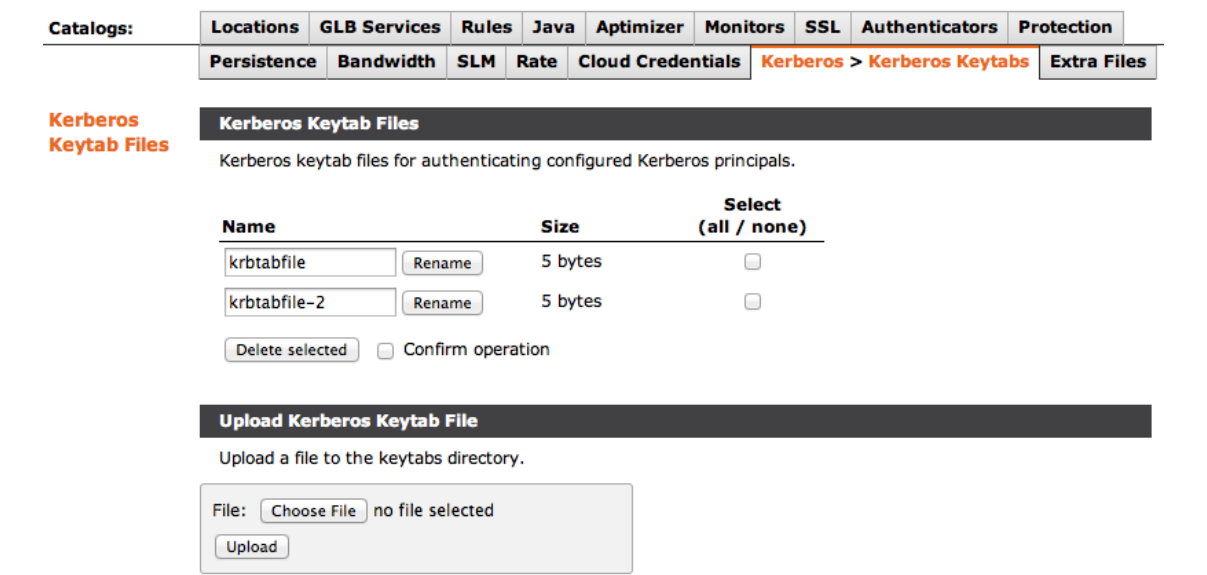


Fig 83. Kerberos Keytabs カタログ

Kerberos プリンシパルのカタログ項目を作成するには、管理 UI の **Catalog** (カタログ) > **Kerberos** > **Kerberos Principals** (Kerberos プリンシパル) ページを使用します。各プリンシパルに対して、以下のフィールドを指定します。

設定項目	説明
name	プリンシパル設定の識別名です。
service	トラフィックマネージャがトラフィックマネージャ自体を認証するために使用する Kerberos プリンシパル名のサービス名要素です。
keytab	このプリンシパル用のクレデンシャルが保持される Kerberos Keytabs カタログの Keytab ファイルの名前です。
realm	Kerberos プリンシパルが一部を成すレルムです。
kdc	レルムに使用する KDC サービスを明確に指定するための、<host>:<port>のペアのオプションリストです。リストを指定しない場合、トラフィックマネージャは DNS を使用して代わりに使用すべき KDC を特定します。
krb5conf	このプリンシパルと使用する、あらかじめ定義された Kerberos 設定ファイルの名前です(オプション)。Kerberos 設定ファイルを使用するには、まず管理 UI の Catalog (カタログ) > Kerberos > Kerberos Configuration Files (Kerberos 設定ファイル) ページを使用して、設定ファイルをアップロード

設定項目	説明
	ードする必要があります。

仮想サーバプロトコル遷移の設定

仮想サーバで Kerberos プロトコル遷移を使用するには、**Virtual Servers** (仮想サーバ) > **Edit** (編集) > **Kerberos Protocol Transition** (Kerberos プロトコル遷移) ページを使用します。

Configuring: Traffic IP Groups Virtual Servers > vs1 > Kerberos Pools Config Summary

Kerberos Protocol Transition

Virtual Server: vs1 (HTTP, port 80)

Kerberos Protocol Transition allows clients to use services that use Kerberos authentication without having Kerberos tickets themselves.

Kerberos Protocol Transition settings

These settings control Kerberos Protocol Transition for this virtual server.

Whether or not the virtual server should use Kerberos Protocol Transition
kerberos_protocol_transition!enabled: ☐ Yes ☒ No

The Kerberos principal this virtual server should use to perform Kerberos Protocol Transition.

	Name	
kerberos_protocol_transition!principal:	☒ Not specified	
	☐ p1	Edit
	☐ p2	Edit

The Kerberos principal name of the service this virtual server targets.
kerberos_protocol_transition!target:

Apply Changes

[Update](#)

Fig 84. 仮想サーバ Kerberos プロトコル遷移ページ

Kerberos を有効化するには、**kerberos_protocol_transition!enabled** を Yes (はい) に設定します。このページを使用して、オプションで Kerberos プリンシパルと、仮想サーバが処理するターゲットのサービスに対するプリンシパル名を設定できます。

注: プロトコル遷移が適切に実行されるためには、トラフィックマネージャとターゲットのプリンシパルの両方を設定する必要があります。これらの設定は、仮想サーバまたはプールのいずれかでを行います。

トラフィックマネージャは、クライアント SSL 証明書からクライアントプリンシパルを取得する必要があります。これを可能にするには、仮想サーバで **require_cert** と併せて **ssl_decrypt** も有効化し、クライアントにクライアント証明書を求めるように設定します。クライアントが適切に認証されるためには、適切な認証局と OCSP を仮想サーバに設定してください。

プールプロトコル遷移の設定

Pool (プール) > **Edit** (編集) > **Kerberos Protocol Transition** (Kerberos プロトコル遷移) ページを使用して、Kerberos プロトコル遷移を設定することもできます。

The screenshot shows the configuration page for Kerberos Protocol Transition for a specific pool. At the top, there is a navigation bar with tabs: "Configuring:", "Traffic IP Groups", "Virtual Servers", "Pools > p1 > Kerberos" (which is highlighted), and "Config Summary". On the left side, there is a sidebar with the text "Kerberos Protocol Transition". The main content area has a header "Pool: p1" and a description: "Kerberos Protocol Transition allows clients to use services that use Kerberos authentication without having Kerberos tickets themselves." Below this is a section titled "Kerberos Protocol Transition settings" with the text "These settings control Kerberos Protocol Transition for this pool." The main configuration area contains two sections. The first section is titled "The Kerberos principal the traffic manager should use when performing Kerberos Protocol Transition." and contains a table with the following data:

	Name	
kerberos_protocol_transition!principal:	☒ Not specified	
	☐ p1	Edit
	☐ p2	Edit

The second section is titled "The Kerberos principal name of the service this pool targets." and contains a text input field labeled "kerberos_protocol_transition!target:".

At the bottom of the configuration area, there is a section titled "Apply Changes" with an "Update" button.

Fig 85. Pool Kerberos Protocol Transition (プール Kerberos プロトコル遷移) ページ

このページでプリンシパルまたはターゲットを指定すると、この設定が、このプールを使用する仮想サーバに存在する全ての Kerberos 設定をオーバーライドします。

CHAPTER 33 トラブルシューティング

本章では、トラフィックマネージャが正常に動作しているかどうかをテストする方法と、発生した問題の詳しい対処方法について説明します。

ツールおよび手法

トラフィックマネージャの設定をテストする場合、様々なツールおよび手法を利用して、設定およびネットワークトラフィックの挙動を把握することができます。

- **診断: Cluster Diagnosis** (クラスタ診断) ページでは、トラフィックマネージャシステムにわたってネットワークコネクション、ヘルスマニタ、設定エラー、および設定の競合を確認する様々なテストを実行できます。
- **イベントログ: イベントログ** (ホームページおよび **Event Log** (イベントログ) ページに表示) には、発生した重要イベントが記録されます。
- **コネクション: Connections** (コネクション) レポートには、継続中および最近のコネクションについて、接続元、処理方法、転送されたデータ量などが即座に表示されます。
- **リクエストロギング**: トラフィックマネージャでは完全なログを書き込むことができますため、各リクエストおよびレスポンスの詳細を全て記録し、後で解析に利用することができます。
- **設定概要**: 高度な設定を行った場合も、**Configuration Summary** (設定概要) で、使用されている設定オブジェクトおよび相互の関係を確認することができます。
- **監査ログ**: **監査ログ** には、全ての設定の変更について、変更がいつ行われたか、どのユーザが行ったか、そのユーザはどこに属しているかが記録されます。

本章では、これらの各ツールと手法について詳しく説明します。

診断とイベントログ

トラフィックマネージャシステムまたは設定で問題が発生した場合、まず、トラフィックマネージャ管理サーバの **Cluster Diagnosis** (クラスタ診断) ページを確認してください。

重要: クラスタ内のどのトラフィックマネージャからも管理 UI にアクセスできない場合は、トラフィックマネージャソフトウェアを起動またはリスタートすることをお奨めします。この手順は、ご使用の製品の『**Installation and Getting Started Guide**』に記載されています。エラーメッセージ、特に ZEUSHOME/zxtm/log/errors のエラーログファイルに追加された全てのメッセージに十分に注意を払ってください。

管理サービインターフェースの任意のページにある **Diagnose** (診断) ボタンをクリックすると、現在のシステムステータスを確認できます。**Diagnose** (診断) ページは複数のセクションに分かれており、発生した問題を理解し、解決する上で役立ちます。

Cluster Diagnosis (クラスタ診断) セクションには、使用不可のマシンや不正な設定など、トラフィックマネージャが検出した問題が報告されます。

イベントログには、各トラフィックマネージャマシンのエラーログの内容が表示されます。最近作成されたログを、期間を指定して表示することができます。**Event Log** (イベントログ) ページは、新しいイベントメッセージがログに追加されると動的に更新されます。

監査ログには、最近行われた設定の変更について、それがいつ行われたか、どのユーザが行ったかが表示されます。

Technical Support (技術サポート) セクションには、様々な資料への便利なリンクのほか、必要に応じて Brocade サポートチームに送信する技術サポートデータをダウンロードするためのオプションが表示されます。

注: Brocade Services Director デプロイ内でトラフィックマネージャインスタンスを使用している場合、お問い合わせいただく際には、このセクションに表示される **SSC サポート資格** の識別子を使用してください。Brocade は、お客様のサポート資格レベルをユニークに特定するためにこの情報を使用します。

通常の運用時に問題が検出された場合、**ステータスアプレット** に警告またはエラー状態が示されます。**Cluster Diagnosis** (クラスタ診断) ページに移動して、問題の詳細を調べてください。



Fig 86. 「エラーなし」時と「エラー検出」時のクラスタ警告画面

リクエストとレスポンスのモニタリング

観察された問題に関するエラーが **Cluster Diagnosis** (クラスタ診断) ページに報告されない場合、トラフィックマネージャがトラフィックを正しく管理しているか、すなわちバックエンドサーバに代わってリクエストを読み取り、バックエンドサーバからのレスポンスをリモートクライアントに返しているかどうかを検証します。

問題を再現するときに、CHAPTER 28 の「テストリクエストの生成」が役立つ場合があります。

コネクション状況レポート

トラフィックマネージャ管理サーバの **Activity** (アクティビティ) ボタンをクリックし、**Connections** (コネクション) タブを選択します。**Connections** (コネクション) ページには、トラフィックマネージャが管理した全ての継続中のコネクションおよび最近完了したコネクションのレポートが表示されます。**Key** (キー) セクションを展開すると、その情報の詳細が表示されます。

現在コネクションを処理しているトラフィックマネージャとそれに関与しているバックエンドノードや現在のコネクションの状態に加え、コネクションに関与している仮想サーバ、ルール、およびプールを確認できます。また、コネクションの再試行回数、アイドル時間、およびバイト数も表示されます。

データを最新の状態にするには、インターネットブラウザの更新ボタンをクリックします。データを .tsv ファイルとしてダウンロードすることもできます。

Connections (コネクション) レポートの使用

注: テストしているリクエストがトラフィックマネージャによって処理されたかどうかを検証します。このリストにテストリクエストが表示されない場合、トラフィックマネージャがリクエストを受信しなかった可能性があります。クライアント設定と DNS 設定を確認し、テストクライアントがトラフィックマネージャの IP アドレスにリクエストを送信しているかどうか、またリクエストがいずれかのバックエンドサーバで処理されているかどうかを検証してください。

一般的な問題として、バックエンドサーバが自己参照型のレスポンスを発行していることが挙げられます。例えば、トラフィックマネージャが `www.example.com` 宛てのトラフィックを処理していて、バックエンドサーバの名前が `server1.example.com` および `server2.example.com` であるとしみます。一方のサーバが `server1.example.com` へのリンクまたはリダイレクトを含むレスポンスを発行すると、以降、クライアントはトラフィックマネージャをバイパスしてバックエンドサーバと直接通信しようとするため、問題の原因となります。このタイプの問題は、レスポンスに DNS 名または IP アドレスを埋め込むその他のプロトコルでも発生する可能性があります。

コネクションレポートのサイズは、**System** (システム) > **Global Settings** (グローバル設定) ページで拡張することができます。**recent_conns** キーの **Logging** (ロギング) セクションを確認してください。

リクエストログ

仮想サーバのリクエストロギング機能を使用して、トランザクションの様々な側面を記録することができます。この機能は、断続的なエラーや再現が難しいエラーについて、長期的なテストを行う場合に非常に役立ちます。設定手順については、CHAPTER 4 の「*Request Logging* (リクエストロギング)」セクションを参照してください。

TrafficScript ロジックが複雑な場合、コネクションのデバッグ情報を保存し、それをリクエストログに記録することができます。TrafficScript ルールで `connection.data.set()` 関数を使用してコネクションに関する特定データを保存し、次に「`%{Key-name}d`」マクロを使用してそのデータをリクエストログに記録します。

高度なロギング

TrafficScript ルールを使用して、仮想サーバが処理したトランザクションに関する情報を記録することができます。

TrafficScript 関数は、リクエストのあらゆる部分の情報を読み込むことができます。`http.getHeader()` や `http.cookie()` などの関数を使用すると、HTTP リクエストに関する情報を取得できます。また、`ssl.getSessionID()` などの関数を使用すると、SSL トラフィックに関する情報を取得できます。その他のプロトコルの場合は、`connection.getData()` 関数を使用して、リクエストの指定バイト数の情報を取得することができます。

`log.info()` 関数を使用すると、受信したリクエストの詳細をトラフィックマネージャのイベントログファイルに書き込み、これを基に受信トラフィックで発生している問題を突き止めることができます。また、`event.emit()` 関数を使用し、適切なイベントハンドラを設定すると、デバッグメッセージの記録場所をより厳密にコントロールすることができます。

TrafficScript 関数の詳細については、『**TrafficScript Reference**』を参照してください。

重要: 全ての受信リクエストについてこの処理を行うと、ログファイルが急速に増大します。このような情報を記録する場合はご注意ください。

イベントのモニタリング

TrafficScript 関数の `counter.increment()` を使用すると、特定イベントの発生回数をカウントすることができます。詳細については、CHAPTER 7 の「*Activity* (アクティビティ) > *Current Activity* (現在のアクティビティ)」セクションを参照してください。

コネクションの詳細なデバッグ

コネクションに失敗すると、コネクション状況レポートおよびリクエストログに記録される情報が不完全になります。これは、コネクションが正常に完了しなかったためです。コネクションの失敗原因は、プロトコルエラー、タイムアウト、予測外の TCP 終了など様々です。

このような予測外の問題をデバッグするには、CHAPTER 4 の「エラーの処理」セクションで説明しているように、影響を受けたサービスの **Virtual Server (仮想サーバ) > Connection Management (コネクション管理)** 設定ページで

log!client_connection_failures および **log!server_connection_failures** 設定を有効化します。これらの設定により、コネクションに失敗した場合に必ず詳細なデバッグメッセージをイベントログに書き込むようトラフィックマネージャが設定されます。

Event Log

The Event Log shows the contents of the error logs on each traffic manager machine.

