

**WebSphere Application Server**



## **Caching Proxy 管理ガイド**

**バージョン 6.1**



**WebSphere Application Server**



## **Caching Proxy 管理ガイド**

**バージョン 6.1**

ご注意

本書および本書で紹介する製品をご使用になる前に、317 ページの『特記事項』に記載されている情報をお読みください。

この版は、以下のプログラムに適用されます。

WebSphere Application Server, バージョン 6.1

また、新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

<http://www.ibm.com/jp/manuals/main/mail.html>

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

<http://www.ibm.com/jp/manuals/> の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典： GC31-6920-00  
WebSphere Application Server  
Caching Proxy Administration Guide  
Version 6.1

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2006.5

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体\*)は、(財)日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注\* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、  
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 2006. All rights reserved.

© Copyright IBM Japan 2006

# 目次

図	xi
---	----

本書について	xiii
本書の対象読者	xiii
本書で使用されている規則と用語	xiii
アクセシビリティ	xiv
関連情報	xiv

## 第 1 部 Caching Proxy 入門 . . . . . 1

第 1 章 概要	3
Caching Proxy の基本構成	3
リバース・プロキシー (デフォルト)	3
フォワード・プロキシー	3
新規フィーチャーのサポート	5

第 2 章 「構成および管理」フォームの使用	7
ブラウザ要件	7
「構成および管理」フォームのアクセス	8
管理者のパスワードの設定	10

## 第 3 章 構成ウィザードの使用法 . . . . . 11

第 4 章 ibmproxy.conf ファイルの手作業での編集	13
----------------------------------	----

## 第 5 章 Caching Proxy の開始および停止 . . . . . 15

Linux および UNIX システムでの自動始動と終了	15
Linux および UNIX システムでの手動による始動	16
AIX の場合:	16
HP-UX の場合:	16
Linux の場合:	17
Solaris の場合:	17
Windows サービスとしての始動	17
Windows アプリケーションとしての始動	18
「スタート」メニューの使用	18
コマンド・プロンプトの使用	19
複数のプロキシー・サーバーの開始	19
Linux および UNIX システムでの手動による終了	20
終了コマンドの制限	21
Windows システムでの手動による終了	21
構成変更後の再始動	22

## 第 2 部 Caching Proxy プロセスの構成および調整 . . . . . 23

## 第 6 章 サーバーの定義 . . . . . 25

関連するディレクティブ	26
「構成および管理」フォーム	26

## 第 7 章 プロセス所有権の確立 . . . . . 29

関連するディレクティブ	29
「構成および管理」フォーム	30

## 第 8 章 接続の管理 . . . . . 31

関連するディレクティブ	32
「構成および管理」フォーム	33

## 第 9 章 プロキシー・サーバー・プロセスの調整 . . . . . 35

パフォーマンスに関連するディレクティブの設定	35
他のアプリケーションの調査	35
ページング・スペースの検査	36
ファイル・システムの調整	36
TCP/IP 構成の調整	36
強度のロード環境	
(HP-UX、Linux、Solaris、Windows) の場合の TCP	
時間待ち間隔の調整	36
Linux カーネルの調整	37
AIX スレッド・チューニング変数の調整	38

## 第 3 部 Caching Proxy の動作の構成 . . . . . 39

## 第 10 章 要求処理の管理 . . . . . 41

HTTP/FTP メソッドを使用可能にする	41
関連するディレクティブ	42
構成および管理フォーム	43
WebDAV メソッド、MS Exchange メソッド、およびユーザー定義のメソッドを使用可能にする	43
関連するディレクティブ	44
マッピング・ルールの定義	44
マッピング・ルール	45
サロゲート・サーバーの構成	46
関連するディレクティブ	46
構成および管理フォーム	46
ジャンクション再書き込みの使用可能化 (オプションル)	47
JunctionPrefix オプションを使用しないジャンクションの定義	47
JunctionPrefix オプションを使用するジャンクションの定義 (推奨される方法)	48
関連するディレクティブ	49
構成および管理フォーム	49
JunctionRewrite に代わる UseCookie	49
JunctionRewrite 機能を拡張する Transmogifier プラグインのサンプル	50

## 第 11 章 ローカル・コンテンツ送達の管理 . . . . . 53

文書ルート・ディレクトリーの定義 . . . . .	53
関連するディレクティブ . . . . .	53
「構成および管理」フォーム . . . . .	54
デフォルト・ウェルカム・ページの定義 . . . . .	54
関連するディレクティブ . . . . .	55
「構成および管理」フォーム . . . . .	55

## 第 12 章 FTP 接続の管理 . . . . . 57

FTP ファイルの保護 . . . . .	57
FTP サーバー・ログインの管理 . . . . .	57
FTP ディレクトリー・パスの管理 . . . . .	58
FTP チェーニングの管理 . . . . .	59

## 第 13 章 サーバー処理のカスタマイズ 61

サーバー側インクルード . . . . .	61
サーバー側インクルードに関する考慮事項 . . . . .	61
サーバー側インクルードの構成 . . . . .	61
サーバー側インクルードの形式 . . . . .	62
サーバー側インクルードのディレクティブ . . . . .	62
エラー・メッセージのカスタマイズ . . . . .	69
Real Time Streaming Protocol (RTSP) リダイレクト . . . . .	69
RTSP のリダイレクトについて . . . . .	69
RTSP の制限 . . . . .	70
RTSP の機能強化 . . . . .	70
RTSP リダイレクトの構成 . . . . .	70

## 第 14 章 ヘッダー・オプションの構成 71

関連するディレクティブ . . . . .	71
「構成および管理」フォーム . . . . .	72

## 第 15 章 アプリケーション・プログラミング・インターフェースについて . . . . . 73

関連するディレクティブ . . . . .	73
「構成および管理」フォーム . . . . .	73

## 第 4 部 プロキシ・サーバー・キャッシングの構成 . . . . . 75

## 第 16 章 プロキシ・サーバー・キャッシングの概説 . . . . . 77

キャッシュ・ストレージ . . . . .	77
キャッシュ索引 . . . . .	77
FTP キャッシュ . . . . .	78
DNS キャッシュ . . . . .	79
キャッシュの除外 . . . . .	79
キャッシュの管理 . . . . .	79

## 第 17 章 基本的なキャッシュの構成 . . . 81

1. キャッシュを使用可能にする . . . . .	81
2. キャッシュ・ストレージを構成する . . . . .	81
オプション・カスタマイズ . . . . .	83
キャッシュ・メモリーを設定する . . . . .	83

キャッシュ・メモリーをディスクに保管またはロードする . . . . .	84
キャッシュ・フィルターを設定する . . . . .	84
照会結果および動的に生成されたファイルのキャッシュを構成する . . . . .	84
ファイルの有効期限とガーベッジ・コレクションを構成する . . . . .	84
自動プリロードを構成する . . . . .	84
キャッシュ共用を構成する . . . . .	85
ログ記録を構成する . . . . .	85

## 第 18 章 キャッシュ対象の制御 . . . . . 87

URL ベースのキャッシュ・フィルターの構成 . . . . .	87
照会応答のキャッシュ . . . . .	88
照会応答キャッシュの追加要件 . . . . .	88
ローカル提供ファイルのキャッシュ . . . . .	89
部分 URL によるファイルのキャッシュ . . . . .	89
関連する構成ファイル・ディレクティブ . . . . .	89

## 第 19 章 キャッシュ・コンテンツの保守 91

ファイルの有効期限 . . . . .	91
キャッシュの新しさに関する追加情報 . . . . .	92
FTP での日付について . . . . .	93
キャッシュの新しさの構成 . . . . .	94
ガーベッジ・コレクション . . . . .	96
ガーベッジ・コレクションの設定 . . . . .	96

## 第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成 . . . . . 99

サーバーのホスト名の設定 . . . . .	100
キャッシュへの特定のファイルのプリロード . . . . .	101
キャッシュへの頻繁にキャッシュされるファイルのプリロード . . . . .	101
探求 . . . . .	102
関連したプロキシ構成ファイル・ディレクティブ . . . . .	104
キャッシュ・エージェントの手動による開始 . . . . .	105

## 第 21 章 共用キャッシュの使用 . . . . . 107

リモート・キャッシュ・アクセス . . . . .	107
リモート・キャッシュ・アクセスの構成 . . . . .	108
インターネット・キャッシング・プロトコル・プラグインの構成 . . . . .	108
ICP プラグインの構成 . . . . .	108

## 第 22 章 動的に生成されたコンテンツのキャッシング . . . . . 111

プロキシ・キャッシングのための IBM WebSphere Application Server の構成 . . . . .	112
アプリケーション・サーバーでの動的キャッシングの構成 . . . . .	112
アプリケーション・サーバー・アダプターの構成 . . . . .	112
動的キャッシングのための Caching Proxy の構成 . . . . .	113
動的キャッシング・プラグインを使用可能にするための Service ディレクティブの設定 . . . . .	113