Timescale:

Event Filter:

Show 'Configuration modified' messages:

Last N events

15

Connection Failures

Yes ☐ No ☒

Stop updating

Display

Download

✓	15/Jul/2009:11:56:46 +0100	INFO	Virtual Server zws-8765: "Invalid HTTP response" S 10.100.1.186:38653 10.100.1.186:8765 "zws-18765" "yinkin:18765" "-" r 71 106 0 0 0 0 0 500 GET "http://yinkin:8765/nph-junk.cgi?foo=bar&quote=%22&tab=%09"	yinkin
✓	15/Jul/2009:11:56:45 +0100	INFO	Virtual Server zws-8765: "Pool has no back-end nodes responding" 10.100.1.186:38651 10.100.1.186:8765 "testtcp" "-" c 38 0 0 0 0 0 1 - GET "http://10.100.1.186:8765/"	yinkin
✓	15/Jul/2009:11:56:45 +0100	INFO	Virtual Server zws-8765: "Connect failure - Connection refused" S 10.100.1.186:38651 10.100.1.186:8765 "testtcp" "yinkin:13765" "-" c 38 0 0 0 0 0 0 - GET "http://10.100.1.186:8765/"	yinkin

Fig 87. イベントログの Connection failure レポート

上図のような **Connection Failures (接続失敗)** イベントフィルタを使用して、接続失敗に関するメッセージのみが表示されるようイベントログをフィルタすることができます。

ログ行には、エラーの記述(「Invalid HTTP Response」など)と、コネクションの性質についての詳細(使用ノード、接続回数、再試行回数など)が表示されます。ログフォーマットの詳細は、**Virtual Server (仮想サーバ) > Connection Management (コネクション管理)** 設定ページのオンラインヘルプで参照することができます。

エラーの記述部分をクリックすると、接続エラーの内容が分かりやすい形で表示されます。

✓	15/Jul/2009:11:56:46 +0100	INFO	Virtual Server zws-8765: "Invalid HTTP response"		 yinkin
			Peer	Server	
			Source Address	10.100.1.186:38653	
			Destination Address	10.100.1.186:8765	
			Pool	zws-18765	
			Node	yinkin:18765	
			Rule	-	
			Connection State	Reading from server	
			Bytes from client	71	
			Bytes to server	106	
			Bytes from server	0	
			Bytes to client	0	
			Connection Established	0s	
			Client Idle Time	0s	
			Server Idle Time	0s	
Connection Retries	0				
Response Code	500				
Request Line	GET "http://yinkin:8765/nph-junk.cgi?foo=bar&quote=%22&tab=%09"				

Fig 88. コネクションエラー詳細

デフォルトでは、コネクションエラーはグローバルイベントログに記録されます。全ての接続失敗イベントをキャプチャし、グローバルイベントログをバイパスして別のログファイルに書き込むイベントハンドラを作成できます。詳細については、CHAPTER 21 の「概要」セクションを参照してください。

個々のノードのテスト

プール内の各バックエンドノードがトラフィックマネージャからリクエストを受信できるかどうかを確認するには、プールの負荷分散アルゴリズムをラウンドロビンに切り替えます(CHAPTER 5 の「負荷分散」セクションを参照)。「ラウンドロビン」負荷分散アルゴリズムでは、リクエストが各ノードに順番にルーティングされるため、バックエンドサーバのログを使用して、全てのノードでこのアルゴリズムが正常に機能しているかどうかを確認することができます。

コネクションをドレインするようノードを設定することもできます(CHAPTER 5 の「ノードのドレインと無効化」セクションを参照)。プールが複数のノードで構成されている場合、一部のノードをドレインするよう設定して、トラフィックマネージャから一時的にこれらのノードにトラフィックが送信されないようにできます。テストするノードを除く全てのノードをドレインすることにより、各ノードを順番にテストすることができます。

各ノードでサーバロギングを有効化し、ノードによって書き込まれたログとシステムのアクティビティが一致していることを検証してください。

設定の確認

Configuration Summary (設定概要)には、トラフィックマネージャでリクエストがどのように処理されたかを示す概要が表示されます。リクエストは、トラフィックマネージャが受信し、仮想サーバおよびプールでの処理後にノードに転送され、その後、戻り経路上にあるプールおよび仮想サーバによってレスポンスが処理されます。

多数の TrafficScript ルールおよびトラフィックマネージャ機能が有効になっている状態では、処理経路が非常に複雑になる可能性があります。

Configuration Summary (設定概要)については、CHAPTER 7 で説明しています。

調査対象の問題が発生した時点を突き止めることができれば、**Audit Log** (監査ログ) (管理サーバの **Diagnose** (診断)部内)と、場合によっては **Backup Management** (バックアップ管理) (管理サーバの **System** (システム)部内)を使用して、問題の原因となった可能性のある設定の変更を特定することができます。

監査ログには、トラフィックマネージャ管理サービインターフェースの各ユーザが行った設定の変更が全て記録されます。認可された各管理者に個別のアカウントを用意すれば、これらの管理者の権限を管理し、そのアクティビティをモニタリングすることができます。ユーザおよびグループの設定については、CHAPTER 24 の「ユーザ管理」セクションで説明します。

Backup Management (バックアップ管理)では、手動で設定のスナップショットを作成することができます。現在の設定と正常に動作していたときのスナップショットを比較し、設定に重大な問題がある場合にバックアップを復元することができます。Backup Management (バックアップ管理)については、CHAPTER 7 の「System (システム)> Backups (バックアップ)」セクションで説明します。

トラブルシューティングのヒント

トラフィック管理される複雑なクラスタで発生した障害の診断プロセスは単純ではないため、体系的なアプローチでトラブルシューティングを行う必要があります。問題が発生した場合、以下の事柄を順番に確認し、障害の原因を突き止めて診断してください。

テストリクエストの生成

システムのテストは、Web ブラウザや電子メールクライアントなどユーザ用の一般的なクライアントソフトウェアを使用して行うことができます。Mozilla ブラウザの Live HTTP Headers 拡張などのツールは、クライアントとトラフィックマネージャシステム間でやり取りされるリクエストおよびレスポンスの流れをチェックする場合に非常に役立ちます。

Wireshark (<http://www.wireshark.org/>)などのネットワークスプーフィングツールでも、ネットワークトラフィックを記録したり、リクエストおよびレスポンスセッションを作成したりすることができます。

httpclient

トラフィックマネージャに付属の httpclient プログラムを使用すると、特定のマシンに対して HTTP リクエストを送信することができます。このプログラムは ZEUSHOME/admin/bin/httpclient にあります。

以下の構文を使用します。

```
$ httpclient --hostheader=<website_name>  
http://<trafficIP>/
```

telnet クライアントを使用して、サービスに対して基本的なテストを行う初期テストを実行することもできます。

```
$ telnet www.mysite.com 80
Trying 62.254.209.66...
Connected to 62.254.209.66.
GET / HTTP/1.1<RETURN>
Host: www.mysite.com<RETURN>
<RETURN>
```

OpenSSL ツールキット(<http://www.openssl.org/>)には、SSL 関連の問題をテストするための、SSL プロトコルを使用する telnet に似たクライアントが含まれています。

zeusbench

ZeusBench プログラムは、多数の HTTP リクエストを送信して負荷および性能テストを実行できる便利なベンチマークツールです。このプログラムは `ZEUSHOME/admin/bin/zeusbench` にあります。

多数のコマンドラインオプションが用意されています。単純な負荷テストは、以下のコマンドを使用して実行できます。

```
$ zeusbench -t 30 -c 100 -k http://host/url
```

zeusbench -h を実行すると、全てのコマンドラインオプションのリストが表示されます。

自動バックエンドフェールオーバーの確認

トラフィックマネージャによるバックエンドの自動フェールオーバーを確認するには、フェールオーバー先のマシンが必要になるため 2 台以上のバックエンドサーバが必要です。フェールオーバーをテストするには、1 台のバックエンドマシンのネットワークケーブルを引き抜くか、そのバックエンドで実行中のサービスを手動で停止します。次に、バックエンドのそのポートで何も監視されていないことを確認します。

バックエンドサーバが 1 台のみの場合は、サービスの 2 つのインスタンスを同じサーバマシンの異なるポートで実行し、いずれかのインスタンスを手動で停止させることで代用できます。

トラフィックマネージャを通してサービスを利用してみます。SMTP の場合は、サーバファームを通して電子メールを送信します。HTTP の場合は、Web ページをリクエストします。少なくとも 1 台のバックエンドサーバでリクエストを満たすことができれば、このテストは成功です。

Diagnose (診断) ページのリンクからイベントログにアクセスし、バックエンドマシンでの障害発生を示す通知があることを確認してください。

自動フロントエンドフェールオーバーの確認

複数のトラフィックマネージャを使用している場合、自動フロントエンドフェールオーバーが正常に機能しているかどうかを確認することができます。全てのトラフィックマネージャにまたがるトラフィック IP グループと、このトラフィック IP グループを使用するサービス(ポート 80 で Web コンテンツを管理する仮想サーバなど)を設定します。このサービスの Web ページを、グループに含まれる各トラフィック IP アドレスでリクエストできることを確認します。これを行うには、ブラウザに DNS 名ではなく IP アドレスを入力します。

次に、管理サーバページの **Services** (サービス) ボタンをクリックします。**Traffic IP Groups** (トラフィック IP グループ) タブをクリックし、**Unfold All** (全て開く) をクリックしてトラフィック IP グループの詳細を表示します。ここには、どのマシンでどの IP アドレスが有効になっているかが示されます。マシンの台数がグループ内のトラフィック IP アドレスの数より多い場合、一部のマシンはスタンバイ状態になり、トラフィックをアクティブに処理しないことに注意してください。

次に、いずれかのトラフィック IP アドレスが有効になっている 1 台のマシンを選択し、そのマシンのネットワークケーブルを全て引き抜きます。これにより、その IP アドレスがグループ内の別のマシンで有効になります。ブラウザで再度そのトラフィック IP にアクセスし、コンテンツを受信できることを確認します。キャッシングされたコンテンツが使用されないよう、ブラウザでページを更新することが必要な場合があります。

CHAPTER 6 の「フォールトトレランスの設定」セクションでは、トラフィックマネージャのフォールトトレランスの動作、実行されるテスト、および決定について説明しています。

起こりやすい問題

root として起動されない

トラフィックマネージャは、メールや HTTP などのネットワークサービスの負荷分散を行うために特権ポートにバインドする必要があります(特権ポートはポート番号が 1024 より小さいポート)。これを行うには、インストール後、**configure** スクリプトを実行し、トラフィックマネージャを **root** として起動する必要があります。

フロントエンドフォールトトレランスモードでは、トラフィックマネージャは、ネットワークインターフェースを有効/無効にする必要があります。このソフトウェアを **root** として設定および起動しなかった場合、フェールオーバーアーキテクチャは機能しません。

トラフィックマネージャを非特権ユーザとして実行する場合でも、**root** ユーザとして設定して起動する必要があることに注意してください。

評価のために、**root** 以外のユーザでトラフィックマネージャを設定および起動することもできます。ただし、この場合は、特権ポートにバインドしたり(上記を参照)フォールトトレランスを使用したりすることはできません。

コネクションが拒否される

フロントエンドで最も起こりやすい設定エラーは、外部 DNS 名をフロントエンドマシンの外部 IP アドレスにマッピングした不正な DNS 設定です。フロントエンドの DNS 名でサーバに接続したときに「**Connection Refused (コネクション拒否)**」メッセージを受信した場合、DNS が正しく設定されていない可能性があります。

DNS 設定を確認するには、**host** または **nslookup** コマンドを使用します。

```
$ host www.w3.org
www.w3.org has address 18.29.1.35
www.w3.org has address 18.7.14.127
www.w3.org has address 18.29.1.34
```

トラフィックマネージャマシンがこれらの IP アドレスで監視していることを確認します。

これらの IP アドレスでトラフィックを監視するよう明示的に仮想サーバを設定することができます。これにより、トラフィックマネージャクラスタからこれらの IP アドレスを使用できない場合に、**Diagnose (診断)** ページにエラーが報告されるようになります。

トラフィック IP アドレスが正しく設定されていない

フロントエンドフォールトトレランスを実現するためにトラフィック IP グループを設定すると、選択したトラフィック IP アドレスが適切かどうかを確認するために、トラフィックマネージャによっていくつかのチェックが行われますが、念のため、これらの IP が現在のネットワーク構成に適しているかどうかを再確認してください。一般的な間違いとして、選択したトラフィック IP アドレスが他の場所で使用されている(トラフィックマネージャマシンの固定 IP アドレスなど)、トラフィックマネージャが使用しているサブネット上にない IP アドレスを選択した、選択した IP アドレスに他の場所からルーティングできない、などがあります。

トラフィックマネージャがプロトコル開始前にコネクションを破棄する

Server First プロトコルでは、トラフィックマネージャが *Server First* プロトコル用の特定ポートでバックエンドサーバと通信できない場合、クライアントとのコネクションを破棄せざるを得ません。(telnet などを使用して)トラフィックマネージャマシンの適切なポートに接続し、コネクションに成功しても(すなわち「**Connection Refused (コネクション拒否)**」メッセージが返されない)、その直後にコネクション

が破棄される場合、トラフィックマネージャはバックエンドと通信しようとしていますが、バックエンドサーバが見つからない状態にあります。

Diagnose (診断) ページのイベントログで、バックエンドサーバのステータスに関するメッセージを確認してください。

Web サーバが Error 400 を返す

トラフィック管理されている Web サイトのテストで Error 400 Bad Request タイプのエラーが返される場合、アクセスしようとしている Web サイトが、他のホストヘッダ(またはエイリアス)を受け入れるよう設定されていない可能性があります。この問題を解決するには、適切なホストヘッダを設定するルールを追加します。

```
http.setHeader( "Host", "www.mysite.com" );
```

間違ったポート番号が設定されている

この他、起こりやすい問題として、負荷分散のため使用しているポート番号が間違っていることが挙げられます。例えば、Web サイトを 80 以外のポートで運用している場合、HTTP プロトコルでデフォルトのポートではなくそのポート番号を使用して負荷分散を行う必要があります。SSL 暗号化サイトを運用している場合は、HTTP ではなく HTTPS プロトコルを選択するか、SSL トラフィックを復号化してください。HTTPS のデフォルトポートは 443 です。

適切なプロトコルが選択されていれば、1 つのトラフィックマネージャにより多数のポートで負荷分散を行うことができます。

ファイル記述子を使い果たした

トラフィックマネージャは、「ファイル記述子」と呼ばれる OS のリソースを使用して、クライアントまたはサーバへの各ネットワークコネクションを管理します。割り当てることのできるファイル記述子の数にはハードリミットがあり(CPU コア単位などプロセス単位で適用される)、この制限は OS で設定されます。

ベンチマーク時など負荷が非常に大きいときは、「ファイル記述子を使い果たした」ことを示すエラーメッセージが表示されることがあります。この場合、トラフィックマネージャが使用するファイル記述子の数を増やすことで対処できます。

管理インターフェースで **System** (システム) > **Global Settings** (グローバル設定) ページに移動します。設定の **System Settings** (システム設定) 部で、**maxfds** の値を大きくします。既に OS による許容最大数に達している場合、システムマニュアルを参照して、このハードリミットを増加させることができるかどうかを確認してください。