ファイル・ソースを指定するための ExternalCacheManager ディレクティブの設定 . . . . .	114
<b>第 23 章 プロキシ・サーバー・キャッシュの調整 . . . . .</b>	<b>115</b>
キャッシュ・ストレージ・メディアの選択 . . . . .	115
ディスク・キャッシュのパフォーマンスの最適化 . . . . .	115
キャッシュのガーベッジ・コレクション . . . . .	116
プラットフォーム固有の最適化 . . . . .	116
AIX . . . . .	116
HP-UX および Solaris . . . . .	116
Windows . . . . .	116
<b>第 5 部 Caching Proxy セキュリ ティの構成 . . . . .</b>	<b>117</b>
<b>第 24 章 プロキシ・サーバーのセキ ュリティー . . . . .</b>	<b>119</b>
<b>第 25 章 サーバー保護セットアップ . . . . .</b>	<b>121</b>
「構成および管理」フォームを使用した保護の設定 . . . . .	121
構成ファイル・ディレクティブを使用した保護の設 定 . . . . .	122
デフォルトの保護設定 . . . . .	123
<b>第 26 章 Secure Sockets Layer (SSL) . . . . .</b>	<b>125</b>
SSL ハンドシェーク . . . . .	125
SSL パフォーマンスの調整 . . . . .	126
SSL トンネリング . . . . .	127
SSL トンネリングの構成 . . . . .	128
セキュア・リモート管理の構成 . . . . .	130
鍵および認証の管理 . . . . .	130
認証局 . . . . .	131
IBM 鍵管理ユーティリティの使用 . . . . .	131
新規の鍵データベース、パスワード、および stash ファイルの作成 . . . . .	133
CA の証明書を受け取る . . . . .	138
CA の証明書を保管する . . . . .	138
サポートしている暗号仕様 . . . . .	139
<b>第 27 章 暗号ハードウェアのサポート の使用可能化 . . . . .</b>	<b>143</b>
<b>第 28 章 Tivoli Access Manager プラ グインの使用 . . . . .</b>	<b>145</b>
構成 . . . . .	145
構成スクリプトを使用する前にとるステップ . . . . .	145
構成スクリプトの使用 . . . . .	145
Caching Proxy および Access Manager プラグイン の始動 . . . . .	146
<b>第 29 章 PAC-LDAP 許可モジュールの 使用 . . . . .</b>	<b>147</b>

概要 . . . . .	147
認証 . . . . .	147
許可 . . . . .	147
Lightweight Directory Access Protocol (LDAP) . . . . .	148
インストール . . . . .	148
セキュア PACD-LDAP サーバー接続のための追加 要件および制限事項 . . . . .	149
LDAP クライアント・パッケージでは GSKit が 必要 . . . . .	149
Linux システムには LD_PRELOAD 環境変数の 設定が必要 . . . . .	150
Linux システムで IBM Tivoli Directory Server (ITDS) 6.0 LDAP クライアントを使用する際、 PACD プロセスの開始に失敗する . . . . .	150
AIX システムで、IBM Tivoli Directory Server (ITDS) LDAP クライアントを使用する際、 PAC-LDAP モジュールをロードできない . . . . .	150
PAC-LDAP 許可モジュールを使用可能にする ibmproxy.conf ファイルの編集 . . . . .	150
PAC-LDAP 許可モジュール構成ファイルの編集 . . . . .	152
paccp.conf . . . . .	152
pac.conf . . . . .	153
pacpolicy.conf . . . . .	153
pac_ldap.cred の作成 . . . . .	154
pacd の始動および停止 . . . . .	155
<b>第 6 部 Caching Proxy のモニタ ー . . . . .</b>	<b>157</b>
<b>第 30 章 ロギングの構成 . . . . .</b>	<b>159</b>
ログについて . . . . .	159
ログ・ファイル名と基本オプション . . . . .	159
アクセス・ログ・フィルター . . . . .	160
ログを制御する理由 . . . . .	161
アクセス・ログ・フィルターの構成 . . . . .	161
デフォルトのログ設定値 . . . . .	162
ログの保守とアーカイブ . . . . .	163
ログ・ファイル・シナリオ . . . . .	164
<b>第 31 章 サーバー・アクティビティ ー・モニターの使用 . . . . .</b>	<b>167</b>
<b>付録 A. Caching Proxy コマンドの使 用 . . . . .</b>	<b>171</b>
cgiparse コマンド . . . . .	172
cgiutils コマンド . . . . .	175
htadm コマンド . . . . .	178
htcformat コマンド . . . . .	181
ibmproxy コマンド . . . . .	183
<b>付録 B. 構成ファイル・ディレクティブ . . . . .</b>	<b>185</b>
再始動時に変更されないディレクティブ . . . . .	185
ディレクティブの概要 . . . . .	185
許容値 . . . . .	186
構成ファイル・レコードの構文 . . . . .	187

Caching Proxy ディレクティブ	187	CacheAlgorithm - キャッシュ・アルゴリズムを指定する	202
AcceptAnything - すべてのファイルを提供する	187	CacheByIncomingUrl - キャッシュ・ファイル名を生成する場合の基準を指定する	202
AccessLog - アクセス・ログ・ファイルのパスを指定する	187	CacheClean - キャッシュされたファイルの保持期間を指定する	202
AccessLogExcludeMethod - 指定されたメソッドに必要なファイルまたはディレクトリーのログ項目を抑制する	188	CacheDefaultExpiry - デフォルトのファイル有効期限時間を指定する	203
AccessLogExcludeMimeType - 特定の MIME タイプのプロキシ・アクセス・ログ項目を抑制する	189	CacheDev - キャッシュ用のストレージを指定する	203
AccessLogExcludeReturnCode - 指定の戻りコードのログ項目を抑制する	190	CacheExpiryCheck - サーバーが有効期限切れファイルを戻すかどうかを指定する	204
AccessLogExcludeURL - 特定のファイルまたはディレクトリーのログ項目を抑制する	190	CacheFileSizeLimit - キャッシュに入れるファイルの最大サイズを指定する	204
AccessLogExcludeUserAgent - 特定のブラウザからのログ項目を抑制する	191	CacheLastModifiedFactor - 有効期限を決定する値を指定する	205
AddBlankIcon - ディレクトリー・リストの見出しの位置合わせに使用するアイコンの URL を指定する	191	CacheLocalDomain - ローカル・ドメインをキャッシュに入れるかどうかを指定する	206
AddDirIcon - ディレクトリー・リスト上のディレクトリーを示すアイコン URL を指定する	192	CacheMatchLanguage - 戻されるキャッシュ・コンテンツの言語プリファレンスを指定してください	206
AddEncoding - 特定の接尾部を持つファイルの MIME コンテンツ・エンコードを指定する	193	CacheMaxExpiry - キャッシュ・ファイルの最大存続時間を指定する	207
AddIcon - アイコンを MIME コンテンツ・タイプまたはエンコード・タイプにバインドする	193	CacheMemory - キャッシュ RAM を指定する	208
AddParentIcon - ディレクトリー・リスト上で親ディレクトリーを表すアイコンの URL を指定する	194	CacheMinHold - ファイルを使用可能に保つ期間を指定する	208
AddType - 特定の接尾部を持つファイルのデータ・タイプを指定する	194	CacheNoConnect - スタンドアロン・キャッシュ・モードを指定する	209
AddUnknownIcon - ディレクトリー・リスト上の不明ファイル・タイプのアイコン URL を指定する	196	CacheOnly - テンプレートと一致する URL を持つファイルだけをキャッシュに入れる	209
AdminPort - 管理ページまたはフォームを要求するためのポートを指定する	196	CacheQueries - 疑問符 (?) を含む URL へのキャッシュ応答を指定する	210
AggressiveCaching - キャッシュ不可能ファイルのキャッシュを指定する	197	CacheRefreshInterval - キャッシュ・オブジェクトの再妥当性検査の時間間隔を指定する	210
AlwaysWelcome - 要求されたディレクトリーからウェルカム・ファイルを検索するかどうかを指定する	198	CacheRefreshTime - キャッシュ・エージェントをいつ開始するかを指定する	211
appendCRLFtoPost - POST 要求に CRLF を付加する	198	CacheTimeMargin - ファイルをキャッシングする場合の最小存続時間を指定する	211
ArrayName - リモート・キャッシュ配列を指定する	198	CacheUnused - 未使用キャッシュ・ファイルの保持期間を指定する	212
Authentication - 認証ステップをカスタマイズする	199	Caching - プロキシ・キャッシュを使用可能にする	212
Authorization - 許可ステップをカスタマイズする	199	CompressAge - ログをいつ圧縮するかを指定する	212
AutoCacheRefresh - キャッシュ・リフレッシュを使用するかどうかを指定する	200	CompressCommand - 圧縮コマンドおよびパラメーターを指定する	213
BindSpecific - サーバーが 1 つまたはすべての IP アドレスのどちらにバインドするかを指定する	200	CompressDeleteAge - ログをいつ削除するかを指定する	214
BlockSize - キャッシュ内のブロックのサイズを指定する	201	CompressionFilterAddContentType - 圧縮したい HTTP 応答のコンテンツ・タイプを指定する	214
CacheAccessLog - キャッシュ・アクセス・ログ・ファイルのパスを指定する	201	CompressionFilterEnable - HTTP 応答を圧縮するための圧縮フィルターを有効にする	215
		ConfigFile - 追加構成ファイルの名前を指定する	216
		ConnThreads - 接続管理に使用する接続スレッドの数を指定	216
		ContinueCaching - キャッシングに必要なファイルの大きさを指定する	216