ディスク空き容量を使い果たした

注: このセクションは、アプライアンスおよびクラウド製品のみを対象としています。

時間とともに、アプライアンスのディスク空き容量が不足する可能性があります。例えば、詳細なリクエストログ情報を生成するようにトラフィックマネージャを設定している場合、システムログが大きくなることがあります。さらに、アーカイブされたソフトウェアの改訂版(トラフィックマネージャがロールバックで使用)が不要になっている可能性もあります。

ディスク空き容量が不足し始めると、トラフィックマネージャはイベントログおよび **Diagnose (診断) > Cluster Diagnosis (クラスタ診断)** ページで警告を発します。また、**System (システム) > Traffic Managers (トラフィックマネージャ)** ページからは、いつでもディスク使用容量を確認できます。

ディスク空き容量を確保するには、トップナビゲーションバーにある **Wizards: (ウィザード:)** ドロップダウンメニューから「Free up some disk space (ディスク空き容量の確保)」をクリックします。また、このウィザードは、**System (システム) > Traffic Managers (トラフィックマネージャ)** ページの「Free Disk Space (ディスク空き容量の確保)」リンクから、またディスク空き容量の警告が表示されているときは **Diagnose (診断) > Cluster Diagnosis (クラスタ診断)** ページからいつでも実行できます。

重要: この操作は取り消すことができません。ウィザードを実行する前に、残しておかなければならない全てのファイルのバックアップを必ず作成してください。また、その後作成する「テクニカルサポートレポート」には、ウィザードが実行された後に生成されたログしか含まれないことにも注意してください。

サポートを利用するには

技術サポートエンジニアに問い合わせるときは、テクニカルサポートレポート(TSR)をダウンロードし、プロブレムレポートに添付してください。テクニカルサポートレポートには、最新のログファイル、設定、システムのアクティビティ、および様々な内部情報が含まれています。

TSR を作成、ダウンロード、および管理するには、管理 UI の **Diagnose (診断) > Technical Support (テクニカルサポート)** ページにある **Manage Technical Support Reports (テクニカルサポートレポートの管理)** をクリックします。

このページを使用すると、生成される TSR に含めるオプションを選択できたり、以前に生成されたレポートにアクセスしたりできます。トラフィックマネージャには、最後に生成された 10 の TSR が保存されており、それぞれをダウンロードまたは削除するオプションが用意されています。TSR を外部ロケーションにダウンロードし、

表示されるチェックボックスを使用してトラフィックマネージャからその **TSR** を削除することによって、ディスク空き容量を確保することができます。

また、コマンドラインを使用するか、**SOAP API** から **TSR** を生成することもできます。このオプションは、管理 **UI** へのアクセスが中断されているときに使用してください。

TSR をコマンドラインから生成するには、以下のスクリプトを実行します。

```
ZEUSHOME/zxtm/bin/support-report [--all-logs] <outfile>
```

オプションの `--all-logs` 引数を使用すると、デフォルトで設定された **10 MB** のログファイルサイズのダウンロード制限を回避することができます。これにより、サイズにかかわらず、ログセット全体がサポートレポートに含まれるようになります。

CHAPTER 34 用語集

ここでは、本書で使用されている用語について解説します。用語によっては、コンピュータ関連の文献において様々な意味で使用されているものもありますが、この用語集では、本書の範囲に限定して意味を定義しています。

アクション – アクションは、特定イベントが発生した結果としてイベントハンドラが実行されると呼び出されます。アクションには、ログファイルへの書き込み、電子メールメッセージの送信、SYSLOG メッセージまたは SNMP トラップの送信、カスタムアクションプログラムの実行などがあります。

管理サーバ – トラフィックマネージャファミリの管理インターフェース。Web ベースのユーザインターフェースからアクセスします。

監査ログ – トラフィックマネージャに対するユーザのアクティビティが記録されるトラフィックマネージャの監査ログ。Diagnose (診断) ページから表示することができます。

バックエンド IP アドレス – トラフィックマネージャがバックエンドサーバとの通信に使用する IP アドレス、またはバックエンドサーバの IP アドレス。

バックエンドサーバ – Web やメールサーバなどのサービスを実行するマシン。通常、バックエンドサーバはローカルネットワーク内に存在し、トラフィックマネージャが処理したリクエストを受け取ります。トラフィックマネージャでは、バックエンドサーバと指定ポートの組み合わせによってノードが形成されます。

帯域制御 – 特定サービスまたは特定タイプのリクエストに利用できるネットワーク帯域をモニタリングおよびコントロールする機能。

カタログ – サービスに適用可能なオブジェクトのセントラルリポジトリ。ルール、モニタ、サービスプロテクションクラス、および様々な SSL 項目のカタログがあります。

認証局 – SSL 証明書への署名を独自に行う広く認められた機関。

認証失効リスト(CRL) – 無効として扱うべきクライアント SSL 証明書のリスト。認証局が発行します。

証明書署名リクエスト(CSR) – 自己署名証明書を基に作成されるリクエスト。CSR は、認証局の署名を得るために認証局に送信することができます。

CIDR IP サブネット – Classless Inter-Domain Routing (CIDR) IP サブネットは、標準の 32 ビット IP アドレスと、ネットワークプレフィックスの使用ビット数に関する追加情報で構成されます。例えば、「10.13.1.48/22」という CIDR IP サブネットの場合、「/22」は最初の 22 ビットが固有の

ネットワークプレフィックスを識別するために使用され、残り 10 ビットによって特定のホストが識別されることを示します。または、トラフィックマネージャでは、サブネットマスク「10.13.1.48/255.255.255.0」または明示的に「10.13.1.」を使用して CIDR IP サブネットを指定することができます。

クライアント証明書 – クライアントが保有する SSL 証明書で、制限されたサイトへのアクセスをクライアントに許可するために使用されます。

クラスター – 同じ設定を共有するトラフィックマネージャのグループ。

configure スクリプト – トラフィックマネージャをインストールしてから起動するまでの間に実行する必要があるスクリプト。パスワードなどの基本的な設定、トラフィックマネージャをスタンドアロンで運用するか既存のクラスターに組み込むかの指定を処理します。

コネクション管理 – リモートクライアントと仮想サーバ間またはプールとそのノード間のコネクションを管理する設定。

クッキー – サーバからクライアントに与えられ、クライアントのファイルシステムまたはブラウザクライアントセッションに保存される小さいデータ項目。クライアントはこのデータを保存し、以降のリクエスト時にサーバに提供します。クッキーは、セッション維持マップなどのクライアントデータをトラッキングするために使用されます。

DoS 攻撃 – サービスの正当な利用を妨げる悪意のある試みで、一般にはネットワークへの過剰なデータの送信やマシン間のコネクションの妨害といった形を取ります。

Diagnose (診断) ページ – トラフィックマネージャ環境の障害の診断と解決に役立つ管理サーバのセクション。

無効化されたノード – プール内の使用されていないノード。このノードに対して、モニタリングやトラフィックの送信は行われません。ノードを無効化すると、ノードをプールから削除した場合と同じ状態になりますが、復旧を容易に行えます。

DDoS 攻撃 – マシン群による DoS 攻撃。多くの場合、これらのマシンはワームやウイルスに感染しています。

DMZ – ファイアウォールの非武装地帯。この地帯は内部ネットワークとインターネットとの間に位置し、多くの場合、ここに外部からアクセス可能なサーバが配置されます。

ドレイン中のノード – プール内の、新しいコネクションが送信されないノード。このノードに対する既存のコネクションおよびセッションが全て期限切れになった時点で、このノードをプールから安全に削除することができます。

エラーファイル - リクエストに対してレスポンスを返すことのできるノードがない場合にクライアントに送信されるファイル。エラーファイルはプール単位で設定されます。

イベント - IP アドレス転送やノードの障害または復旧など、特定の変更や事象が起これば、イベントが報告されます。全てのイベントはグローバルイベントログに記録されます。

イベントハンドラ - イベントハンドラは、関連付けられたイベントタイプのイベントが発生したときにトラフィックマネージャが実行するアクションを設定します。

イベントログ - トラフィックマネージャ内で発生したイベントが記録されるトラフィックマネージャのログ。重大度に応じて INFO (情報)、WARN (警告)、SERIOUS (重大)、FATAL (致命的)などに分類され、Diagnose (診断)ページから表示することができます。

イベントタイプ - 管理を容易にするためにグループ化されたイベントのセット。特定イベントタイプのイベントが発生した場合、トラフィックマネージャはそのイベントタイプに関連付けられたアクションを実行できます。

フェールオーバー - 障害が発生したマシンの負荷をクラスタ内の他のマシンが引き継ぐ動作。

障害プール - プール内の全てのノードで障害が発生した場合、リクエストは、そのプールの障害プールに送信されます(設定されている場合)。

フォールトレランス - サービスを中断することなく、1 台以上のサーバの障害に対処するためのクラスタまたはプールの機能。

フロントエンド IP アドレス - トラフィックマネージャの、外部からアクセス可能な固定 IP アドレス。

フロントエンドサーバ - トラフィックマネージャなど、インターネット側から通信可能なサーバ。

ヘルスマニタリング - 障害が発生したマシンにリクエストが送信されないようにするために、トラフィックマネージャがトラフィックマネージャおよびバックエンドのヘルスをトラッキングするプロセス。

IPv6 - 現在のプロトコルである IPv4 の後継として考えられているネットワークレイヤプロトコル。従来プロトコルよりアドレス範囲が広く、セキュリティが強化されているなどの特長があります。

Java Extension - Java Extensions により、TrafficScript の機能を強化(拡張)することができます。リクエストまたはレスポンスを処理するために、TrafficScript ルール(または RuleBuilder ルール)から Java Extension を呼び出すことができます。Java Run-time 環境では、TrafficScript のみ

を使用した場合よりはるかに強力なプログラミング言語とライブラリを利用できます。

負荷分散 – 複数のバックエンドサーバにリクエストを分配すること。これにより、全てのリクエストをできる限り迅速に処理することができます。

MD5 - 暗号化において、MD5 (Message-Digest algorithm 5)は広く利用されている暗号化ハッシュ関数で、128 ビットのハッシュ値を出力します。インターネット規格(RFC 1321)として様々なセキュリティアプリケーションに採用されており、ファイルの完全性チェックにも広く利用されています。

MIME タイプ - Multipart Internet Mail Extension (多目的インターネットメール拡張)。PDF ファイルの場合は「application/pdf」など、転送されるデータのタイプを識別する名前。この規格は当初電子メールアプリケーションに使用されていましたが、その後 Web サーバなど他のインターネット技術に広く採用されました。

モニター – トラフィックマネージャがプールのヘルスをトラッキングするために使用する定期的なテスト。

ネームサーバ – ネームサービスプロトコルを実装するサーバ。例えば、ドメインネームシステム(DNS)サーバは、ドメイン名 `www.example.com` を IP アドレス `208.70.196.59` に変換します。

NAT – ネットワークアドレス変換(NAT、ネットワークマスカレードとも呼ばれる)では、IP パケットがルータまたはファイアウォールを通過するときにその送信元または宛先アドレスの書き換えが行われます。NATを使用する多くのシステムでは、プライベートネットワーク上の複数のホストが単一のパブリック IP アドレスでインターネットにアクセスできるようにするためにこの機能が使用されます。

ノード – ポートが関連付けられたバックエンドサーバ。トラフィックマネージャから送信されたリクエストを受信し、レスポンスとしてリクエストされたコンテンツを返します。

PEM – Private Enhanced Mail。 .pem ファイル拡張子は、Base 64 で符号化された X.509 鍵および証明書に使用されます。これらは出力可能なテキストで、----- BEGIN CERTIFICATE -----形式の行から始まります。

PKCS#11 – セキュアなハードウェアとの通信に使用される規格。

プール – ノードの論理グループ。仮想サーバがリクエストをプールに割り当てると、プールはリクエストをノードに負荷分散します。

Real Time Streaming Protocol (RTSP) - TCP および UDP を使用してオーディオおよびビデオフォーマットなどのリアルタイムデータのオンデマンド配信を可能にするアプリケーションレベルプロトコル。

Real-time Transport Protocol (RTP) - オーディオおよびビデオファイルを配信するための UDP パケットフォーマット。ストリーミングデータを配信するために、RTSP とともに広く使用されています。

リクエストルール - トラフィックマネージャでどのようにリクエストを処理するかをコントロールする **TrafficScript** ルールまたは **RuleBuilder** ルール。ルールは仮想サーバから呼び出され、リクエストをチェック、操作、書き換え、およびルーティングすることができます。また、リクエストを処理するためにトラフィックマネージャの他の機能をどのように使用するかもコントロールできます。

レスポンスルール - レスポンスルールはリクエストルールと対称をなすもので、トラフィックマネージャでどのようにレスポンスを処理するかをコントロールします。

RuleBuilder - ルールを作成するためのビジュアルエディタ。リクエストのチェック条件と、その条件の一部または全てが満たされた場合のアクションを選択することができます。

自己署名 SSL 証明書 - 認証局ではなく所有者によって署名された SSL 証明書。セキュリティテストには役立ちますが、商用サイトでは使用しないでください。認証局の署名のある証明書にするために、証明書署名リクエストを送信することができます。

サーバ証明書 - SSL サービス(Web サイトなど)を識別するために使用される SSL 証明書で、暗号化されたトランザクションを確立するために必要な公開鍵が含まれます。

サービス - 一般的なインターネットサービスとして、Web、メール、DNS、.NET や SOAP などのアプリケーションがあります。IP または DNS アドレスとポートの組み合わせとして、パブリックまたはローカルに提供されます。

サービスレベルモニタリング - バックエンドサーバのレスポンス時間をモニタリングし、それを適合値と比較して、利用可能なリソースを動的に調整するなど変化に対処する機能。

サービスプロテクション - インターネットサービスを DoS(サービス妨害)や DDoS(分散型サービス妨害)などの攻撃から保護すること。

サービスプロテクションクラス - ユーザが定義するサービスプロテクション設定のグループ。このクラスはサービスプロテクションカタログに保持され、任意の仮想サーバに適用できます。

Session Initiation Protocol (SIP) - カンファレンス、電話、プレゼンス、イベント、およびインスタントメッセージングのためにインターネットで 사용되는プロトコル。

セッション維持 - 同じユーザセッションに属するリクエストを常に同じノードに送信するためのプロセス。セッションは、IP アドレス、クッキー、SSL セッション ID など複数の方法で識別できます。

SNMP - Simple Network Management Protocol (簡易ネットワーク管理プロトコル)。ネットワークのモニタリングおよび管理に関するオープン規格。SNMP により、ネットワーク上のデバイスをモニタリングし、性能データを収集することができます。

SOAP - Simple Object Access Protocol。Web サービスメッセージのための XML ベースの規格。

SSL - Secure Sockets Layer。暗号化されたネットワーク通信を実現するプロトコル。リクエストはトラフィックマネージャ内で復号化し、再度暗号化することができます。

Tar アーカイブ - tar ファイルフォーマットのファイル。アーカイブファイルフォーマットの標準タイプ。ユーザやグループの許可、日付、ディレクトリ構造などのファイルシステム情報を維持しながら、ファイルをアーカイブおよびアーカイブ解除するために広く使用されています。

トラフィック IP アドレス - インターネットサービスを公開するために使用する IP アドレスで、通常はパブリック DNS エントリに紐付けられます。

トラフィック IP グループ - 複数のトラフィック IP アドレスから成る論理グループで、一部または全てのトラフィックマネージャにまたがります。これらのトラフィックマネージャは、グループ内の IP アドレスを常時使用可能にするためのネゴシエーションを行います。

トラフィックマネージャ - トラフィックマネージャを実行し、サービスを管理するフロントエンドサーバ。

トラフィッククラスタ - 無制限数のアクティブおよびパッシブトラフィックマネージャを配備する機能。これにより、複数の障害が発生しても可用性を確保し、必要に応じてトラフィックマネージャクラスタを拡張することができます。N+M 冗長性とも言います。