DefinePicsRule - コンテンツのフィルター操作規則を提供する	217
DefProt - テンプレートと一致する要求にデフォルトの保護セットアップを指定する	217
DelayPeriod - 要求間の一時停止を指定する	219
DelveAcrossHosts - ドメイン間のキャッシュへの格納を指定する	220
DelveDepth - キャッシュへの格納中にリンクをどこまで追跡するかを指定する	220
DelveInto - キャッシュ・エージェントがリンクをたどるかどうかを指定する	220
DirBackgroundImage - 背景イメージをディレクトリー・リストに指定する	221
DirShowBytes - 小さなファイルのバイト・カウントをディレクトリー・リストに表示する	221
DirShowCase - ディレクトリー・リスト上のファイルのソート時に大/小文字を区別する	222
DirShowDate - ディレクトリー・リストに最終変更日を表示する	222
DirShowDescription - ファイルの記述をディレクトリー・リストに表示する	222
DirShowHidden - 隠しファイルをディレクトリー・リストに表示する	223
DirShowIcons - アイコンをディレクトリー・リストに表示する	223
DirShowMaxDescrLength - ディレクトリー・リストの記述の最大長を指定する	223
DirShowMaxLength - ディレクトリー・リストに表示するファイル名の最大長を指定する	223
DirShowMinLength - ディレクトリー・リストに表示するファイル名の最小長を指定する	224
DirShowSize - ディレクトリー・リストにファイル・サイズを表示する	224
Disable - HTTP メソッドを使用不可にする	224
DisInheritEnv - CGI プログラムによって継承放棄される環境変数を指定する	225
DNS-Lookup - クライアントのホスト名を検索するかどうかを指定する	225
Enable - HTTP メソッドを使用可能にする	226
EnableTcpNodelay - TCP NODELAY ソケット・オプションを使用可能にする	226
Error - エラー・ステップをカスタマイズする	227
ErrorLog - サーバー・エラーがログに記録される場所のファイルを指定する	227
ErrorMessage - 特定のエラー条件にカスタマイズされたメッセージを指定する	228
Default - デフォルト	229
EventLog - イベント・ログ・ファイルのパスを指定する	230
Exec - 一致する要求に対して CGI プログラムを実行する	230
ExportCacheImageTo - キャッシュ・メモリーをディスクにエクスポートする	232
ExternalCacheManager - IBM WebSphere Application Server からの動的キャッシング用の Caching Proxy の構成	232

Fail - 一致する要求を拒否する	233
FIPSEnable - SSLV3 および TLS 用の Federal Information Processing Standard (FIPS) 承認済み暗号を使用可能にする	234
flexibleSocks - フレキシブルな SOCKS のインプリメントを使用可能にする	234
FTPDirInfo - ディレクトリーのウェルカム・メッセージまたは記述メッセージを生成する	235
ftp_proxy - FTP 要求のための別のプロキシ・サーバーを指定する	235
FTPUrlPath - FTP URL をどう解釈するかを指定する	235
Gc - ガーベッジ・コレクションを指定する	236
GCAdvisor - ガーベッジ・コレクション・プロセスをカスタマイズする	236
GcHighWater - ガーベッジ・コレクションをいつ開始するかを指定する	237
GcLowWater - ガーベッジ・コレクションをいつ終了するかを指定する	237
gopher_proxy - Gopher 要求のための別のプロキシ・サーバーを指定する	237
GroupId - グループ ID を指定する	238
HeaderServerName - HTTP ヘッダーに戻されるプロキシ・サーバーの名前を指定する	238
Hostname - サーバーの完全修飾ドメイン・ネームまたは IP アドレスを指定する	239
http_proxy - HTTP 要求のための別のプロキシ・サーバーを指定する	239
HTTPSCheckRoot - HTTPS 要求をフィルターに掛ける	239
ICP_Address - ICP 照会用の IP アドレスを指定する	240
ICP_MaxThreads - ICP 照会用の最大スレッド数を指定する	240
Occupier - ICP クラスターのメンバーを指定する	240
ICP_Port - ICP 照会用のポート番号を指定する	241
ICP_Timeout - ICP 照会に対する最大待機時間を指定する	241
IgnoreURL - リフレッシュしない URL を指定する	242
imbeds - サーバー側インクルード処理が使用されるかどうかを指定する	242
ImportCacheImageFrom - ファイルからキャッシュ・メモリーをインポートする	243
InheritEnv - CGI プログラムによって継承される環境変数を指定する	244
InputTimeout - 入力タイムアウトを指定する	244
JunctionReplaceUrlPrefix - JunctionRewrite プラグインと併用時に、接頭部を挿入する代わりに URL を置き換える	245
JunctionRewrite - URL 再書き込みを使用可能にする	245
JunctionRewriteSetCookiePath - JunctionRewrite プラグインとの併用時に Set-Cookie ヘッダーのパス・オプションを再書き込みする	246

JunctionSkipUrlPrefix - JunctionRewrite プラグインと併用時に接頭部を既に含んでいる URL の再書き込みをスキップする . . . . .	246
KeepExpired - リソースがプロキシで更新済みである場合、期限切れのリソースのコピーを戻すように指定する . . . . .	247
KeyRing - 鍵リング・データベースへのファイル・パスを指定する . . . . .	247
KeyRingStash - 鍵リング・データベースのパスワード・ファイルへのファイル・パスを指定する . . . . .	247
LimitRequestBody - PUT 要求または POST 要求の最大ボディ・サイズを指定する . . . . .	248
LimitRequestFields - クライアント要求のヘッダーの最大数を指定する . . . . .	248
LimitRequestFieldSize - 最大ヘッダー長および最大要求行を指定する . . . . .	248
ListenBacklog - サーバーが持てる listen バックログ・クライアント接続の数を指定する . . . . .	249
LoadInlineImages - 組み込みイメージのリフレッシュを制御する . . . . .	249
LoadTopCached - リフレッシュを実行する頻りにアクセスされるページの数指定する . . . . .	249
LoadURL - リフレッシュする URL を指定する . . . . .	250
Log - ログ・ステップをカスタマイズする . . . . .	250
LogArchive - ログ・アーカイブの動作を指定する . . . . .	251
LogFileFormat - アクセス・ログの形式を指定する . . . . .	251
LogToGUI (Windows only) - サーバー・ウィンドウにログ項目を表示する . . . . .	252
LogToSyslog - アクセス情報をシステム・ログに送信するかどうかを指定する (Linux と UNIX 専用) . . . . .	252
Map - ルールの突き合わせを行うため、要求パス・ストリングを使用して、マッチング要求を新規要求ストリングに変更する . . . . .	253
MapQuery - ルールの突き合わせを行うため、要求パスおよび照会ストリングを使用して、マッチング要求を新規要求ストリングに変更する . . . . .	254
MaxActiveThreads - アクティブ・スレッドの最大数を指定する . . . . .	256
MaxContentLengthBuffer - 動的データのためのバッファのサイズを指定する . . . . .	256
MaxLogFileSize - 各ログ・ファイルの最大サイズを指定する . . . . .	256
MaxPersistRequest - 持続接続で受信する要求の最大数を指定する . . . . .	257
MaxQueueDepth - キューに入れる URL の最大数を指定する . . . . .	258
MaxRuntime - キャッシュ・エージェントが実行する最大時間数を指定する . . . . .	258
MaxSocketPerServer - サーバーのオープン・アイドル状態のソケットの最大数を指定する . . . . .	258
MaxUrls - リフレッシュする URL の最大数を指定する . . . . .	259
Member - 配列のメンバーを指定する . . . . .	259

Midnight - ログのアーカイブに使用される API プラグインを指定する . . . . .	260
NameTrans - 名前変換ステップをカスタマイズする . . . . .	260
NoBG - Caching Proxy プロセスをフォアグラウンドで実行する . . . . .	261
NoCaching - URL がテンプレートと一致したファイルはキャッシュに入れないことを指定する . . . . .	262
NoLog - テンプレートと一致する特定のホストまたはドメインのログ項目を抑制する . . . . .	262
no_proxy - ドメインに直接接続するためのテンプレートを指定する . . . . .	263
NoCacheOnRange - Range 要求でキャッシングなしを指定する . . . . .	263
NoProxyHeader - ブロックするクライアント・ヘッダーを指定する . . . . .	264
NumClients - 使用するキャッシュ・エージェントのスレッドの数を指定する . . . . .	265
ObjectType - オブジェクト・タイプ・ステップをカスタマイズする . . . . .	265
OptimizeRuleMapping - ルールの数が増加したときに、着信要求のためのルール・マッピング・プロセスを最適化する . . . . .	266
OutputTimeout - 出力タイムアウトを指定する . . . . .	266
PacFilePath - PAC ファイルを含むディレクトリを指定する . . . . .	266
Pass - 要求を受け入れるためのテンプレートを指定する . . . . .	267
PersistTimeout - クライアントが別の要求を送信するのを待機する時間を指定する . . . . .	269
PICSDbLookup - PICS ラベル検索ステップをカスタマイズする . . . . .	269
PidFile (Linux および UNIX 専用) - Caching Proxy のプロセス ID を保管するファイルを指定する . . . . .	269
PKCS11DefaultCert, PKCS11DriverPath, PKCS11TokenPassword - IBM 4960 PCI 暗号アクセラレーター・カードをサポートする (AIX のみ) . . . . .	270
プラグイン・モジュールのディレクティブ . . . . .	271
Port - サーバーが要求を listen するポートを指定する . . . . .	271
PostAuth - PostAuth ステップをカスタマイズする . . . . .	272
PostExit - PostExit ステップをカスタマイズする . . . . .	272
PreExit - PreExit ステップをカスタマイズする . . . . .	273
Protect - テンプレートと一致する要求の保護セットアップを活動化する . . . . .	273
Protection - 名前付き保護セットアップを構成ファイル内に定義する . . . . .	278
Protection subdirectives - 一連のリソースの保護方法を指定する . . . . .	279
Proxy - プロキシ・プロトコルまたはリバース・プロキシを指定する . . . . .	282
ProxyAccessLog - プロキシ・アクセス・ログ・ファイルのパスを指定する . . . . .	283

ProxyAdvisor - プロキシ要求のサービスをカスタマイズする	284	ServerRoot - サーバー・プログラムのインストール先ディレクトリーを指定する	298
ProxyForwardLabels - PICS フィルター操作を指定する	284	ServerTerm - サーバー終了ステップをカスタマイズする	298
ProxyFrom - From: ヘッダーのクライアントを指定する	285	Service - サービス・ステップをカスタマイズする	299
ProxyIgnoreNoCache - 再ロード要求を無視する	285	SignificantURLTerminator - URL 要求の終了コードを指定する	300
ProxyPersistence - 持続接続機能を許可する	285	SMTPServer (Windows のみ) - sendmail ルーチン用に SMTP サーバーを設定する	300
ProxySendClientAddress - Client - IP: ヘッダーを生成する	286	SNMP - SNMP サポートを使用可能および使用不可にする	301
ProxyUserAgent - User Agent スtringを変更する	286	SNMPCommunity - SNMP のセキュリティ・パスワードを指定する	301
ProxyVia - HTTP ヘッダーの形式を指定する	287	SSLCaching - セキュア要求のキャッシュを使用可能にする	301
ProxyWAS - 要求が WebSphere Application Server に送信されることを指定する	287	SSLCertificate - 証明書の鍵ラベルを指定する	301
PureProxy - 専用プロキシを使用不可にする	287	SSLCryptoCard - インストール済み暗号カードを指定する	303
PurgeAge - ログの経過時間限度を指定する	288	SSLEnable - セキュア要求をポート 443 で listen するように指定する	303
PurgeSize - ログ・アーカイブのサイズの限度を指定する	288	SSLForwardPort - HTTP SSL のアップグレードのためにアドレス指定するポートを指定する	304
RCAConfigFile - ConfigFile の別名を指定する	289	SSLOnly - HTTP 要求のリスナー・スレッドを使用不可にする	304
RCAThreads - ポート当たりのスレッドの数を指定する	289	SSLPort - デフォルト以外の HTTPS listen ポートを指定する	304
ReadTimeout - 接続の時間制限を指定する	289	SSLTunneling - SSL トンネリングを使用可能にする	305
Redirect - 別のサーバーに送信される要求のテンプレートを指定する	290	SSLVersion - SSL のバージョンを指定する	305
RegisterCachedTransformer - Cookie ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる	291	SSLV2Timeout - SSLV2 セッションが有効期限切れになるまでの待ち時間を指定する	305
ReversePass - 自動的にリダイレクトされた要求をインターセプトする	292	SSLV3Timeout - SSLV3 セッションが有効期限切れになるまでの待ち時間を指定する	306
RewriteSetCookieDomain - 再書き込みする必要があるドメイン・パターンを指定する	292	SuffixCaseSense - 接尾部定義で大/小文字の区別を行うかどうかを指定する	306
RTSPEnable - RTSP リダイレクトを使用可能にする	293	SupportVaryHeader - HTTP Vary ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる	306
rtsp_proxy_server - リダイレクト用のサーバーを指定する	293	TLSV1Enable - トランスポート層セキュア・プロトコルを使用可能にする	308
rtsp_proxy_threshold - キャッシュへのリダイレクトまでの要求の数を指定する	294	Transmogriker - データ操作ステップをカスタマイズする	308
rtsp_url_list_size - プロキシ・メモリー中の URL の数を指定する	294	TransmogrikerWarning - 警告メッセージをクライアントへ送信する	309
RuleCaseSense - 大/小文字を区別しないアプリケーション URL からの要求をマップする	294	TransparentProxy - Linux で透過プロキシを使用可能にする	309
ScriptTimeout - スクリプトのタイムアウト設定値を指定する	295	UpdateProxy - キャッシュ宛先を指定する	310
SendHTTP10Outbound - プロキシ要求のプロトコル・バージョンを指定する	295	UserId - デフォルトのユーザー ID を指定する	311
SendRevProxyName - HOST ヘッダーの Caching Proxy ホスト名を指定する	296	V2CipherSpecs - SSL バージョン 2 についてサポートされる暗号仕様をリストする	311
ServerConnGCRun - ガーベッジ・コレクション・スレッドを実行する間隔を指定する	296	V3CipherSpecs - SSL バージョン 3 についてサポートされる暗号仕様をリストする	312
ServerConnPool - 起点サーバーへの接続のプールを指定する	297	WebMasterEmail - 選ばれたサーバー報告書を受け取るための電子メール・アドレスを設定する	313
ServerConnTimeout - 最大活動停止中期間を指定する	297		
ServerInit - サーバー初期化ステップをカスタマイズする	297		

WebMasterSocksServer (Windows のみ) - sendmail ルーチン用に Socks サーバーを設定する . . . . .	313
Welcome - ウェルカム・ファイルの名前を指定する . . . . .	313

特記事項. . . . .	<b>317</b>
商標 . . . . .	318



1. 探求 . . . . .	103	2. SSL トンネリング . . . . .	127
-----------------	-----	-------------------------	-----



---

## 本書について

まえがきでは、本書の対象読者と目的、その編成、アクセシビリティ機能、規則と用語、および関連する文書について説明します。

---

## 本書の対象読者

*Caching Proxy* 管理ガイド は、オペレーティング・システムおよび提供されるインターネット・サービスの経験豊富なネットワークおよびシステム管理者を対象に作成されています。*Caching Proxy* を事前に経験する必要はありません。

本書は、前のリリースの *Caching Proxy* をサポートするためのものではありません。

---

## 本書で使用されている規則と用語

本書では、以下のような書体およびキー操作の規則を使用しています。

表 1. 本書の規則

規則	意味
太字	グラフィカル・ユーザー・インターフェース (GUI) に関しては、太字は、メニュー、メニュー項目、ラベル、ボタン、アイコン、およびフォルダーを示します。また、太字にしないと周りのテキストと混同される恐れがあるコマンド名を強調するためにも使用されます。
モノスペース	コマンド・プロンプトから入力する必要のあるテキストを示します。また、モノスペースは、画面上のテキスト、コード例、およびファイルからの引用も示します。
イタリック	指定する必要がある可変値を示します (例: <i>fileName</i> でファイルの名前を指定します)。イタリックは、強調表示および書名の表示にも使用されます。
Ctrl-x	<i>x</i> がキーの名前である場合、制御文字のシーケンスを示します。例えば、Ctrl-c は、Ctrl キーを押しながら c キーを押すという意味になります。
Return	Return、Enter または左矢印が表示されていたキーを指します。
%	root 権限を必要としないコマンドの Linux™ および UNIX® コマンド・シェル・プロンプトを示します。
#	root 権限を必要とするコマンドの Linux および UNIX コマンド・シェル・プロンプトを示します。
C:¥	Windows® コマンド・プロンプトを示します。
コマンド入力	コマンドを“入力”する、または“発行”すると指示された場合は、コマンドを入力してから Return を押します。例えば、「ls コマンドを入力してください」という指示は、コマンド・プロンプトに <b>ls</b> とタイプしてから Return を押すという意味になります。
[ ]	構文記述内のオプション項目を囲みます。
{ }	構文記述内の、項目を選択する必要があるリストを囲みます。
	構文記述の中で { } (中括弧) で囲まれた選択項目リストにある項目を区切るために使用されます。



表 1. 本書の規則 (続き)

規則	意味
...	構文記述内の省略記号は、前の項目を 1 回以上繰り返すことができることを示します。 例にある省略記号は、簡潔にするために例から情報が省略されていることを示します。