TrafficScript - トラフィックマネージャの一連のソリューションで使用するルールを作成するためのスクリプティング言語。

誘導手段 - DoS 攻撃や DDoS 攻撃を行うための手段。

仮想サーバ - トラフィックマネージャとインターネット間のインターフェース。仮想サーバはいずれかのサービスを管理します。インターネットからリ

クエストを受信し、場合によっては SSL 復号化を行い、その後ルールを適用してプールに割り当てます。

XML – Extensible Markup Language. 構造化データの保存および交換に使用されます。

XPath – XML ドキュメントの各部分のアドレス指定を行うための言語。**TrafficScript** ルールと組み合わせて使用することにより、ドキュメントのコンテンツに基づいて決定を行うことができます。

ZEUSHOME – トラフィックマネージャのインストール場所。デフォルトでは、トラフィックマネージャソフトウェア製品の場合は `/usr/local/zeus`、アプリケーションとクラウドインスタンスの場合は `/opt/zeus` が使用されます。

CHAPTER 35 ソフトウェア使用許諾契約書

Berkeley DB コード(バージョン 1.85)の使用許諾条件

以下の使用許諾条件は、トラフィックマネージャが紐付けられている libdb コードに関するものです。このコードのソースは以下のサイトで参照できます。

<http://www.sleepycat.com/update/snapshot/db.1.85.tar.gz>

Copyright (c) 1991, 1993 The Regents of the University of California。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。

本製品には、カリフォルニア大学バークレー校およびその貢献者により開発されたソフトウェアが含まれています。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、REGENTS および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、REGENTS または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

RSA PKCS11

本ソフトウェアのコピーまたは使用は、本ソフトウェアまたは本機能に言及する全てのものに「RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)」と明記することを条件に許諾されます。

また、派生物の作成および使用についても、派生物に言及する全てのものに「derived from the RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)」と明記することを条件に許諾されます。

OpenLDAP コード、バージョン 2.4.23 の使用許諾条件

以下の使用許諾条件は、トラフィックマネージャが紐付けられている libldap コードに関するものです。このコードのソースは以下のサイトで参照できます。

<ftp://ftp.openldap.org/pub/OpenLDAP/openldap-stable/openldap-stable-20100719.tgz>

The OpenLDAP Public License

バージョン 2.8、2003 年 8 月 17 日

本ソフトウェアおよび付随するドキュメント(以下「ソフトウェア」)の再配布および使用は、変更の有無を問わず、以下の条件が満たされていることを条件に許諾されます。

1. ソース形式での再配布物には、著作権の宣言文および情報を含める必要があります。
2. バイナリ形式での再配布物には、該当する著作権の宣言文および情報、この条件リスト、および以下の免責事項を再配布物とともに提供されるドキュメントおよび/またはその他資料に複製する必要があります。
3. 再配布物には、一切の変更を加えずに本ドキュメントのコピーを含める必要があります。

OpenLDAP Foundation は、本使用許諾条件を改訂することがあります。各改訂は、バージョン番号によって管理されます。本ソフトウェアは、使用許諾条件の本改訂または以降の改訂の条件に従って使用することができます。

本ソフトウェアは、OPENLDAP FOUNDATION および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、OPENLDAP FOUNDATION、その貢献者、あるいはソフトウェアの作成者または所有者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使

用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。また、このような損傷の可能性を知らされていた場合も同様です。

事前に書面による明確な許諾を得ない限り、広告内、あるいは本ソフトウェアの販売、使用、またはその他取引を促進する目的で、作成者および著作権所有者の名称を使用することはできません。本ソフトウェアの著作権は、常に著作権所有者が留保します。

OpenLDAP は、OpenLDAP Foundation の登録商標です。

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA.
著作権所有。本ドキュメントをコピーし、一切の変更を加えずに本ドキュメントのコピーを配布することを許可します。

PCRE2 の使用許諾条件

以下の使用許諾条件は、トラフィックマネージャが紐付けられている `pcre2` コードに関するものです。

バージョン: 10.21

ライセンス概要: BSD

PCRE2 とは、正規表現をサポートするための関数のライブラリです。その構文とセマンティックは、Perl 5 言語の正規表現に可能な限り近づけられています。

PCRE2 のリリース 10 は、以下に規定するように、「BSD」ライセンスの条件に基づいて配布されています。「doc」ディクショナリで提供される PCRE2 のドキュメントは、ソフトウェア自体と同じ条件に基づいて配布されます。`testdata directory` ディレクトリ内のデータは著作権保護されず、パブリックドメインに属します。

基本ライブラリ関数は C で記述され、独立したものとなっています。配布ファイルには、パターンマッチングを最適化するために使用できる JIT(Just-In-Time)コンパイラも含まれています。これは、ライブラリの作成時に省略できるオプションの機能です。

基本ライブラリ関数

作成者: Philip Hazel

電子メールのローカル部: ph10

電子メールのドメイン: `cam.ac.uk`

ケンブリッジ大学コンピューティングサービス学科
イングランド、ケンブリッジ

Copyright (c) 1997-2016 University of Cambridge。

著作権所有。

PCRE2 JIT コンパイルのサポート

作成者: Zoltan Herczeg

電子メールのローカル部: `hzmester`

電子メールのドメイン: `freemail.hu`

Copyright(c) 2010-2016 Zoltan Herczeg

著作権所有。

スタックレス JIT コンパイラ

作成者: Zoltan Herczeg

電子メールのローカル部: `hzmester`

電子メールのドメイン: `freemail.hu`

Copyright(c) 2009-2016 Zoltan Herczeg

著作権所有。

「BSD」ライセンス

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

* ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

* バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、ケンブリッジ大学の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Libnet の使用許諾条件

以下の使用許諾条件は、トラフィックマネージャが紐付けられている libnet コードに関するものです(Solaris バージョンのみ)。

libnet 1.1.x

Copyright (c) 1998 - 2002 Mike D. Schiffman <mike@infonexus.com>

<http://www.packetfactory.net/libnet>

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアは、作成者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、デー

タ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Yahoo! UI Library の使用許諾条件

Copyright (c) 2009, Yahoo! Inc. 著作権所有。

ソースおよびバイナリ形式での本ソフトウェアの再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による Yahoo! Inc.の明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、Yahoo! Inc.の名称およびその貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

ssleay 暗号化ライブラリの使用許諾条件

以下の使用許諾条件は、Brocade 仮想トラフィックマネージャで使用されている SHA-1 および MD5 アルゴリズムの実装に関するものです。

Copyright (C) 1997 Eric Young (ey@cryptsoft.com)。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。「本製品には、Eric Young (eay@cryptsoft.com)により作成された暗号化ソフトウェアが含まれています。使用するライブラリのルーチンが暗号化に関連するものでない場合は、「暗号化」という語を省略することができます。
4. apps ディレクトリ(アプリケーションコード)にある Windows 固有のコード(またはその派生物)を含める場合、次の情報を含める必要があります:「本製品には、Tim Hudson (tjh@cryptsoft.com)により作成されたソフトウェアが含まれています。」

本ソフトウェアは、ERIC YOUNG により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

libxml2 および libxslt の使用許諾条件

以下の使用許諾条件は、MIT License による許諾のもと Brocade 仮想トラフィックマネージャで使用されている XML パーサおよびツールキットライブラリに関するものです。

libxml2 2.9.1

libxslt 1.1.28

Copyright (c) 1998-2012 Daniel Veillard

本ライセンスにより、
以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、
配布、サブライセンス、
および/または販売する無制限の権利を含め、
ソフトウェアを自由に扱うことが無償で許諾されます。
また、ソフトウェアを提供された人についても
同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの
全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め
(ただしこれらに限定されない)、
明示であると暗黙であるとを問わず、
何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、
作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、
あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、
またはその他責任に対し、
契約の行為、不法行為、またはその他の有無を問わず、
一切責任を負いません。

Java Servlet API の使用許諾条件

Brocade 仮想トラフィックマネージャで使用されている Java Servlet API には、
Apache Jakarta Project のコードが含まれており、ASF ライセンスバージョン 2.0 に
より使用が許諾されています。

Copyright 1999-2007, The Apache Software Foundation

Apache License バージョン 2.0 (以下「License」)のもと使用が許諾されています。
お客様は、本ファイルを使用するときは、必ず以下の License に従う必要があります。
License のコピーは以下のサイトから入手できます。

<http://www.apache.org/licenses/LICENSE-2.0>

適用法で義務付けられているか書面により合意がない限り、License のもと配布され
るソフトウェアは、明示であると暗黙であるとを問わず、何ら保証または条件のな
い状態で「現状のまま」配布されます。License において許諾または制限を規制する
特定言語については、License を参照してください。

expat XML パーサの使用許諾条件

以下の使用許諾条件は、Brocade 仮想トラフィックマネージャで使用されている expat XML パーサに関するものです。

expat 2.1.0

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
および Clark Cooper

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。

また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

MooTools の使用許諾条件

Brocade 仮想トラフィックマネージャには、オープンソース MIT License による許諾のもと MooTools JavaScript フレームワークのコードが含まれています。

MooTools 1.2

Copyright (c) 2006-2009 Valerio Proietti

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/

または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

OpenLayers の使用許諾条件

Brocade 仮想トラフィックマネージャには、MetaCarta、Rico、および XMLHttpRequest.js のコードが含まれています。以下の著作権情報が適用されます。

Copyright (c) 2005-2008 MetaCarta, Inc.

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に(以下の免責事項に記載された制限に従って)許諾されます。

- ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
- バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
- 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、MetaCarta, Inc.の名称および貢献者の氏名を使用することはできません。

本使用許諾では、明示であると暗黙であるとを問わず、いかなる第三者の著作権に対する使用許諾も与えられません。本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Rico の一部を含む<<http://openrico.org/>>

Copyright 2005 Sabre Airline Solutions

Apache License バージョン 2.0 (以下「License」)のもと使用が許諾されています。お客様は、本ファイルを使用するときは、必ず以下の License に従う必要があります。License のコピーは以下のサイトから入手できます。

<http://www.apache.org/licenses/LICENSE-2.0>

適用法で義務付けられているか書面により合意がない限り、License のもと配布されるソフトウェアは、明示であると暗黙であるとを問わず、何ら保証または条件のない状態で「現状のまま」配布されます。License において許諾または制限を規制する特定言語については、License を参照してください。

XMLHttpRequest.js を含む<<http://code.google.com/p/xmlhttprequest/>>

Copyright 2007 Sergey Ilinsky (<http://www.ilinsky.com>)

Apache License バージョン 2.0 (以下「License」)のもと使用が許諾されています。お客様は、本ファイルを使用するときは、必ず以下の License に従う必要があります。License のコピーは以下のサイトから入手できます。

<http://www.apache.org/licenses/LICENSE-2.0>

rsync の使用許諾条件

Brocade 仮想トラフィックマネージャには、高速リモートファイルコピープログラムである rsync のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 2.6.9

ライセンス概要: GPL v2

著作権:

rsync は、当初 Andrew Tridgell により作成され、世界中の多くの開発者により改善が行われてきました。rsync は、以下のサイトで公開されている GNU General Public License の条項に従って使用、変更、および再配布することができます。

<http://www.fsf.org/licenses/old-licenses/gpl-2.0.html>

mod_imap.c の使用許諾条件

Brocade 仮想トラフィックマネージャには、mod_imap.c のコードが含まれています。mod_imap.c は、Apache HTTP Server (<http://httpd.apache.org>)によって作成され、Apache Software License バージョン 1.1 のもと使用が許諾されています。以下の著作権情報が適用されます。

Apache Software License バージョン 1.1

Copyright (c) 2000 The Apache Software Foundation。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 再配布物にエンドユーザ向けドキュメントを含める場合は、以下の情報を含める必要があります。

「本製品には、Apache Software Foundation (<http://www.apache.org/>)により開発されたソフトウェアが含まれています。」

または、ソフトウェア自体の中でこのようなサードパーティ情報が通常表示される箇所にこの情報を表示することもできます。

4. 事前に書面による許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、「Apache」および「Apache Software Foundation」の名称を使用することはできません。書面による許諾については、apache@apache.org にお問い合わせください。

5. 事前に書面による Apache Software Foundation の許諾を得ない限り、本ソフトウェアから派生した製品の名称を「Apache」とすることはできません。また、その製品名に「Apache」を含めることもできません。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、APACHE SOFTWARE FOUNDATION またはその貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Antlr および libantlr の使用許諾条件

Brocade 仮想トラフィックマネージャには、JavaScript のレキサジェネレータおよびランタイムライブラリである Antlr および libantlr のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 3.4

ライセンス概要: BSD

著作権:

Copyright (c) 2005-2009 Jim Idle, Temporal Wave LLC

<http://www.temporal-wave.com>

<http://www.linkedin.com/in/jimidle>

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、作成者の名称を使用することはできません。

本ソフトウェアは、作成者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

es3-grammar の使用許諾条件

Brocade 仮想トラフィックマネージャには、es3-grammar を使用して生成されたコードが含まれています。es3-grammar は、ANTLR 3 用の完全な ECMAScript 3 grammar です。以下の著作権情報が適用されます。

バージョン: N/A

ライセンス概要: BSD

著作権:

Copyright (c) 2008-2010, Xebic Research B.V. All rights reserved.

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

3. 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、作成者の名称を使用することはできません。

本ソフトウェアは、作成者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

jsoncpp の使用許諾条件

Brocade 仮想トラフィックマネージャには、設定ファイルの解析に使用される JSON ライブラリである jsoncpp のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 0.5.0

ライセンス概要: MIT

著作権:

Copyright (c) 2007-2010 Baptiste Lepilleur

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

libjpeg の使用許諾条件

Brocade 仮想トラフィックマネージャには、JPEG 操作ライブラリである libjpeg のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 8c

ライセンス概要: IJG ライセンス

著作権:

本ソフトウェア、その品質、正確性、商品性、または特定目的への適合性に関して、明示であると暗黙であることを問わず作成者は何ら保証も明言もしません。本ソフトウェアは、「現状のまま」提供されるものであり、その品質と正確性に関する全てのリスクは購入者とその利用者が負うものとします。

本ソフトウェアの著作権: Copyright (C) 1991-2011, Thomas G. Lane, Guido Vollbeding. All Rights Reserved (以下の場合を除く)

本ソフトウェア(またはその一部)を任意の目的に無償で使用、コピー、変更、配布する権利が、以下の条件に従って許諾されます。

(1)本ソフトウェアのソースコードのいかなる部分を配布する場合でも、著作権情報および無保証の但し書きとともに本 README ファイルを含める必要があります。また、元のファイルに対する追加、削除、変更がある場合には、付属文書にてそれを明記しなければなりません。

(2)実行可能コードのみを配布する場合には、付属文書に「this software is based in part on the work of the Independent JPEG Group (本ソフトウェアには、Independent JPEG Group で作成されたコードの一部が利用されています)」と記載する必要があります。

(3)本ソフトウェアの使用許諾は、望ましくない結果に対する責任を全てユーザが負う場合に限って付与されます。作成者はかかる損害に対して一切の責任を負いません。

以上の条件は、変更されないライブラリのみならず、IJG コードから派生した、または IJG コードに基づく全てのソフトウェアに適用されます。弊社の作成物を使用する場合は、ご連絡いただく必要があります。

IJG の作成者名または会社名を、本ソフトウェアまたはそれに派生する製品の広告宣伝に使用することは一切許可されません。本ソフトウェアは、「the Independent JPEG Group's software (Independent JPEG Group のソフトウェア)」と称する必要があります。