## アクセシビリティ

アクセシビリティ機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。

WebSphere® Application Server, バージョン 6.1 の主なアクセシビリティ機能は次のとおりです。

- スクリーン・リーダー・ソフトウェアおよびデジタル・スピーチ・シンセサイザーを使用して、画面に表示された内容を聞くことができます。IBM ViaVoice などの音声認識ソフトウェアを使用してデータを入力し、ユーザー・インターフェースをナビゲートすることもできます。
- マウスの代わりにキーボードを使用して機能を操作できます。
- 提供されているグラフィカル・インターフェースの代わりに標準テキスト・エディターまたはコマンド行インターフェースを使用して、Application Server 機能を構成および管理できます。特定の機能のアクセシビリティについての詳細は、それらの機能に関する資料を参照してください。

## 関連情報

- *Edge Components* 概念、計画とインストール, GD88-6859-00
- *Edge Components* プログラミング・ガイド, GD88-6860-00
- *Load Balancer* 管理ガイド, GD88-6862-00
- *IBM WebSphere Edge Services Architecture*
- IBM ホーム Web サイト: [www.ibm.com/](http://www.ibm.com/)
- IBM® WebSphere Application Server 製品 Web サイト:  
[www.ibm.com/software/webservers/appserv/](http://www.ibm.com/software/webservers/appserv/)
- IBM WebSphere Application Server ライブラリー Web サイト:  
[www.ibm.com/software/webservers/appserv/library.html](http://www.ibm.com/software/webservers/appserv/library.html)
- IBM WebSphere Application Server サポート Web サイト:  
[www.ibm.com/software/webservers/appserv/support.html](http://www.ibm.com/software/webservers/appserv/support.html)
- IBM WebSphere Application Server Information Center:  
[www.ibm.com/software/webservers/appserv/infocenter.html](http://www.ibm.com/software/webservers/appserv/infocenter.html)
- IBM WebSphere Application Server Edge Components Information Center:  
[www.ibm.com/software/webservers/appserv/ecinfocenter.html](http://www.ibm.com/software/webservers/appserv/ecinfocenter.html)



---

## 第 1 部 Caching Proxy 入門

ここでは、Caching Proxy コンポーネントの概説、構成および管理フォームおよび構成ウィザードの使用方法、ibmproxy.conf ファイルの手動による編集、プロキシ・サーバーの開始手順と停止手順について説明します。

ここには、次の章が含まれます。

3 ページの『第 1 章 概要』

7 ページの『第 2 章 「構成および管理」 フォームの使用』

11 ページの『第 3 章 構成ウィザードの使用法』

13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』

15 ページの『第 5 章 Caching Proxy の開始および停止』



---

## 第 1 章 概要

リバース・プロキシまたはフォワード・プロキシとして機能することで、Caching Proxy は、クライアントからのデータ要求をインターセプトし、コンテンツ・ホスティング・マシンから要求された情報を取り出し、そのコンテンツをクライアントに引き渡します。一般に、この要求は Web サーバー・マシン (起点サーバーまたはコンテンツ・ホストとも呼びます) に保管された文書に対して出されていて、HTTP (Hypertext Transfer Protocol) 経由で送達されることが最も多くなっています。ただし、FTP (ファイル転送プロトコル) や Gopher など、他のプロトコルを扱うように Caching Proxy を構成することができます。

Caching Proxy では、キャッシュ可能なコンテンツをローカル・キャッシュに保管してからリクエスターに引き渡します。キャッシュ可能なコンテンツの例としては、静的な Web ページ、および動的に生成されるものの、まれにフラグメントが変化する JavaServer Pages (JSP) ファイルがあります。キャッシングにより、Caching Proxy は、以後同じコンテンツが要求されたときにそのコンテンツをキャッシュから直接引き渡すことができるので、改めてコンテンツ・ホストから検索するよりはるかに迅速に処理できます。

重要: Caching Proxy は、すべての Edge Components インストールで利用可能です。ただし、以下の例外があります。

- Caching Proxy は、Itanium 2 または AMD Opteron 64 ビット・プロセッサで稼働している Edge Components インストールには利用不可です。
- Caching Proxy は、Load Balancer for IPv4 and IPv6 の Edge Components インストールには利用不可です。

---

## Caching Proxy の基本構成

2 つの基本プロキシ構成、リバース・プロキシとフォワード・プロキシがあります。

### リバース・プロキシ (デフォルト)

デフォルトでは、Caching Proxy は、リバース・プロキシ・サーバーとして構成されています。リバース・プロキシ構成では、プロキシ・サーバーは、1 つ以上のコンテンツ・サーバーとインターネットの間に配置されます。この構成は、プロキシ・サーバーのホーム・サイトに保管されているコンテンツについて、インターネット・クライアントからの要求を受信します。プロキシ・サーバーは、クライアントにとっては起点 (コンテンツ) サーバーであるように見えます。クライアントには、要求が別のサーバーに送信されたことは分かりません。

### フォワード・プロキシ

Caching Proxy を、フォワード・プロキシ・サーバーとして構成することもできます。ただし、プロキシを使用するために、クライアント・ブラウザを個別に構成する必要があります。フォワード・プロキシ構成では、プロキシ・サーバーは、クライアントとインターネットの間に配置されます。Caching Proxy は、クラ

クライアントの要求をインターネットを通じてコンテンツ・ホストに転送し、取得データをキャッシュして、その取得データをクライアントに配布します。

フォワード・プロキシー構成を有効にするには、`ibmproxy.conf` 構成ファイルで以下の変更を行う必要があります。

- 以下の行からコメントを外し、`Caching Proxy` が転送するプロトコルを指定します。

```
Proxy http:*
Proxy ftp:*
Proxy gopher:*
```

- SSL 要求をフォワード・プロキシー構成で処理できるようにするため、`SSL トンネリング`を使用可能に設定します。

```
SSLTunneling On
```

SSL トンネリングの詳細については、128 ページの『SSL トンネリングの構成』を参照してください。

- `Enable` ディレクティブを使用して、`CONNECT` メソッドを有効にします。

```
Enable CONNECT OutgoingPorts All
```

または

```
Enable CONNECT OutgoingPorts 443
```

`Enable CONNECT` メソッドのフォーマットおよび使用可能なオプションについては、128 ページの『SSL トンネリングの構成』を参照してください。

これらの変更を行うと、フォワード・プロキシーは以下のことができるようになります。

- HTTP または FTP で、クライアントからの要求に応答する。
- `gopher` 検索エンジンからの要求に応答する。
- トランザクションの期間中に、クライアントとその現行サーバーの間のアフィニティーを維持する。

## 透過プロキシー (Linux システムのみ)

フォワード `Caching Proxy` のバリエーションとして、透過 `Caching Proxy` があります。このロールでは、`Caching Proxy` は基本のフォワード `Caching Proxy` と同じ機能を実行しますが、その実行はクライアントに認識されません。透過 `Caching Proxy` 構成は、Linux システムでのみサポートされています。

通常のフォワード `Caching Proxy` と同様、透過 `Caching Proxy` は、インターネット/ゲートウェイ近くのマシンにインストールします。ただし、クライアント・ブラウザ・プログラムは、フォワード `Caching Proxy` に要求を直接送信するようには構成されません。クライアントは、その構成にプロキシーが存在することを認識しません。代わりに、クライアント要求をインターセプトし、それらの要求を透過 `Caching Proxy` に送信するよう、ルーターが構成されます。

この構成のためのディレクティブについては、309 ページの『TransparentProxy - Linux で透過プロキシーを使用可能にする』を参照してください。

---

## 新規フィーチャーのサポート

バージョン 6.1 の「*Caching Proxy 管理ガイド*」には、新しく文書化されたフィーチャーおよび修正に関する更新情報が含まれています。

最も重大なフィーチャーには、次のようなものがあります。

- フォワード・プロキシのサポート

フォワード・プロキシの構成については、3 ページの『フォワード・プロキシ』を参照してください。

- Linux システム専用の透過プロキシのサポート

透過 (フォワード) プロキシ・ディレクティブについては、309 ページの『TransparentProxy - Linux で透過プロキシを使用可能にする』を参照してください。

- WebDAV メソッド、Microsoft Exchange Server メソッド、およびユーザー定義メソッドのサポート

これらのメソッドについては、43 ページの『WebDAV メソッド、MS Exchange メソッド、およびユーザー定義のメソッドを使用可能にする』を参照してください。

- HTTP 圧縮フィルターのディレクティブ

これらのディレクティブについては、214 ページの『CompressionFilterAddContentType - 圧縮したい HTTP 応答のコンテンツ・タイプを指定する』および 215 ページの『CompressionFilterEnable - HTTP 応答を圧縮するための圧縮フィルターを有効にする』を参照してください。

- キャッシングなしの Range 要求のディレクティブ

このディレクティブについては、263 ページの『NoCacheOnRange - Range 要求でキャッシングなしを指定する』を参照してください。

- ルール・マッピング最適化のディレクティブ

このディレクティブについては、266 ページの『OptimizeRuleMapping - ルールの数が増加したときに、着信要求のためのルール・マッピング・プロセスを最適化する』を参照してください。