本ソフトウェアは、全ての保証または賠償請求を製品ベンダーが負う限りにおいて、商用製品の基盤として使用することを許諾または推奨されるものです。

ansi2knr.c は、その著作権所有者 Aladdin Enterprises (Menlo Park, CA) の個人経営者である L. Peter Deutsch の許可を受けて本ディストリビューションに含まれています。ansi2knr.c は上記の著作権および条件の対象にはならず、Free Software Foundation の通常の配布条件に従います。主として、再配布する場合にはソースコードを含める必要があります(詳細についてはファイル ansi2knr.c を参照)。ただし、ansi2knr.c は IJG コードから生成されるプログラムの一部として含める必要はなく、後続の段落以上に利用者を制限するものではありません。

UNIX 設定スクリプト「configure」は、GNU Autoconf により生成されました。その著作権は Free Software Foundation が保有していますが、自由に配布できます。サポートスクリプト(config.guess、config.sub、ltmain.sh)についても同様です。もうひとつのスクリプト、install-sh の著作権は X Consortium が保有していますが、これも自由に配布が可能です。

IJG のディストリビューションには、GIF ファイルを読み書きするコードが正式に含まれていました。Unisys LZW 特許の侵害を回避するために、GIF の読み取りサポートは完全に除外され、GIF ライタは非圧縮 GIF を生成するように簡略化されています。この手法で、LZW アルゴリズムは使用しません。生成される GIF ファイルは通常より大きくなりますが、全ての標準 GIF デコーダで読み取ることができます。

以下の記述が必要です。

「The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated.」
(Graphics Interchange Format(c)は、CompuServe Incorporated の著作物です。
GIF(sm)は CompuServe Incorporated のサービスマークです。)

libunwind の使用許諾条件

Brocade 仮想トラフィックマネージャには、スタックのアンワインドライブラリである libunwind のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 1.0.1

ライセンス概要: MIT

著作権:

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

Perl JSON ライブラリの使用許諾条件

Brocade 仮想トラフィックマネージャには、JSON (JavaScript Object Notation) 暗号化および復号化用の Perl モジュールである Perl JSON ライブラリのコードが含まれています。以下の著作権情報が適用されます。

バージョン: 2.53

ライセンス概要: Perl Artistic License v.1

著作権:

前文

本書は、パッケージの使用者に事実上の慣習的方法においてパッケージを使用および配布する権利、および合理的変更を行う権利を付与しながらも、著作権所有者がパッケージ開発に関する一定の芸術的管理権を維持するように、パッケージの複製が許容される条件を示すことを目的としています。

定義:

「パッケージ」とは、著作権所有者が配布するファイルの集まり、および当該ファイルの集まりに対してテキストの変更を行うことで作成された派生物を指します。

「標準バージョン」とは、変更されていないパッケージ、または以下に規定する著作権所有者の意思に従って変更されたパッケージを指します。

「著作権所有者」とは、パッケージの著作権で指名された全ての人物を指します。

「お客様」とは、あなたが本パッケージを複製または配布しようとしている場合、あなた自身を指します。

「合理的複製料」とは、メディア費用、複製費用、関係者の拘束時間などに基づいてお客様が正当化できる料金を指します。(お客様は、著作権所有者に対してではなく、当該料金を負担する市場としての一般的コンピュータ関係社会に対してのみ当該複製料を正当化するよう求められます。)

「自由に利用可能」とは、商品自体に対して料金が課されないことを指しますが、商品の取り扱いに関する料金が課されることがあります。また、商品の受領者は、当該商品を受領したときと同じ条件に基づいて当該商品を再配布できます。

1. お客様は、元の著作権情報および関連する免責事項の全てを複製する限りにおいて、本パッケージの標準バージョンのソース形式の逐語的複製を無制限に作成および配布することができます。

2. お客様は、バグの修正、移植性の修正、およびパブリックドメインまたは著作権所有者に由来するその他の変更を加えることができます。そのような方法において変更されたパッケージは、標準バージョンと見なされるものとします。

3. いつどのような変更をファイルに対して行ったかを記述した明示的な情報を変更された各ファイルに挿入し、かつ以下の少なくとも 1 つに該当する限り、本パッケージの複写を、形を問わず別の方法で変更することができます。

a) お客様が行った変更事項をパブリックドメインに掲載する、またはそれらの変更事項を自由に利用可能な状態にする。これには、Usenet または同等の媒体へ前述の変更事項を掲示する、uunet.uu.net などの大手アーカイブサイトへ変更事項を掲載する、お客様が行った変更事項を著作権所有者が本パッケージの標準バージョンに取り込むことを許可するなどの方法が該当します。

b) 変更したパッケージをお客様の会社または組織内においてのみ使用する。

c) 標準の実行ファイルと名前が競合しないように、非標準の全ての実行ファイルの名前を変更し、標準の実行ファイルと併せて提供する。また、非標準の実行ファイルごとに、標準バージョンとの差異を明確に記した個別のマニュアルページを提供する。

d) 著作権所有者と配布に関するその他の取り決めを行う。

4. 以下の少なくとも 1 つに該当する限り、お客様は本パッケージのプログラムをオブジェクトコードまたは実行ファイルの形式で配布することができます。

a) 実行ファイルおよびライブラリファイルの標準バージョンを、(マニュアルページまたはそれと同等のものにおいて)使用法とともに標準バージョンを入手できる場所において配布する。

b) コンピュータが解読できるパッケージのソースにお客様が行った変更事項を加えて配布する。

c) 非標準の実行ファイルに非標準の名前を付け、標準バージョンを入手できる場所において、その差異を使用法とともにマニュアルページ(または同等のもの)に明確に記す。

d) 著作権所有者と配布に関するその他の取り決めを行う。

5. お客様は、本パッケージの任意の配布に対して合理的複製料を課すことができます。お客様は、本パッケージのサポートに対して任意の料金を課すことができます。お客様は、本パッケージ自体に対する料金を課すことはできません。ただし、お客様が本パッケージをお客様自身の製品として広告しない限り、より大規模な(できる限り商用の)ソフトウェアの配布の一環として、その他の(できる限り商用の)プログラムと併せて本パッケージを配布することができます。お客様は、本パッケージのインタプリタを実行ファイルの内部に(リンク付けによって)埋め込むことができます。インタプリタの完全に標準のバージョンが同様に埋め込まれている場合、アグリゲーションに似せた形式で解釈される必要があります。

6. 本パッケージへのインプットとして供給される、または本パッケージからのアウトプットとして生成されるスクリプトおよびライブラリファイルは、自動的に本パッケージの著作権の影響を受けることはなく、それらを生成した人物に属するものとし、それらを商業的に販売したり、本パッケージと集約したりすることができます。そのようなスクリプトまたはライブラリファイルが、バイナリ実行可能イメージを生成するいわゆる「undump」または「unexec」の方法で本パッケージに集約される場合、お客様がそのような実行可能イメージを本パッケージの標準バージョンであると明言しない限り、そのようなイメージの配布が本パッケージの配布と解釈されたり、第3および第4段落の制限が適用されたりすることはないものとします。

7. 本パッケージで定義されている言語のサブルーチンおよびバリアブルをエミュレートするために、お客様によって供給され、本パッケージにリンクされるCサブルーチン(またはその他の言語で同等にコンパイルされたサブルーチン)は、当該ルーチンが、当該言語に対する回帰テストが失敗に終わる原因となるような方法で当該言語を変更しない限り、本パッケージの一部と見なされることはなく、第6段落のインプットに相当します。

8. 本パッケージの使用が埋め込まれている場合、すなわち、本パッケージのインターフェースが商業的配布のエンドユーザに見えるようにする公然たる試みが一切行われない限り、商業的配布を伴う本パッケージのアグリゲーションは常に認められます。そのような使用は、本パッケージの配布とは解釈されません。

9. 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、著作権所有者の名称を使用することはできません。

10. 本パッケージは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。

OpenSSL の使用許諾条件

Brocade 仮想トラフィックマネージャには、暗号化および SSL/TLS ツールキットである OpenSSL ライブラリのコードが含まれています。以下の著作権情報が適用されます。

バージョン: 1.0.2g

ライセンス概要: BSD スタイル

著作権情報:

ライセンス交付

OpenSSL ツールキットはデュアルライセンスを受けています。つまり、OpenSSL ライセンスの条件と元の SSLeay ライセンスがツールキットに適用されます。実際のライセンス文については、以下を参照してください。実際に、両ライセンスとも BSD スタイルのオープンソースライセンスです。OpenSSL に関するライセンス交付については、openssl-core@openssl.org にお問い合わせください。

OpenSSL ライセンス

Copyright (c) 1998-2016 The OpenSSL Project。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。「本製品には、OpenSSL Project により開発された OpenSSL ツールキットを使用するためのソフトウェアが含まれています。
(<http://www.openssl.org/>)」
4. 4. 事前に書面による許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、「OpenSSL ツールキット」および「OpenSSL Project」の名称を使用することはできません。書面による許諾については、openssl-core@openssl.org にお問い合わせください。
5. 事前に書面による OpenSSL Project の許諾を得ない限り、本ソフトウェアから派生した製品の名称を「OpenSSL」とすることはできません。また、その製品名に「OpenSSL」を含めることもできません。

6. いかなる形式の再配布も、以下の情報を表記する必要があります。「本製品には、OpenSSL Project により開発された OpenSSL ツールキットを使用するためのソフトウェアが含まれています。(http://www.openssl.org/)」

本ソフトウェアは、OpenSSL Project により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、OpenSSL Project またはその貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、

その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

「本製品には、Eric Young (eay@cryptsoft.com)により作成された暗号化ソフトウェアが含まれています。本製品には、Tim Hudson (tjh@cryptsoft.com)により作成されたソフトウェアが含まれています。

オリジナル SSLeay ライセンス

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)。著作権所有。

本パッケージは、Eric Young (eay@cryptsoft.com)により作成された SSL 実装です。この実装は、Netscape SSL に適合するように作成されています。

以下の条件が順守される限り、本ライブラリは営利使用および非営利使用において無料です。以下の条件は、本配布に使用される全てのコード(SSL コードだけでなく、RC4、RSA、lhash、DES などのコード)に適用されます。本配布に含まれる SSL ドキュメントは、著作権所有者が Tim Hudson (tjh@cryptsoft.com)である場合を除き、同様の著作権条件が適用されます。

著作権は Eric Young が保有するため、コード内のいかなる著作権情報も削除してはなりません。本パッケージが製品内で使用される場合、使用されるライブラリの一部の作成者としての属性が Eric Young に付与されます。これは、プログラム開始処理のテキストメッセージの形式、またはパッケージに付属するドキュメント内(オンラインまたはテキスト)に記します。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

3. 本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。「本製品には、Eric Young (eay@cryptsoft.com)により作成された暗号化ソフトウェアが含まれています。」使用するライブラリのルーチンが暗号化に関連するものでない場合は、「暗号化」という語を省略することができます。

4. apps ディレクトリ(アプリケーションコード)にある Windows 固有のコード(またはその派生物)を含める場合、次の情報を含める必要があります:「本製品には、Tim Hudson (tjh@cryptsoft.com)により作成されたソフトウェアが含まれています。」

本ソフトウェアは、ERIC YOUNG により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

パブリックに利用可能なバージョンまたは本コードの派生物に対するライセンスおよび配布条件を変更することはできません。すなわち、本コードを単純に複製し、別の配布ライセンス(GNU 公衆ライセンス)の適用を受けることはできません。

WebP の使用許諾条件

Brocade 仮想トラフィックマネージャには、イメージフォーマットである WebP のコードが対応するコンバージョンライブラリとともに含まれています。以下の著作権情報が適用されます。

バージョン: 0.3.1

ライセンス概要: BSD スタイル

著作権情報:

Copyright (c) 2010, Google Inc.著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

* ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

* バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

* 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、Google の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Flex の使用許諾条件

Brocade 仮想トラフィックマネージャには、高速な字句解析ツールである Flex により生成されるコードが含まれています。以下の著作権情報が適用されます。

バージョン: 2.5.37

ライセンス概要: BSD スタイル

著作権情報:

Flex は、エネルギー省との契約に基づき運営される Lawrence Berkeley (Livermore ではありません) Laboratory により考案されたため若干変更されている、BSD ソフトウェアに使用される著作権を保有します。

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006, 2007 The Flex Project.

Copyright (c) 1990, 1997 The Regents of the University of California.

著作権所有。

本コードは、Vern Paxson によりバークレー校に寄与されたソフトウェアの派生物です。

アメリカ合衆国政府は、アメリカ合衆国エネルギー省およびカリフォルニア大学との間で締結された契約(No. DE-AC03-76SF00098)に準じて本著作物の権利を保有します。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。

これは、基本的には「この情報を削除したり、大学(または flex の作成者)の名前を悪用したりすることを除けば、本ソフトウェアを使用して何を行ってもよい」ことを意味します。

「flex.sk1」スキャナスケルトンには著作権情報がないことに注意してください。flex を使用して生成されるスキャナを使用して行うことに制限はありません。それらの行為に対しては、上記の著作権に制約されることもありません。

CryptoJS の使用許諾条件

Brocade 仮想トラフィックマネージャには、JavaScript に実装される標準的かつセキュアな暗号化アルゴリズムを集めた CryptoJS のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 3.1.2

ライセンス概要: BSD

著作権情報:

Copyright (c) 2009-2013 Jeff Mott

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

zlib の使用許諾条件

Brocade 仮想トラフィックマネージャには、圧縮ライブラリである **zlib** のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 1.2.8

著作権情報:

zlib.h -- 「zlib」汎用圧縮ライブラリのインターフェース、バージョン 1.2.8、2013 年 4 月 28 日

Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler

本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合にも、作成者は、本ソフトウェアの使用に起因する損傷に責任を負いません。

以下の制限を条件として、商用アプリケーションを含め任意の目的に本ソフトウェアを使用し、自由に修正および再配布する許可があらゆる者に与えられます。

1. 本ソフトウェアの元の作成者について偽りの表記をしてはならず、元のソフトウェアを自ら開発したと主張してはなりません。製品内で本ソフトウェアを使用する場合、製品ドキュメントでその旨を記載することが望まれますが、必須ではありません。
2. 修正したソースバージョンは、その旨を明記しなければならず、元のソフトウェアとして偽って表記してはなりません。
3. 本通知は、配布するソースから削除し、または修正することはできません。

Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

zlib.js の使用許諾条件

Brocade 仮想トラフィックマネージャには、JavaScript Zlib ライブラリである zlib.js のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 0.2.0

ライセンス概要: MIT

著作権情報:

Copyright (c) 2012 imaya

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

XML::Twig の使用許諾条件

Brocade 仮想トラフィックマネージャには、XML データの処理に使用される Perl ライブラリである XML::Twig のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 3.48

ライセンス概要: Perl 5 License (Artistic 1 & GPL 1)

著作権情報:

著作権

Copyright (c) 1999-2012, Michel Rodriguez. 著作権所有。

本ライブラリは無料ソフトウェアであり、Perl 自体と同じ条件に基づいて再配布および/または変更することができます。

MIT Kerberos の使用許諾条件

Brocade 仮想トラフィックマネージャには、ネットワーク認証プロトコルである MIT Kerberos のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 1.12.1

ライセンス概要: MIT

著作権情報:

Copyright © 1985-2014 by the Massachusetts Institute of Technology.

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

本ソフトウェアのダウンロードは、合衆国輸出規則(EAR) 15 CFR 730-774 が課されるアメリカ合衆国からの暗号化ソフトウェアの輸出になることがあります。その他の法令が適用されることもあります。必要とされるあらゆるライセンスをアメリカ合衆国政府から取得することも含め、適用される全ての輸出法令を順守する責任は、輸出しようとする個人または団体が負います。

アメリカ合衆国政府は、キューバ、イラン、北朝鮮、スーダン、シリア、およびこれらの諸国の住民および国民を含め(ただしこれらに限定されない)、一部の国および個人に対する暗号化ソースコードの輸出を禁止しています。

本ソフトウェア配布のドキュメント要素は、クリエイティブコモンズ表示-継承 3.0 非移植ライセンスを受けています。(<http://creativecommons.org/licenses/by-sa/3.0/>)

個々のソースコードファイルは、MIT、Cygnus Support、Novell、OpenVision Technologies、Oracle、Red Hat、Sun Microsystems、FundsXpress およびその他の著作権で保護されています。

Project Athena、Athena、Athena MUSE、Discuss、Hesiod、Kerberos、Moirá、および Zephyr は、マサチューセッツ工科大学(MIT)の商標です。事前に書面による MIT の許諾を得ない限り、これらの商標を営利使用することはできません。

「営利使用」とは、名前を製品やその他の営利目的の方法で使用することを指します。これは、営利目的の会社が情報の伝達を目的として MIT の商標に言及することを妨げるものではありません(ただし、そのような場合でもそれが商標であるという認識を得なければなりません)。

以下の著作権および許可情報は、kadmin/create、kadmin/dbutil、kadmin/passwd、kadmin/server、lib/kadm5、および lib/rpc の一部にある OpenVision Kerberos 管理システムに適用されます。

Copyright, OpenVision Technologies, Inc., 1993-1996。著作権所有。

警告: 後述のとおり、OpenVision Kerberos 管理システムのソースコードを取り込むことは、以下の条件が受諾されたことを意味します。以下の条件に同意されない場合は、OpenVision Kerberos 管理システムを取り込まないでください。

ソースコードおよびそこからコンパイルされたオブジェクトコードを、変更の有無を問わず自由に使用および配布できますが、このソースコードは「現状のまま」提供されるものであり、商品性および特定目的への適合性のいかなる保証(ただしこれらに限定されない)またはその他の保証を含め、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、OpenVision は本ソースコードの使用、ソースコードの実行失敗、またはその他のいかなる理由を含め(ただしこれらに限定されない)、本合意により生じた利益の損失、データの喪失、代替品またはサービスの調達費用に対し、またはいかなる例外的、間接的、または必然的損傷に対し、一切責任を負いません。

OpenVision は、提供されたソースコードに対するあらゆる著作権を保有します。また OpenVision は、OpenVision またはサードパーティのいずれによって作成されたかを問わず、ソースコードの派生的著作物に対する著作権も保有します。派生的著作物が提供されたソースコードをベースに作成されている場合、OpenVision の著作権情報が引き継がれる必要があります。

OpenVision Technologies, Inc.は、標準 Kerberos 5 配布の一部として本 Kerberos 管理システムを MIT に提供しています。この提供は、Kerberos 技術の継続的開発に対する当社の深い関与を示すと同時に、MIT および Kerberos コミュニティにより行われている価値ある取り組みへの感謝の意も表しています。

Matt Crawford (crawdad@fnal.gov)により提供される部分は、Universities Research Association, Inc.により運営されるフェルミ国立加速器研究所において、アメリカ合衆国エネルギー省との契約(DE-AC02-76CHO3000)に基づいて行われた取り組みです。

src/lib/crypto の部分は、以下の著作権を有します。

Copyright © 1998 by the FundsXpress, INC.

著作権所有。

本ソフトウェアをアメリカ合衆国から輸出する場合、アメリカ合衆国政府から特定ライセンスを取得しなければならないことがあります。輸出前に当該ライセンスを取得する責任は、輸出しようとする個人または組織が負います。

その制限内において、本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記され、さらに FundsXpress. の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布に関する広告または広報で使用されない限り、本合意によって許諾されます。FundsXpress は、いかなる目的においても、本ソフトウェアの適合性に関して明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。

src/lib/crypto/builtin/aes への AES 暗号化アルゴリズムの実装は、以下の著作権を有します。

Copyright © 2001, Dr Brian Gladman brg@gladman.uk.net, Worcester, UK.

著作権所有。

ライセンス条項

以下の場合、ソースおよびバイナリの両形式による本ソフトウェアの無料配布および使用は(変更の有無を問わず)許可されます。

本ソースコードの配布に、前述の著作権情報、この条件リスト、および以下の免責事項が含まれている。

バイナリ形式での配布に、上記の著作権情報、この条件リスト、および以下の免責事項がドキュメントおよび/またはその他の関連資料に含まれている。

書面による明確な許諾を得ないまま、本ソフトウェアを使用してビルドされた製品を宣伝するために著作権所有者の名前が使用されていない。

免責事項

本ソフトウェアは、「現状のまま」提供されるものであり、正確性および目的に応じた適合性を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、あらゆる属性に関して何ら保証されません。

Red Hat により提供される部分は、事前認証プラグインフレームワークおよび NSS 暗号実装を含め、以下の著作権を有します。

Copyright © 2006 Red Hat, Inc.

Portions copyright © 2006 Massachusetts Institute of Technology

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、Red Hat, Inc.の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

バンドリングされる Verto ソースコードには、以下のライセンスが適用されます。

Copyright 2011 Red Hat, Inc.

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/

または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

以下のファイルを含む、src/lib/gssapi の GSSAPI-SPNEGO への GSSAPI Mechglue の実装

lib/gssapi/generic/gssapi_err_generic.et
lib/gssapi/mechglue/g_accept_sec_context.c
lib/gssapi/mechglue/g_acquire_cred.c
lib/gssapi/mechglue/g_canon_name.c
lib/gssapi/mechglue/g_compare_name.c
lib/gssapi/mechglue/g_context_time.c
lib/gssapi/mechglue/g_delete_sec_context.c
lib/gssapi/mechglue/g_dsp_name.c
lib/gssapi/mechglue/g_dsp_status.c
lib/gssapi/mechglue/g_dup_name.c
lib/gssapi/mechglue/g_exp_sec_context.c
lib/gssapi/mechglue/g_export_name.c
lib/gssapi/mechglue/g_glue.c
lib/gssapi/mechglue/g_imp_name.c
lib/gssapi/mechglue/g_imp_sec_context.c
lib/gssapi/mechglue/g_init_sec_context.c
lib/gssapi/mechglue/g_initialize.c

lib/gssapi/mechglue/g_inquire_context.c
lib/gssapi/mechglue/g_inquire_cred.c
lib/gssapi/mechglue/g_inquire_names.c
lib/gssapi/mechglue/g_process_context.c
lib/gssapi/mechglue/g_rel_buffer.c
lib/gssapi/mechglue/g_rel_cred.c
lib/gssapi/mechglue/g_rel_name.c
lib/gssapi/mechglue/g_rel_oid_set.c
lib/gssapi/mechglue/g_seal.c
lib/gssapi/mechglue/g_sign.c
lib/gssapi/mechglue/g_store_cred.c
lib/gssapi/mechglue/g_unseal.c
lib/gssapi/mechglue/g_userok.c
lib/gssapi/mechglue/g_utils.c
lib/gssapi/mechglue/g_verify.c
lib/gssapi/mechglue/gssd_pname_to_uid.c
lib/gssapi/mechglue/mglueP.h
lib/gssapi/mechglue/oid_ops.c
lib/gssapi/spnego/gssapiP_spnego.h
lib/gssapi/spnego/spnego_mech.c
および以下の新規または変更済みファイルを含む、増分の初期実装
include/iprof_hdr.h
kadmin/server/iprof_svc.c
lib/kdb/iprof.x
lib/kdb/kdb_convert.c
lib/kdb/kdb_log.c
lib/kdb/kdb_log.h

lib/krb5/error_tables/kdb5_err.et

slave/kpropd_rpc.c

slave/kproplog.c

には、以下のライセンスが適用されます。

Copyright © 2004 Sun Microsystems, Inc.

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

Kerberos V5 には、カリフォルニア大学バークレー校で開発された、本著作権情報を含むドキュメントおよびソフトウェアが含まれています。

Copyright © 1983 Regents of the University of California.

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、REGENTS および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、REGENTS または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

LDAP データベースバックエンドを含め、Novell, Inc.により提供される部分には、以下のライセンスが適用されます。

Copyright © 2004-2005, Novell, Inc.

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ないまま、本ソフトウェアから派生した製品を宣伝または販売促進するために著作権所有者の名前が使用されていない。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

サンディア国立研究所により提供され、ミシガン大学情報技術統合センターにより開発される部分には、PKINIT 実装も含め、以下のライセンスが適用されます。

COPYRIGHT © 2006-2007

THE REGENTS OF THE UNIVERSITY OF MICHIGAN

著作権所有

派生的著作物の使用、複製および作成、および本ソフトウェアおよび派生的著作物の再配布は、その目的を問わず、ミシガン大学の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布での使用に関する広告または広報で使用されない限り許諾されます。本ソフトウェアの任意の部分の複製に、前述のミシガン大学の著作権情報およびその他の識別情報が含まれている場合、以下の免責事項も含まれていなければなりません。

本ソフトウェアは、「現状のまま」提供されるものであり、目的を問わずミシガン大学はその適合性に関して明言しません。また、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、ミシガン大学は何ら保証しません。ミシガン大学の REGENTS は、本ソフトウェアの使用によりまたはそれに関連して生じた請求に関して、例外的、間接的、偶発的、または必然的損傷に対し、そのような損傷の可能性をすでに知らされていたまたは今後知らされる場合でも、一切責任を負いません。

PKINIT コードに含まれる pkcs11.h ファイルには、以下のライセンスが適用されます。

Copyright 2006 g10 Code GmbH

Copyright 2006 Andreas Jellinghaus

本ファイルは無料ソフトウェアです。この情報が引き継がれている限り、作成者は特例として、変更の有無を問わず、本ファイルの複製および/または配布を無制限に許可します。

本ファイルは、有益であることを願って配布されますが、法律が許す範囲内で、商品性または特定目的への適合性の暗黙的保証を含め、何ら保証されません。

Apple Inc.により提供される部分には、以下のライセンスが適用されます。

Copyright 2004-2008 Apple Inc.著作権所有。

本ソフトウェアをアメリカ合衆国から輸出する場合、アメリカ合衆国政府から特定ライセンスを取得しなければならないことがあります。輸出前に当該ライセンスを取得する責任は、輸出しようとする個人または組織が負います。

その制限内において、本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記され、さらに Apple Inc.の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布に関する広告または広報で使用されない限り、本合意によって許諾されます。Apple Inc.は、いかなる目的においても、本ソフトウェアの適合性に関して明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。

src/util/support および src/lib/krb5/unicode への UTF-8 文字列操作の実装には、以下の著作権および許可情報が適用されます。

The OpenLDAP Public License

バージョン 2.8、2003 年 8 月 17 日

本ソフトウェアおよび付随するドキュメント(以下「ソフトウェア」)の再配布および使用は、変更の有無を問わず、以下の条件が満たされていることを条件に許諾されます。

ソース形式での再配布物には、著作権の宣言文および情報を含める必要があります。

バイナリ形式での再配布物には、該当する著作権の宣言文および情報、この条件リスト、および以下の免責事項を再配布物とともに提供されるドキュメントおよび/またはその他資料に複製する必要があります。

再配布物には、一切の変更を加えずに本ドキュメントのコピーを含める必要があります。

OpenLDAP Foundation は、本使用許諾条件を改訂することがあります。各改訂は、バージョン番号によって管理されます。本ソフトウェアは、使用許諾条件の本改訂または以降の改訂の条件に従って使用することができます。

本ソフトウェアは、OPENLDAP FOUNDATION および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、OPENLDAP FOUNDATION、その貢献者、あるいはソフトウェアの作成者または所有者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。また、このような損傷の可能性を知らされていた場合も同様です。

事前に書面による明確な許諾を得ない限り、広告内、あるいは本ソフトウェアの販売、使用、またはその他取引を促進する目的で、作成者および著作権所有者の名称を使用することはできません。本ソフトウェアの著作権は、常に著作権所有者が留保します。

OpenLDAP は、OpenLDAP Foundation の登録商標です。

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA.
著作権所有。本ドキュメントをコピーし、一切の変更を加えずに本ドキュメントの
コピーを配布することを許可します。

src/lib/krb5/krb のマークテストプログラムは、以下の著作権を有します。

Copyright © 2006 Kungliga Tekniska Högskola

(スウェーデン王立工科大学、ストックホルム)。

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の
条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の
免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下
の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に
複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支
持または促進する目的で、KTH の名称および貢献者の氏名を使用することはできま
せん。

本ソフトウェアは、KTH および貢献者により「現状のまま」提供されるものであり、
商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、
明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、KTH
または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例
外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、ま
たは利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その
経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過
失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性
を知らされていた場合も同様です。

src/lib/rpc および src/include/gssrpc への RPC 実装の一部は、以下の著作権および許
可情報を有しています。

Copyright © 2010, Oracle America, Inc.

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の
条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の
免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、「Oracle America, Inc.」の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Copyright © 2006,2007,2009 NTT (日本電信電話株式会社)。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布には、上記の著作権情報、この条件リスト、および以下の免責事項を、変更されていないこのファイルの1行目として含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアは、NTTにより「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、NTTは、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Copyright 2000 by Carnegie Mellon University

著作権所有

本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記され、さらにカーネギーメロン大

学の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布に関する広告または広報で使用されない限り、本合意によって許諾されます。

カーネギーメロン大学は、商品性および適合性のあらゆる暗黙的保証を含め、本ソフトウェアに関していかなる保証もしません。また、いかなる場合も、カーネギーメロン大学は、本ソフトウェアの使用または性能によりまたはそれに関連して生じた例外的、間接的または必然的損傷、または使用できないこと、データの喪失または利益の損失から生じるいかなる損傷に対して、契約の行為、過失またはその他の不法行為であるかどうかに関わらず、一切責任を負いません。

Copyright © 2002 Naval Research Laboratory (NRL/CCS)

本ソフトウェアおよびそのドキュメントを使用、複製、変更、および配布することは、著作権情報および本許可情報の両方がソフトウェアの複製、派生的著作物または変更されたバージョン、およびそれに関連する部分の全てにおいて記される限り、本合意によって許諾されます。

NRL は、本ソフトウェア「現状のまま」の状態での無料使用を許可するとともに、本ソフトウェアの使用により生じるあらゆる損傷に対して一切責任を負いません。

インターネット RFC からの引用部分には、以下の著作権情報が適用されます。

Copyright © The Internet Society (2006)。

本ドキュメントは、BCP 78 に含まれる権利、ライセンスおよび制限の対象であり、そこで説明されている場合を除き、作成者がそれらの全権利を有します。

本ドキュメントおよびここに記されている情報は、「現状のまま」を基本として提供されるものであり、貢献者、その貢献者が代表するまたは支援を受ける組織(該当する場合)、インターネット社会、およびインターネットエンジニアリング作業部会は、個々に記されている情報の使用がいかなる権利も侵害しないという保証、または商品性または特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、いかなる保証もしません。

Copyright © 1991, 1992, 1994 by Cygnus Support。

本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記される限り、本合意によって許諾されます。Cygnus Support は、いかなる目的においても、本ソフトウェアの適合性に関して明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

Copyright © 2006 Secure Endpoints Inc.

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/

または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

Fortuna を模した疑似乱数生成器の実装の一部には、以下の情報が適用されます。

Copyright © 2005 Marko Kreen

著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアは、作成者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、作成者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Copyright © 1994 by the University of Southern California

本ソフトウェアをアメリカ合衆国から輸出する場合、アメリカ合衆国政府から特定ライセンスを取得しなければならないことがあります。輸出前に当該ライセンスを取得する責任は、輸出しようとする個人または組織が負います。

その制限内において、本ソフトウェアおよびそのドキュメントをソースまたはバイナリ形式で複製、変更、および配布することは、そのような配布または使用に関連

するあらゆるドキュメントまたはその他の資料において、本ソフトウェアは南カリフォルニア大学が開発したものであることが認知されている限り、本合意によって許諾されます。

保証責任の排除。本ソフトウェアは、「現状のまま」で提供されます。南カリフォルニア大学は、明示であると暗黙であるとを問わず、一切の明言または保証を行いません。例として(ただしこれに限定されない)、南カリフォルニア大学は、商品性および特定目的への適合性を一切明言または保証しません。南カリフォルニア大学は、KSU ソフトウェアの使用者または配布者による請求に関するいかなる直接的、間接的、または必然的損傷に対しても、一切責任を負いません。

Copyright © 1995

The President and Fellows of Harvard University

本コードは、Jeremy Rassen によりハーバード大学に寄与されたソフトウェアの派生物です。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。

本製品には、カリフォルニア大学バークレー校およびその貢献者により開発されたソフトウェアが含まれています。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、REGENTS および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、REGENTS または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Copyright © 2008 by the Massachusetts Institute of Technology.