- MapQuery ディレクティブ

MapQuery は、Map ディレクティブと類似したディレクティブです。ルールを突き合わせるために、パス・ストリングと照会ストリングの両方を使用します。

このディレクティブについては、254 ページの『MapQuery - ルールの突き合わせを行うため、要求パスおよび照会ストリングを使用して、マッチング要求を新規要求ストリングに変更する』を参照してください。

- RuleCaseSense ディレクティブ

このディレクティブについては、294 ページの『RuleCaseSense - 大/小文字を区別しないアプリケーション URL からの要求をマップする』を参照してください。

- AIX システム用に、IBM 4960 PCI 暗号アクセラレーター・カードをサポートするための、追加のディレクティブが用意されています。

これらのディレクティブについては、270 ページの

『PKCS11DefaultCert、PKCS11DriverPath、PKCS11TokenPassword - IBM 4960 PCI 暗号アクセラレーター・カードをサポートする (AIX のみ)』を参照してください。

- SSLCertificate ディレクティブでの論理式オプション

このディレクティブの論理式オプションについては、301 ページの

『SSLCertificate - 証明書の鍵ラベルを指定する』を参照してください。

- Proxy または ProxyWAS ルール用として使用可能な追加オプション

Caching Proxy には、実行時に追加のパターン・マッチングを必要とするディレクティブが提供されています。Caching Proxy のパフォーマンスを向上させるために、これらのディレクティブを、Proxy または ProxyWAS ルールのオプションとして使用することができます。Proxy または ProxyWAS ルールのこれらの追加オプションについての詳細は、282 ページの『Proxy - プロキシー・プロトコルまたはリバース・プロキシーを指定する』を参照してください。

---

## 第 2 章 「構成および管理」フォームの使用

Caching Proxy は、プロキシ・サーバーを構成するようにクライアントに要求し、そのために使用できる HTML フォームと一緒に出荷されます。これらのフォームは、ローカル・プロキシ・サーバー構成ファイル `ibmproxy.conf` を編集する CGI プログラムを実行します。これらのフォームを使用するには、プロキシ・サーバーが実行中で、フォームが入っているローカル・ディレクトリーからそのフォームを渡せるように構成されていなければなりません。

デフォルトでは、Caching Proxy は `ibmproxy.conf` に組み込まれた PASS ディレクティブでインストールされ、これにより「構成および管理」フォームにアクセスできるようになります。クライアントがこのプロキシ・サーバーからデフォルトのホーム・ページを要求すると、`Frntpage.html` が表示されます。このページには、「構成および管理」フォームの開始ページ `wte.html` へのハイパーテキスト・リンクが入っています。

「構成および管理」フォームは保護されているので、使用する前にクライアント認証が必要です。管理者の ID およびパスワードの設定方法については、10 ページの『管理者のパスワードの設定』を参照してください。

---

### ブラウザ要件

「構成および管理」フォームへのアクセスに使用する Web ブラウザーは、以下の要件をサポートしている必要があります。

- **HTML 4.0:** すべてのフォームは HTML 4.0 仕様に沿って記述されています。使用する Web ブラウザーは HTML 4.0 とフレームセットをサポートしている必要があります。
- **Java 1.1 と JavaScript:** アプレットは Java 1.1 仕様に沿って記述されています。使用する Web ブラウザーは Java 1.1 準拠の Java 仮想マシンをサポートしている必要があります。アプレットは Java 2.0 仕様に準拠する Java 仮想マシンとは互換性がありません。JavaScript と Java の両方が使用できなければなりません。
- **256 色:** Web ブラウザーを実行するワークステーションでは、最低 256 色をサポートしている必要があります。

推奨されるブラウザは、Mozilla および Firefox (Linux、UNIX、および Windows システムの場合)、および Internet Explorer (Windows システムの場合) です。

Mozilla、Firefox、および Internet Explorer などの各ブラウザの具体的なバージョンについては、次の Web サイトを参照し、サポートされるソフトウェアについての Web ページへのリンクをたどってください:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006921>

注:

1. 64 ビットの PowerPC Linux システムでは、Mozilla ブラウザーで「構成および管理」フォームにアクセスすることはできません。このアーキテクチャーには使

用可能な SDK がいないためです。代わりに、サポートされた Web ブラウザーのある別のマシンから「構成および管理」フォームにアクセスすることができます。

2. 管理コンソールを開始するときに、ログインのプロンプトが 2 回出される場合は、Internet Explorer の Java 設定が正しく設定されていません。Internet Explorer でこれを訂正するには、「ツール」>「インターネット オプション」>「詳細設定」を選択し、「Use Java 2 v1.4.X」チェック・ボックスを選択解除します。

---

## 「構成および管理」フォームのアクセス

「構成および管理」フォームにアクセスするには、次の手順を実行してください。

1. プロキシ・サーバーが実行されていることを確認する。プロキシ・サーバー開始の詳細については、15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。
2. HTTP ブラウザーに、そのプロキシ・サーバーのホーム・ページ (Frntpage.html) または「構成および管理」開始ページ (wte.html) を要求するように指示します。

注: このページは、ユーザーのプロキシ・サーバーの実際のマッピング規則に依存するため、括弧で示したデフォルトのページとは異なる場合があります。

`http://your.server.name[:port][/directory][/page.html]`

ここで、

- `your.server.name` は、ユーザーのホスト (例えば、`http://www.ibm.com/`) の完全パス名です。
  - `[:port]` ユーザーのプロキシ・サーバーが 80 以外のポートで管理要求を listen している場合には、サーバー名の後にポート番号を含めてください。  
`http://your.server.name:port`
  - `[/directory]` URL へのディレクトリーの追加は、マッピング規則によって異なります。
  - `[/page.html]` HTML ページを指定する必要があるのは、それがウェルカム・ページとしてリストされていない場合だけです。ウェルカム・ページについては、54 ページの『デフォルト・ウェルカム・ページの定義』を参照してください。
3. 「構成および管理フォーム」リンクをクリックして、サーバー構成フォームを表示します。管理者のユーザー名およびパスワードを求めるプロンプトが出されます。許可されたユーザー名とパスワードを入力します。「Caching Proxy 構成クライアント」ウィンドウがオープンします。

注:

- a. 左側のナビゲーション・フレームの内容は、メインページが表示されてから数秒かかってロードされることがあります。
- b. Windows 2003 システムでは、管理フォーム (CGI スクリプト) を要求する接続は、接続が完了する前にリセットされる可能性があります。その結果、ブラウザーはデータを受信しなかったことを報告するか、ページが使用不可で



あるというメッセージを表示する可能性があります。この問題を回避するには、MaxActiveThreads を 200 より大きい値に増やすか、ConnThreads を 50 より大きい値に増やしてリセットされる接続を解決するようにします。これらのディレクティブについて詳しくは、256 ページの『MaxActiveThreads - アクティブ・スレッドの最大数を指定する』および 216 ページの

『ConnThreads - 接続管理に使用する接続スレッドの数を指定』を参照してください。

4. 左側にあるナビゲーション・フレームで、以下の 5 つのメイン・カテゴリの構成フォームが表示されます。

- プロキシ構成
- キャッシュ構成
- サーバー構成
- サーバー活動モニター
- プラグイン構成

ヘッダーの左側にある三角形のポインターをクリックして、そのカテゴリ内の構成フォームを展開します。フォームをクリックしてオープンします。フォームの入力フィールドに現在の構成値 (ある場合に) が表示されます。インストール以降に構成を変更していない場合は、構成値はデフォルト値です。

5. 任意のフォームで、特定の機能についての構成情報を入力します。フォームはそれぞれ、どのような変更を行うかを決定する際に役立つ指示を提供しています。詳細については、それぞれのフォームの一番上にあるヘルプ・アイコン (疑問符(?)) をクリックしてください。以下のリンクが提供されます。
  - 「フィールド・ヘルプ」 - 各画面パネルのフィールドについての説明。
  - 「...実行方法?」 - 特定のタスクを実行するためにフォームを使用する際の詳細ステップ。
  - 「索引」 - ヘルプ情報の索引。

6. フォームに入力したら、「**実行依頼**」をクリックして、行った変更でサーバー構成を更新します。「**実行依頼**」ボタンは、各フォームの入力フィールドの下にあります。フォームで示した変更を行いたくない場合には、「**リセット**」をクリックしてください。これにより、フォーム上のフィールドが元の値に戻ります。

7. 「**実行依頼**」をクリックしてその入力を受け入れられると、上部フレームに次のメッセージが表示されます。

The requested configuration changes have been completed successfully

入力が受け入れられない場合には、上部のフレームにエラー・メッセージが表示され、どの設定が受け入れられなかったかが示されます。

8. プロキシ・サーバーを再始動するには、上部のフレームにあるサーバー再始動のアイコン「I」をクリックします。プロキシ・サーバーは再始動コマンドを受け取ると、クライアントからの要求の受け入れを停止します。ただし、すでに処理中の要求についてはその処理は完了します。変更した構成ファイルを再ロードすると、プロキシ・サーバーはクライアント要求の受け入れを再開します。