Copyright 1995 by Richard P. Basch. 著作権所有。

Copyright 1995 by Lehman Brothers, Inc. 著作権所有。

本ソフトウェアをアメリカ合衆国から輸出する場合、アメリカ合衆国政府から特定ライセンスを取得しなければならないことがあります。輸出前に当該ライセンスを取得する責任は、輸出しようとする個人または組織が負います。

その制限内において、本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記され、さらに Richard P. Basch、Lehman Brothers および M.I.T. の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布に関する広告または広報で使用されない限り、本合意によって許諾されます。Richard P. Basch、Lehman Brothers および M.I.T. は、いかなる目的においても、本ソフトウェアの適合性に関して明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であることを問わず、何ら保証されません。

以下の情報は `src/lib/krb5/krb/strptime.c` および `src/include/k5-queue.h` に適用されます。

Copyright © 1997, 1998 The NetBSD Foundation, Inc.

著作権所有。

本コードは、Klaus Klein により The NetBSD Foundation に寄与されました。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。

本製品には、The NetBSD Foundation, Inc. およびその貢献者により開発されたソフトウェアが含まれています。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、The NetBSD Foundation の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、NetBSD Foundation, Inc.により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、The NetBSD Foundation または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

以下の情報は、src/lib/krb5/unicode 内の Unicode ライブラリファイルに適用されます。

Copyright 1997, 1998, 1999 Computing Research Labs,

ニューメキシコ州大学

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、Computing Research Labs またはニューメキシコ州大学は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

以下の情報は、src/util/support/strncpy.c に適用されます。

Copyright © 1998 Todd C. Miller Todd.Miller@courtesan.com

本ソフトウェアを、有償または無償を問わず、使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報およびこの許可情報があらゆる複製に記される限り、本合意によって許諾されます。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および適合性のあらゆる暗黙的保証を含め、作成者は本ソフトウェアに関して何ら保証しません。いかなる場合も、作成者は、本ソフトウェアの使用または性能によりまたはそれに関連して生じた例外的、直接的、間接的または必然的損傷、または使用できないこ

と、データの喪失または利益の損失から生じるいかなる損傷に対して、契約の行為、過失またはその他の不法行為であるかどうかに関わらず、一切責任を負いません。

以下の情報は `src/util/profile/argv_parse.c` および `src/util/profile/argv_parse.h` に適用されます。

Copyright 1999 by Theodore Ts'o.

本ソフトウェアを、有償または無償を問わず、使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報およびこの許可情報があらゆる複製に記される限り、本合意によって許諾されます。本ソフトウェアは、「現状のまま」提供されるものであり、商品性および適合性のあらゆる暗黙的保証を含め、Theodore Ts'o (作成者)は本ソフトウェアに関して何ら保証しません。いかなる場合も、作成者は、本ソフトウェアの使用または性能によりまたはそれに関連して生じた例外的、直接的、間接的または必然的損傷、または使用できないこと、データの喪失または利益の損失から生じるいかなる損傷に対して、契約の行為、過失またはその他の不法行為であるかどうかに関わらず、一切責任を負いません。(この種の免責事項を必要とする、弁護士がすぐに訴訟を起こすアメリカの文化は病的ではないか?)

以下の情報は `src/util/profile/profile_tcl.c` の SWIC により生成されたコードに適用されます。

Copyright © 1999-2000, The University of Chicago

本ファイルは、この著作権に関する声明がそのまま残されている限り、ライセンスまたは課金なしで自由に再配布できます。

以下の情報は `src/lib/rpc` および `src/include/gssrpc` の一部に適用されます。

Copyright © 2000 The Regents of the University of Michigan. 著作権所有。

Copyright © 2000 Dug Song dugsong@UMICH.EDU. 著作権所有、全ての誤りは無効

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、REGENTS または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

MD4 アルゴリズムの実装には、以下の情報が適用されます。

Copyright © 1990, RSA Data Security, Inc.著作権所有。

本ソフトウェアの複製および使用は、本ソフトウェアまたは本機能に言及または参照する全てのものに「RSA Data Security, Inc. MD4 Message Digest Algorithm」と明記する限り許諾されます。

また、派生的著作物の作成および使用についても、派生的著作物に言及または参照する全てのものに「derived from the RSA Data Security, Inc. MD4 Message Digest Algorithm」と明記する限り許諾されます。

RSA Data Security, Inc.は、本ソフトウェアの商品性、および特定目的への本ソフトウェアの適合性のいずれに関しても明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

これらの情報は、本ドキュメントおよび/またはソフトウェアのいかなる部分の複製においても保持されなければなりません。

MD5 アルゴリズムの実装には、以下の情報が適用されます。

Copyright © 1990, RSA Data Security, Inc.著作権所有。

本ソフトウェアの複製および使用は、本ソフトウェアまたは本機能に言及または参照する全てのものに「RSA Data Security, Inc. MD5 Message Digest Algorithm」と明記する限り許諾されます。

また、派生的著作物の作成および使用についても、派生的著作物に言及または参照する全てのものに「derived from the RSA Data Security, Inc. MD5 Message Digest Algorithm」と明記する限り許諾されます。

RSA Data Security, Inc.は、本ソフトウェアの商品性、および特定目的への本ソフトウェアの適合性のいずれに関しても明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

これらの情報は、本ドキュメントおよび/またはソフトウェアのいかなる部分の複製においても保持されなければなりません。

以下の情報は、src/lib/crypto/crypto_tests/t_md driver.c に適用されます。

Copyright © 1990-2, RSA Data Security, Inc. Created 1990. 著作権所有。

RSA Data Security, Inc.は、本ソフトウェアの商品性、および特定目的への本ソフトウェアの適合性のいずれに関しても明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

これらの情報は、本ドキュメントおよび/またはソフトウェアのいかなる部分の複製においても保持されなければなりません。

src/lib/krb5 の一部には、以下の情報が適用されます。

Copyright © 1994 CyberSAFE Corporation。

Copyright 1990,1991,2007,2008 by the Massachusetts Institute of Technology。

著作権所有。

本ソフトウェアをアメリカ合衆国から輸出する場合、アメリカ合衆国政府から特定ライセンスを取得しなければならないことがあります。輸出前に当該ライセンスを取得する責任は、輸出しようとする個人または組織が負います。

その制限内において、本ソフトウェアおよびそのドキュメントを無償で使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報があらゆる複製に記され、当該著作権情報および本許可情報の両方が解説文書に記され、さらに M.I.T.の名称が事前に書面による明確な許諾を得ずに本ソフトウェアの配布に関する広告または広報で使用されない限り、本合意によって許諾されます。さらに、本ソフトウェアを変更する場合、そのソフトウェアに「変更されたソフトウェア」であることを示す名前を付ける必要があります、オリジナルの M.I.T.ソフトウェアと混同されるおそれのある方法で配布してはなりません。M.I.T.、Open Computing Security Group、および CyberSAFE Corporation は、いかなる目的においても、本ソフトウェアの適合性に関して明言しません。本ソフトウェアは、「現状のまま」提供されるものであり、明示であると暗黙であるとを問わず、何ら保証されません。

PADL Software により提供される部分には、以下のライセンスが適用されます。

Copyright (c) 2011, PADL Software Pty Ltd.著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、PADL Software の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、PADL Software および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、PADL Software または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

バンドリングされる libev ソースコードには、以下のライセンスが適用されます。

All files in libev are Copyright (C)2007,2008,2009 Marc Alexander Lehmann.

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

その代わりに、本パッケージのコンテンツは、GNU 一般公衆利用許諾書（「GPL」）バージョン 2 またはそれ以降のバージョンの条項の下で使用することができます。その場合、GPL の規定が前述の規定に代わり適用されます。本パッケージの独自のバージョンの使用を GPL の条項の下でのみ許可し、本ファイルの独自のバージョンを BSD ライセンスの下で他人に使用させないようにするには、前述の条項を削除す

ることによってその決定を示し、それらの条項に代えて、GPL が必要とする情報およびその他の条項を本パッケージの本ファイルおよびその他のファイルにおいて使用してください。前述の条項が削除されていない場合、受領者は本ファイルの独自のバージョンを BSD または GPL のいずれでも使用できるようになります。

Intel AESNI Sample Library から複製されたファイルには、以下のライセンスが適用されます。

Copyright © 2010, Intel Corporation。著作権所有。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。

事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、Intel Corporation の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、著作権所有者および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、著作権所有者または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Libedit の使用許諾条件

Brocade 仮想トラフィックマネージャには、コマンドラインエディタライブラリである Libedit のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 0.3

ライセンス概要: BSD 第 3 項「新規」または「改訂」ライセンス

著作権情報:

Copyright (c) 1997 The NetBSD Foundation, Inc. 著作権所有。

本コードは、Jaromir Dolecek により The NetBSD Foundation に寄与されたソフトウェアの派生物です。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。
2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。

本製品には、The NetBSD Foundation, Inc. およびその貢献者により開発されたソフトウェアが含まれています。

4. 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、The NetBSD Foundation の名称および貢献者の氏名を使用することはできません。

本ソフトウェアは、NetBSD Foundation, Inc. により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、The NetBSD Foundation または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

Copyright (c) 1992, 1993 The Regents of the University of California. 著作権所有。

本コードは、コーネル大学の Christos Zoulas によりバークレー校に寄与されたソフトウェアの派生物です。

ソースおよびバイナリ形式での再配布および使用は、変更の有無を問わず、以下の条件を満たしていることを条件に許諾されます。

1. ソースコードの再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項を含める必要があります。

2. バイナリ形式での再配布物には、上記の著作権情報、この条件リスト、および以下の免責事項をドキュメントおよび/または再配布物とともに提供されるその他資料に複製する必要があります。
3. 本ソフトウェアの機能または使用に言及する全ての広告物には、以下の情報を表記する必要があります。

本製品には、カリフォルニア大学バークレー校およびその貢献者により開発されたソフトウェアが含まれています。

4. 事前に書面による明確な許諾を得ない限り、本ソフトウェアから派生した製品を支持または促進する目的で、当大学名および貢献者の氏名を使用することはできません。

本ソフトウェアは、REGENTS および貢献者により「現状のまま」提供されるものであり、商品性および特定目的への適合性の暗黙的保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証されません。いかなる場合も、REGENTS または貢献者は、本ソフトウェアの使用により生じた直接的、間接的、偶発的、例外的、典型的、または必然的損傷(代替品またはサービスの調達、使用、データ、または利益の喪失、あるいは事業中断を含むが、これらに限定されない)に対し、その経緯および基づく責任の法理に関わらず、また契約、厳格責任、または不法行為(過失その他を含む)の有無を問わず、一切責任を負いません。このような損傷の可能性を知らされていた場合も同様です。

ZebOS の使用許諾条件

Brocade 仮想トラフィックマネージャには、IP Infusion, Inc.のライセンスの下で配布される ZebOS ルーティングスイートのコードが含まれています。The ZebOS routing suite is Copyright (C) 2001-2014 IP Infusion, Inc.著作権所有。

Curl の使用許諾条件

Brocade 仮想トラフィックマネージャには、データ転送用のコマンドラインツールおよびライブラリである Curl のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 7.42.1

ライセンス概要: MIT

著作権情報:

Copyright (c) 1996 - 2015, Daniel Stenberg, <daniel@haxx.se>.

著作権所有。

本ソフトウェアを、有償または無償を問わず、使用、複製、変更、および配布することは、その目的を問わず、前述の著作権情報およびこの許可情報があらゆる複製に記される限り、本合意によって許諾されます。

ソフトウェアは、商品性、特定目的への適合性、第三者への権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

本通知に記載がある場合を除き、著作権者から事前に書面の許可を得ない限り、著作権者の氏名は、本ソフトウェアの販売、使用またはその他の取引を推進するために、広告またはその他で使用するはなりません。

Jansson の使用許諾条件

Brocade 仮想トラフィックマネージャには、JSON サポートライブラリである Jansson のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 2.7

ライセンス概要: MIT

著作権情報:

Copyright (c) 2009-2014 Petri Lehtinen <petri@digip.org>

本ライセンスにより、本ソフトウェアおよび付随するドキュメントファイル(以下「ソフトウェア」)のコピーを入手した全ての人は、以下の条件に従い、ソフトウェアのコピーを使用、コピー、変更、マージ、公開、配布、サブライセンス、および/または販売する無制限の権利を含め、ソフトウェアを自由に扱うことが無償で許諾されます。また、ソフトウェアを提供された人についても同様です。

上記の著作権情報および本使用許諾情報をソフトウェアの全てのコピーまたは重要な部分に含めるものとします。

ソフトウェアは、商品性、特定目的への適合性、権利侵害がないことの保証を含め(ただしこれらに限定されない)、明示であると暗黙であるとを問わず、何ら保証のない状態で「現状のまま」提供されるものです。いかなる場合も、作成者または著作

権所有者は、ソフトウェアから、またはソフトウェアに関連して、あるいはソフトウェアの使用またはその他の扱いによって生じる申し立て、損傷、またはその他責任に対し、契約の行為、不法行為、またはその他の有無を問わず、一切責任を負いません。

Digest::SHA の使用許諾条件

Brocade 仮想トラフィックマネージャには、SHA メッセージダイジェストの計算に使用される Perl ライブラリである Digest::SHA のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 5.95

ライセンス概要: Perl Artistic License

著作権情報:

Copyright (C) 2003-2015 Mark Shelor

このライブラリはフリーソフトウェアであり、Perl 自体と同じ条件に基づいてこのライブラリを再配布および/または変更することができます。詳細については、Perl Artistic License を参照してください。

<http://search.cpan.org/perl/doc?perlartistic>

Sys::SysLog の使用許諾条件

Brocade 仮想トラフィックマネージャには、UNIX syslog プログラムとのインターフェースとして使用される Perl ライブラリである Sys::SysLog のコードが含まれています。以下の著作権情報が適用されます。

バージョン: 0.27

ライセンス概要: Perl Artistic License

著作権情報:

Copyright (C) 1990-2012 by Larry Wall and others.

このライブラリはフリーソフトウェアであり、Perl 自体と同じ条件に基づいてこのライブラリを再配布および/または変更することができます。詳細については、Perl Artistic License を参照してください。

<http://search.cpan.org/perl/doc?perlartistic>

CHAPTER 36 索引

:設定, 382

Application Firewall. 「Virtual Web Application Firewall」を参照

Client First. 「プロトコル」、
「Generic Client First」を参照

Configure スクリプト, 384, 502

DNS, 25, 29, 437, 450

DNS 名によるアクセスの制限, 387
アクティビティのモニタリング, 144
パブリックアドレス, 39, 40, 54, 502
特定の DNS アドレスでの監視, 59

DoS 攻撃におけるウイルス, 281

EDNS0 Client Subnet のサポート, 470

Error 400, 503

FTP. 「プロトコル」、「FTP」を参照

FTP (セキュア). 「SSL-wrapped FTP」
を参照

Generic Client First. 「プロトコル」、
「Generic Client First」を参照

Generic Server First. 「プロトコル」、
「Generic Server First」を参照

Generic Server First バナー, 193

Generic streaming. 「プロトコル」、
「Generic streaming」を参照

Google 検索リクエストルール, 177

HTTP. 「プロトコル」、「HTTP」を参
照

HTTPS, 504

セッション維持, 219

HTTP コンテンツキャッシング, 71

HTTP モニタ, 276

HTTP リダイレクト, 225

HTTP 攻撃, 281

IPv6

Linux、および Solaris での重複アド
レス検出の調整, 52

トラフィックマネージャの機能, 52

一般的な機能, 51

技術的制約, 52

概要, 51

IP アドレス

トラフィック IP アドレス, 25, 32, 37,
40, 43, 49, 56, 112, 383, 501, 503

トラブルシューティング, 125

トランスファ, 123

パッシブ, 117

共有トラフィック IP アドレス, 115
設定, 114

固定, 29, 37

IP トランスペアレンシ, 34

トラフィックマネージャのクラスタ,
44

ルーティング設定, 36

ローカルルーティングの制限事項, 37

IP ベースセッション維持, 218, 220, 224

Java

Java Extension のコンパイル, 186

TrafficScript からの Extensions の呼
び出し, 185

トラフィックマネージャの設定, 186

技術的な必要条件, 186

概要, 185

keepalive, 77, 91

N+M 冗長性, 43

トラフィッククラスタ, 44

NFS ファイルサーバ. 「ファイルシス
テム、共有」を参照

OCSP, 68, 232, 248

OCSP Stapling, 253

OS

セキュリティ, 280, 381, 385

OSPF, 50

PEM 符号化ファイル, 241

Ping モニタ, 275

PKCS 準拠ハードウェア, 259

POP3, 32, 54

Real Time Streaming プロトコル. 「プロトコル」、「RTSP」を参照

REST API, 64

RTSP. 「プロトコル」、「RTSP」を参照

トラフィックマネージャを使用する
メリット, 206

一般的な動作, 205

関連プロトコル, 206

RTSP カスタムモニタ, 276

RTSP ポート, 206

RuleBuilder, 169

Server First. 「プロトコル」、
「Generic Server First」を参照

Server First バナー. 「Generic Server
First バナー」を参照

Session Initiation Protocol. 「プロトコル」、
「SIP」を参照

トラフィックマネージャでの動作モ
ード, 210

動作, 207

機能, 207

SIP. 「プロトコル」、「SIP」を参照

SMTP, 32, 54, 501, 「プロトコル」を参
照 SMTP

SNMP, 144

トラップ, 360

バージョン, 145

認証とプライバシー, 147

SNMP による性能のモニタリング, 144

SOAP, 67, 176, 177, 178

SSL. 「プロトコル」、「SSL」を参照

HTTPS プロトコル, 231

カタログ, 233, 237

クライアント証明書

復号化あり, 68, 232, 243, 247

セッション ID キャッシュ, 253

中間証明書

追加、インポート、および更新,
239

復号化, 26, 64, 224, 233, 504

設定, 68, 241, 242, 243, 247

復号化ウィザード, 234

暗号化, 26, 68, 85, 231, 233, 234, 468

設定, 91, 254

概要, 231

証明書, 26, 68, 231

インポート, 238

クライアント, 235

コピー, 237

サーバ, 235

バックアップ, 139

作成, 235

管理サーバ, 386

自己署名, 237

証明書の設定, 235

証明書署名リクエスト(CSR), 237

SSL Session ID persistence (SSL セッ
ション ID による維持), 89, 220, 223, 224

SSL-wrapped FTP, 203

STARTTLS. 「プロトコル」を参照
SMTP

TCP, 88, 272, 383

TCP コネクションモニタ, 276

TCP トランザクションモニタ, 275

telnet, 500, 503

TrafficScript, 287

Extensions の使用, 185

Java のサポート, 26, 185

Java の使用, 185

RuleBuilder, 169

- RuleBuilder からの変換, 171
- XML のサポート, 176
- イベント, 149
 - ドキュメント, 167, 172
 - どんな場合にルールを使用するか, 167
 - リクエストルール, 67, 172, 173
 - ルールの作成, 171
 - ルールの適用, 66, 173
 - レスポンスルール, 172, 173
 - ログへの書き込み, 496
 - 例, 165, 174, 175, 176, 177, 229
 - 概要, 164
 - 認証, 179
- UDP, 78
 - SIP, 214
- Universal session persistence (ユニバーサルセッション維持), 167, 218, 220, 224, 229
- URL
 - リダイレクト, 89, 225
 - 攻撃, 282, 286
 - 書き換え, 165
 - 管理サーバ, 387
- Virtual Web Application Firewall, 156
- VLAN (ネットワーキング), 368
- Web サーバ, 30, 31, 40, 58, 72, 272, 281, 286, 503
- Web サービス, 67, 176, 177
- Web ワーム, 168, 233, 280, 281, 286
- X-Cluster-Client-Ip, 35, 255
- XML, 67, 176, 177, 178
- XPath, 67, 176, 177
- X-Zeus-Backend クッキー, 219, 223
- zconf, 137, 423
 - コマンド, 423
 - バックアップのエクスポート, 424
 - 設定ファイルのリスト表示, 425
 - 部分インポート, 139, 425
- アーキテクチャ, 25, 54, 281, 502
- アクション(イベント処理), 358
- アクティブ-アクティブ構成, 40, 41
 - ループバック付き, 41
- アクティブ-スタンバイ構成, 39, 42, 47
- アップグレード
 - OS, 385
 - バックエンド, 98, 153
- アプリケーションクッキーのモニタリング, 219, 222
- アプリケーションサーバ, 165
- アラート, 355
- イベント
 - カスタム, 149
- イベントタイプ, 356
- イベントログ, 131, 404
- イベント処理, 355
- インターフェースとサブネットのマッピング, 114, 117
- イントラネット, 232
- ウィザード, 56, 99, 153
 - ノードのドレイン, 99
- ウイルス, 168, 233, 280
- エラー
 - HTTP エラー, 287
 - IP アドレスの割り当て, 503
 - 一般的な設定エラー, 502
- エラーファイル内の HTTP ヘッダ, 83
- エラーログ. 「ログ」を参照
- オンライン証明書ステータスプロトコル. 「OCSP」を参照
- カスタマのプライオリティ設定のルール, 175
- カタログ, 132
 - CA および CRL, 232
 - SSL 証明書, 68, 233, 237, 239
 - サービスプロテクション, 69
 - モニタ, 272

- ルール, 67
- クッキー, 223
 - ...を使用したセッション維持, 222
 - クッキーを使用したセッション維持, 89, 219, 221, 222, 223
 - ルールとクッキー, 169, 175
- クッキー設定, 77
- クラウドクレデンシャル, 100, 154
- クラウドトラフィックマネージャ.
「トラフィックマネージャ仮想アプライアンス」を参照
- クラスタ
 - クラスタでのセッション維持, 218, 220, 221, 222, 223
 - クラスタのサイジング, 31
- クラスタの作成, 59
- グローバルサーバ負荷分散, 451, 「グローバル負荷分散」を参照
- グローバル設定, 135
- グローバル負荷分散, 142, 450
- コネクション
 - デバッグ, 150
- コネクションの終了, 225
- コネクション分析, 73
- コネクション制限
 - プール, 91
- コネクション拒否, 502, 503
- コネクション状況レポート, 495
- コネクション管理
 - プール, 91
 - 仮想サーバ, 77
- コンテンツキャッシング, 27, 304
 - TrafficScript, 312
 - Web キャッシュの設定, 307
 - ディスクベースのキャッシング, 309
 - バリエント, 311
 - ポリシー, 310
 - モニタリング, 308
 - 概要, 304
- 設定, 305
- コンテンツ圧縮, 72, 144
- コンテンツ管理, 32
- コントロール API, 406
 - C Sharp の例, 408
 - Mono の例, 408
 - Perl の例, 407
 - 概要, 406
- サービスプロテクション, 69, 70, 280
 - HTTP 固有の設定, 286
 - アクセス制限, 285
 - クラスのテスト, 284
 - クラスの作成, 283
 - クラスの使用, 287
 - コネクション制限, 282, 285
 - ネットワークアクセス制限, 282
 - ルール, 168, 287
 - ルールベースの保護, 282
 - 不正形式の HTTP のフィルタ, 282
 - 基本設定, 283
 - 性能, 288
 - 有効化, 283
- サービスレベルモニタリング, 299
 - TrafficScript, 301
 - カタログへのクラスの追加, 300
 - グラフ化, 303
 - リソースのプライオリティ設定, 302
 - 仮想サーバへのクラスの適用, 301
 - 例, 302
 - 設定, 300
- サービス妨害, 69, 277, 280, 281, 282, 284, 287
 - 分散型, 69, 280, 281
- サポートを利用するには, 505
- スーパーユーザ権限, 384, 502
- スループット、最大化, 34
- セキュア FTP. 「SSL-wrapped FTP」を参照
- セキュリティ
 - アクセスの制限, 387

セッションイニシエーションプロトコル
 モニタ, 276

セッション維持, 88
 ASP.net, 222
 ASP.net の使用, 219
 SSL セッション ID, 223
 X-Zeus-Backend クッキー, 223
 アプリケーションクッキーのモニタリング, 222
 クッキーの使用, 89
 サイジング, 226
 トランスペアレントセッションアフィニティ, 221
 ネームドノード, 219, 221
 ノード障害発生時のオプション, 225
 ユニバーサル, 165, 167, 220, 229
 各プール内, 85
 手法, 218
 概要, 216
 設定, 217
 負荷分散への影響, 217
 障害モード, 89

セッション維持マッピング, 224

タイムアウト設定, 78, 401

テストツールと手法, 493

テストリクエストの生成, 499

トップページ, 131

トラフィック IP ネットワーク, 114, 117

トラフィックマネージャ
 製品バージョン, 21

トラフィックマネージャのセキュリティ保護, 30, 31, 281, 385, 386

トラフィックマネージャの評価, 30, 502

トラフィックマネージャ仮想アプライアンス
 IP フォワーディング, 374
 NAT, 374

VLAN, 368
 ネットワーク設定, 367
 日付と時間, 377

トランスペアレントセッションアフィニティ, 219, 221

ネットワーク
 VLAN のタグ付け, 368

ネットワークレイアウト, 25, 29, 33, 383

ノードのドレイン. 「プール」を参照,
 「プール」を参照, 「プール」を参照,
 「プール」を参照

ノードの無効化. 「プール」を参照

パーセプティブ負荷分散アルゴリズム, 86

ハードウェアの必要条件, 25, 29, 33
 最小インストール, 30

パーミッショングループ. 「ユーザパーミッション」を参照

バックアップ, 96, 137, 139

バックアップ管理, 137
 部分インポート, 139, 425
 部分バックアップ, 139, 140

バックエンドサーバ, 29, 31, 37, 58, 86, 216, 501, 506

パッシブヘルスマonitoring
 有効化と無効化, 271

バッファ処理, 284

バナー, 402

ヒストリカルアクティビティグラフ, 149

ファイアウォール
 設定, 381

ファイアウォールの設定, 27, 30, 281, 381, 383, 385, 387
 手法, 381

ファイルシステムのセキュリティ, 384

プール

- SSL 暗号化, 91, 254
- オートスケーリング, 154
- コネクション管理, 91
- セッション維持, 88, 167, 217
- デフォルト, 56, 57, 59, 171
- ノードのドレイン, 98, 153, 226, 498
- ノードの無効化, 98
- プライオリティリスト, 96
- ヘルスのモニタリング, 95, 272, 277
- 作成, 57, 85
- 編集, 85
- 負荷分散, 85
- 障害, 95, 96
- フェールオーバー. 「フォールトトレランス」を参照
- フォールトトレランス, 29, 56, 118
 - バックエンド, 95, 501
 - ブロードキャスト, 122
 - フロントエンド, 32, 501, 502
- プライオリティリスト, 96
- プロトコル, 54, 64
 - FTP, 200
 - Generic Client First, 191
 - Generic Server First, 190
 - Generic Streaming, 194
 - HTTP, 194
 - RTSP, 205
 - SIP, 207
 - SMTP, 200
 - SSL. 参照
 - ファイアウォールに関する考慮事項, 381
- フロントエンドサーバの設定, 32
- ヘッダ
 - コンテンツ圧縮, 72
 - サービスプロテクションのためのフィルタ, 282, 286, 287
 - ホストヘッダのトラブルシューティング, 503
 - 値のロギング, 75
- 復号化された HTTPS に関する情報を含む, 243, 247
- 書き換え, 165
- ヘルスのモニタリング
 - カスタムモニタ, 275
 - ノード単位のモニタとプール全体のモニタ, 277
 - プロトコル固有のモニタ, 274
 - 内蔵モニタ, 273
 - 基本モニタ, 273
 - 概要, 272
 - 高機能モニタ, 273
- ホームページ, 131
- マルチサイトクラスタの管理, 427
- メモリ
 - 攻撃, 280, 286
 - 設定, 288
- モニタリング
 - Perl での作成, 279
 - SNMP, 144
 - アクティビティ, 25, 99, 143
 - ヘルス, 55, 272, 277
 - 外部モニタ, 277
- ユーザ
 - サスペンド, 401
- ユーザパーミッション
 - OS 内, 384
 - トラフィックマネージャ内, 392, 393, 400, 401, 499
- ラウンドロビン負荷分散アルゴリズム, 86
- ランダムノード負荷分散アルゴリズム, 86
- リクエストの割り当て, 41, 54, 59, 66, 85, 171
- リクエストレートシェイピング. レートシェイピングを参照
- リモート認証, 179
- ループ、無限. 「無限ループ」を参照

ルールを使用したルーティング, 68,
165, 174, 176, 229, 233

レートシェイピング, 293

TrafficScript, 296

Web スパイダ, 297

きめ細かい, 296

グラフ化, 298

設定, 294

ログ

TrafficScript によるログへの書き込み, 170, 496

イベントログ, 136, 493, 501, 503

サービスプロテクションの使用,
284

リクエスト, 75

ログイン失敗

許容回数, 401, 403

ロケーションヘッダ設定, 77

ワールドマップ, 150

仮想サーバ, 64

SSL トラフィックの復号化, 68, 241,
242, 243, 247

コネクション管理, 77

コンテンツ圧縮, 72

サービスプロテクションの適用, 69,
70, 287

リクエストロギング, 75

ルールの適用, 66, 173

低レベル設定, 79

例

TrafficScript, 165, 174

ユニバーサルセッション維持, 229

個々のノードのテスト, 498

共有ファイルシステム, 32, 34, 90, 277

初期設定, 54

加重ラウンドロビン負荷分散アルゴリズム, 86

地理的負荷分散. 「グローバル負荷分散」を参照

外部プログラムモニタ, 275

多重冗長構成. 「N+M 冗長性」を参照

帯域制御

TrafficScript の使用, 292

プールへのクラスの割り当て, 291

仮想サーバへのクラスの追加, 291

例, 292

概要, 289

設定, 290

手動でのプールと仮想サーバの作成,
58

接続レート, 69

最小接続負荷分散アルゴリズム, 86

最速レスポンス時間負荷分散アルゴリズム, 86

現在のアクティビティグラフ, 144

管理サーバ, 27, 386, 387, 400

ポート, 387

監査ログ, 404

管理サーバへのログイン, 56, 387, 401

簡易ネットワーク管理プロトコル.
「SNMP」を参照

設定

クッキー, 77

ロケーションヘッダ, 77

設定概要, 132

証明書失効リスト(CRL)

インポート, 240

証明書署名リクエスト(CSR), 231

認証, 231, 235

LDAP, 179

SSL, 232

認証コード. 「リモート認証」を参照

認証局(CA), 231, 232, 237

インポート, 240

負荷分散, 54, 85

アルゴリズム, 85, 498

およびセッション維持, 216, 217, 225

障害の診断, 25, 493, 499, 503

静的(Web)コンテンツ, 31, 32, 88, 170

障害プール. 「プール」を参照