注: 「構成および管理」フォームを使用するか、または ibmproxy.conf ファイルを編集して特定のディレクティブを変更するには、再始動でなく、サーバー

を完全に停止してからもう一度始動して、変更を有効にする必要があります。これらのディレクティブは、185 ページの表 6 に示されています。

---

## 管理者のパスワードの設定

Caching Proxy パッケージをインストールした後で、「構成および管理」フォームにアクセスするために管理者 ID とパスワードを作成する必要があります。デフォルトの プロキシ・サーバー 構成では、「構成および管理」フォームを要求したユーザーについて、Linux および UNIX システムの

/opt/ibm/edge/cp/server\_root/protect/ ディレクトリー、または Windows システムの %Program Files%\IBM\edge\cp\etc\ ディレクトリーにある webadmin.passwd パスワード・ファイルを使用して認証を行います。パッケージのインストールでは、既存の webadmin.passwd ファイルは上書きされません。

webadmin.passwd ファイルに管理者項目を追加するには、次のコマンドを使用します。

- Linux および UNIX システムの場合:

```
# htadm -adduser /opt/ibm/edge/cp/server_root/protect/webadmin.passwd
```

プロンプトが出されたら、**htadm** プログラムに管理者のユーザー名、パスワード、および実名を指定します。

- Windows システムの場合:

```
cd "%Program Files%\IBM\edge\cp\server_root\protect%"  
htadm -adduser webadmin.passwd"
```

プロンプトが出されたら、**htadm** プログラムに管理者のユーザー名、パスワード、および実名を指定します。

**注:** 管理者ユーザー名およびパスワードには、たとえオペレーティング・システムが大/小文字の区別をしない場合でも、大/小文字の区別があります。「構成および管理」フォームにアクセスするときは、htadm コマンドを使用して、必ず正確なユーザー名およびパスワードを入力するようにしてください。

**htadm** コマンドの詳細については、178 ページの『htadm コマンド』を参照してください。

---

## 第 3 章 構成ウィザードの使用法

Caching Proxy 構成ウィザードを使用すると、インストール済みの Caching Proxy を迅速に構成することができます。このプログラムは、Caching Proxy の動作を代理として機能するように変更する、必須のディレクティブのみを設定します。プロキシ・サーバーには、追加の構成が必要になる場合があります。

Caching Proxy 構成ウィザードを使用するには、次のようにします。

1. 「構成」ウィザードを開始します。

Windows システムでは、「スタート」->「プログラム」->「IBM WebSphere」->「Edge Components」->「Caching Proxy」->「構成ウィザード」をクリックします。

Linux および UNIX システムでは、コマンド  
`/opt/ibm/edge/cp/cpwizard/cpwizard.sh` を入力します。

2. プロキシ・サーバーが HTTP 要求を listen するネットワーク・ポートを選択します。
3. ターゲット・コンテンツ・サーバーの名前を入力します。
4. プロキシ・サーバー管理者のユーザー ID とパスワードを入力します。

注:

1. 構成ウィザードでは、以下のディレクティブを設定します。

```
Port port
Proxy /* http://content server :port
```

2. 「構成ウィザード」を使用してプロキシ・サーバーを構成し、SSL を使用可能にする場合は、ポート 443 を介して受け取るプロキシ要求に対するマッピング・ルールを作成する必要があります。詳細については、44 ページの『マッピング・ルールの定義』を参照してください。

例:

```
Proxy /* http://content server :443
```

または

```
Proxy /* https://content server :443
```

**Linux システムにおける制限事項:** Caching Proxy 構成ウィザードでは、キーボード・ショートカットは機能しません。



---

## 第 4 章 ibmproxy.conf ファイルの手作業での編集

Caching Proxy は、ibmproxy 構成ファイルを編集することで手動で構成するか、構成および管理フォームを使用して構成することができます。

- Linux および UNIX システムでは、ibmproxy.conf ファイルは /etc/ ディレクトリにあります。
- Windows システムでは、ibmproxy.conf ファイルは C:\Program Files\IBM\edge\cp\etc\ja\_JP\ にあります。

構成ファイルはディレクティブと呼ばれるステートメントで構成されます。構成を変更するには、構成ファイルを編集し、ディレクティブを変更することによって、変更を保管します。構成ファイルの編集には、emacs や vi など、ほとんどすべてのテキスト・エディターを使用できます。

**注:** Solaris Common Desktop Environment (CDE) に組み込まれているテキスト・ファイル・エディターは使用しないようにしてください。Solaris エディターは、場合によってはファイルの所有グループを変更し、ファイル・リンクのプロパティを変更してしまうので、「構成および管理」フォームで構成ファイルに書き込むことができなくなります。

構成ファイルに対する変更は、サーバーを再始動したときに有効となります。ただし、185 ページの表 6 に示されているディレクティブのいずれかを変更した場合は別です。そのリストにあるディレクティブのいずれかを変更した場合には、サーバーを一度停止してから、再び始動しなければなりません。詳細については、15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。

185 ページの『付録 B. 構成ファイル・ディレクティブ』では、各構成ファイル・ディレクティブおよび構文の詳細について説明しています。



---

## 第 5 章 Caching Proxy の開始および停止

Caching Proxy は、オペレーターの介入を最低限に抑え、バックグラウンド・プロセスとして継続実行するように設計されています。通常、プロキシ・サーバーはマシンのブート・サイクル中に開始し、保守が必要な場合にのみ停止します。プロキシ・サーバーは必要に応じて手動で開始することができます。また、プロキシ・サーバーにはアクティブなクライアント接続を中断させることなく、効果的にプロキシ・サーバーを停止して開始するという再始動命令を渡すことができます。

---

### Linux および UNIX システムでの自動始動と終了

Linux および UNIX システムでは、Caching Proxy のインストール時に、**ibmproxy** 初期化スクリプトと、関連するシンボリック・リンクが、該当する `/etc/` ディレクトリーに配置されます。これらのスクリプトは、オペレーティング・システムの始動および終了ルーチンに組み込まれます。自動再始動のための構成の設定値は、**ibmproxy** スクリプトを編集し、**ibmproxy** コマンドのオプションを変更して変更することができます。

#### 注: Solaris ファイル記述子の制限

Caching Proxy 初期化スクリプトは、Solaris のシステム全体のファイル記述子に関する制限によって、必要なファイル記述子の最大数を設定できない場合があります。システム全体の最大値が Caching Proxy 初期化スクリプトの設定値より小さい場合には、システム全体の制限が使用されます。値が小さすぎる (1024 未満) ことから生ずるプロキシのパフォーマンス問題を回避するために、ファイル記述子の制限を変更することができます。現在使用可能な記述子の数を調べるには、**ulimit** コマンドを出してください。値が 1024 より小さい場合には、ファイル記述子の制限を大きくしてください。ファイル記述子の制限を 1024 まで大きくするには、以下の行を `/etc/system` ファイルに追加してください。

```
set rlim_fd_cur=0x400
```

#### 自動始動と終了を使用不可にする

自動始動と終了を使用不可にするには、次のようにします。

- AIX システムでは、初期化ファイルから **ibmproxy** コマンドを除去します。
- HP-UX システムでは、**ibmproxy** を参照する次のリンクを除去します。
  - `/sbin/rc1.d/K154ibmproxy`
  - `/sbin/rc2.d/S880ibmproxy`
- Linux システムでは、実行レベルのサブディレクトリーにある `/etc/rc.d/init.d/ibmproxy` へのシンボリック・リンクを除去します。

SUSE Linux では、**ibmproxy** への次のリンクを除去します。

- `/etc/rc.d/rc3.d/S20ibmproxy`

- /etc/rc.d/rc3.d/K20ibmproxy
- /etc/rc.d/rc4.d/S20ibmproxy
- /etc/rc.d/rc4.d/K20ibmproxy
- /etc/rc.d/rc5.d/S20ibmproxy
- /etc/rc.d/rc5.d/K20ibmproxy

Red Hat Linux では、**ibmproxy** への次のリンクを除去します。

- /etc/rc.d/rc0.d/K54ibmproxy
- /etc/rc.d/rc1.d/K54ibmproxy
- /etc/rc.d/rc2.d/K54ibmproxy
- /etc/rc.d/rc6.d/K54ibmproxy
- /etc/rc.d/rc3.d/S88ibmproxy
- /etc/rc.d/rc5.d/S88ibmproxy

- Solaris システムでは、**ibmproxy start** コマンドと、その 2 つの強制終了スクリプトを除去します。
  - /etc/rc2.d ディレクトリーから S88ibmproxy を削除します。
  - /etc/rc0.d ディレクトリーから K54ibmproxy を削除します。
  - /etc/rc1.d ディレクトリーから K54ibmproxy を削除します。

---

## Linux および UNIX システムでの手動による始動

始動メソッドには関係なく、最終的に **ibmproxy** コマンドが直接コマンド・プロンプトから呼び出されるか、またはスクリプトから呼び出されます。**ibmproxy** コマンドの詳細については、183 ページの『**ibmproxy** コマンド』を参照してください。以下に、最も一般的に使用される引き数だけを指定した例を挙げます。

### AIX の場合:

- **startsrc** コマンドを使用してデフォルト・ロケールのプロキシ・サーバーを始動するには、以下のように入力します。  

```
startsrc -s ibmproxy
```
- **startsrc** コマンドを使用してデフォルト以外のロケールのプロキシ・サーバーを始動するには、以下のように入力します。  

```
startsrc -s ibmproxy -e "LC_ALL=locale"
```
- **startsrc** コマンドを使用せずに、デフォルトのランタイム設定でプロキシ・サーバーを始動するには、以下のように入力します。  

```
ibmproxy
```

### HP-UX の場合:

- 初期化スクリプトを実行してプロキシ・サーバーを始動するためには、ルート・プロンプトで次のように入力します。  

```
/sbin/init.d/ibmproxy start
```
- 初期化スクリプトを実行せずにバックグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。  

```
/usr/sbin/ibmproxy
```



- 初期化スクリプトを実行せずにフォアグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
/usr/sbin/ibmproxy -nobg
```

## Linux の場合:

- 初期化スクリプトを実行してプロキシ・サーバーを始動するためには、ルート・プロンプトで次のように入力します。

```
/etc/rc.d/init.d/ibmproxy start
```

- 初期化スクリプトを実行せずにバックグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
/usr/sbin/ibmproxy
```

- 初期化スクリプトを実行せずにフォアグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
/usr/sbin/ibmproxy -nobg
```

- 既存 SQUID 構成ファイルの `squidConfig.file` を使用してプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
squidConfig.file -r /etc/errors_icons.conf
```

ここで、`errors_icons.conf` ファイルは、ディレクトリーのブラウズ時に指定されたファイル・タイプに対して使用するアイコンを識別します。

## Solaris の場合:

- 初期化スクリプトを実行してプロキシ・サーバーを始動するためには、ルート・プロンプトで次のように入力します。

```
/etc/init.d/ibmproxy start
```

- 初期化スクリプトを実行せずにバックグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
/usr/sbin/ibmproxy
```

- 初期化スクリプトを実行せずにフォアグラウンド・プロセスとしてプロキシ・サーバーを始動するには、ルート・プロンプトで次のように入力します。

```
/usr/sbin/ibmproxy -nobg
```

---

## Windows サービスとしての始動

Caching Proxy が Windows サービスとしてインストールされている場合には、他の Windows サービスと同様に開始することができます。

1. 「スタート」->「設定」(Windows 2000 の場合) ->「コントロール パネル」をクリックします。
2. 「コントロール パネル」ウィンドウで、「管理ツール」->「サービス」をダブルクリックします。
3. 「サービス」ウィンドウで、「Caching Proxy」を強調表示します。
4. 「開始」をクリックして、Caching Proxy サービスを開始します。

サービスとして Caching Proxy をインストールした場合、Windows の開始時に、そのサーバーが自動的に始動するように構成することができます。そのような場合、

プロキシが要求を満たすことができるようにするために、ログオンする必要はありません。プロキシを自動的に開始させるには次のようにしてください。

1. 「スタート」->「設定」(Windows 2000 の場合) ->「コントロール パネル」をクリックします。
2. 「コントロール パネル」ウィンドウで、「管理ツール」->「サービス」をダブルクリックします。
3. 「サービス」ウィンドウで、「**Caching Proxy**」を強調表示します。
4. 「自動」ラジオ・ボタンをクリックしてから「開始」をクリックし、Windows の始動時に Caching Proxy サービスを自動開始します。

#### PATH 環境変数のリフレッシュ

「サービス」ウィンドウで Caching Proxy が開始済みとマークされているが、プロキシが作動していない場合は、プロキシのインストール後にマシンが再始動されていない可能性があります。Caching Proxy サービスがデスクトップと対話するように設定されている場合には、再始動していないと、ポップアップ・ボックスにエラー・メッセージ「メッセージ・カタログ・エラー: メッセージ・カタログをロードできないか、または無効です」が現れることがあります。

Windows レジストリーで PATH 環境変数がリフレッシュされるように、マシンを再始動する必要があります。レジストリーがリフレッシュされていない場合には、PATH 変数は正しい Caching Proxy および GSK7 パスを示していても、正しく機能しない可能性があります。

注: Caching Proxy と他のアプリケーション (ネットワーク・ファイル・システムなど) の両方が稼働する場合は、Windows システムに競合が存在する可能性があります。Caching Proxy では、同じくサービスとして稼働中のファイル・システム・アプリケーションによって所有されているリモート・ドライブを含んだパスを解釈できない場合もあります。

この問題は、Windows PATH 環境変数内で Caching Proxy サービスのパスの前に、ファイル・システム・サービスのパスが現れた場合に発生します。PATH 変数を変更してファイル・システム・サービスを設定の終わりの方に配置すると、この問題を解決できます。

この問題は、Windows サービスとして稼働していないアプリケーションによって制御されているリモート・ドライブには影響しません。例えば、Caching Proxy ローカル・エリア・ネットワーク (LAN) を介して可視の Windows マシン上の共用ドライブにアクセスできます。

---

## Windows アプリケーションとしての始動

### 「スタート」メニューの使用

Caching Proxy が Windows アプリケーションとしてインストールされると、インストール・プロシージャによって、**Caching Proxy** 項目が「スタート」メニューのサブメニューとして作成されます。Caching Proxy をアプリケーションとして始動するには、「スタート」->「プログラム」->「IBM WebSphere」->「Edge Components」->「Caching Proxy」をクリックします。

この始動プロシージャによって、現在の構成設定値を使用してプロキシー・サーバーが実行されます。始動時に他の設定値を指定したい場合は、コマンド始動プロシージャを使用します (次のセクションを参照してください)。

## コマンド・プロンプトの使用

Windows または DOS コマンド・プロンプトからサーバーを始動するには、**ibmproxy** コマンドを使用します。サーバーをインストールした後で Windows を終了して再始動していない場合には、このコマンドで絶対パス名を入力します。デフォルトでは、以下のとおりです。

```
c:\Program Files\IBM\edge\cp\bin\ibmproxy.exe
```

**ibmproxy** コマンドは、サーバーを現行の構成の設定で始動します。インストール以降、サーバー構成を変更していなければ、現行の構成は、インストール時に入力した情報と、デフォルト・オプションをベースにします。

Caching Proxy をサービスとして実行するようにインストールしている場合でも、**ibmproxy** コマンドは、サーバーをアプリケーションとして始動します。強制的にサーバーをアプリケーションとして実行する場合には、コマンド・オプション **-noservice** を指定することもできます。他のコマンド・オプションでは、実行時に構成の設定値が変更されます。

---

## 複数のプロキシー・サーバーの開始

プロキシー・サーバーの複数のインスタンスを同時に実行することができますが、それぞれのインスタンスは、別々のポートを **listen** しなければなりません。AIX システムは、SRC では、1 つのインスタンスしか開始することができません。構成ファイルはポート番号を識別し、この番号は特定のマシンで各サーバーに対して異なっている必要があるので、サーバーのすべてのインスタンスに対して固有の構成ファイルを指定しなければなりません。サーバーの追加のインスタンスを (少なくとも 1 つのインスタンスがすでに実行中のとき) 開始するには、次のコマンドを入力します。

- Linux および UNIX:  

```
ibmproxy -r other_config_file
```
- Windows の場合:  

```
ibmproxy -noservice -r other_config_file
```

ここで、*other\_config\_file* は固有の構成ファイルです。

サーバーの複数のインスタンスを開始するときには、インスタンスごとに表示されるプロセス ID を記録しておきます。この ID はサーバーの特定のインスタンスを停止するために必要です。

注: サーバーの複数のインスタンスが実行中の Linux システムでは、コマンド **/etc/rc.d/init.d/ibmproxy stop** で最後に開始されたサーバーだけが停止します。その他のインスタンスは、個別に停止しなければなりません。関連情報については、20 ページの『Linux および UNIX システムでの手動による終了』を参照してください。

## Linux および UNIX システムでの手動による終了

サーバーを停止するには、次のような条件があります。

- サーバーを停止するのは、プロセスを開始したユーザーまたはスーパーユーザー `root` のいずれかでなければならない。
- サーバーを始動したメソッドと同じメソッドを使用しなければならない。 次の表には、始動メソッドおよびそれらに関連した停止メソッドがリストされています。

表 2. Linux および UNIX システムでの始動および停止メソッド

始動メソッド	停止メソッド
/etc/inittab (AIX の場合)	stopsrc -s ibmproxy を入力します。
/sbin/init.d (HP-UX の場合)	/sbin/init.d/ibmproxy stop を入力します。
/etc/rc.d/init.d (Linux の場合)	/etc/rc.d/init.d/ibmproxy stop を入力します。
ibmproxy	<ol style="list-style-type: none"><li>1. <b>ibmproxy</b> プロセス ID の検索: AIX では、<code>ps -aef   grep "ibmproxy"</code> と入力します。Linux では、<code>ps -aux   grep ibmproxy   grep server_ID</code> と入力します。Solaris および HP-UX では、<code>ps -ef   grep "ibmproxy"</code> と入力します。</li><li>2. <b>ibmproxy</b> プロセスの停止: <code>kill process_id</code> を入力します。</li></ol> <p>このマシン上のすべてのサーバーを停止するには、<code>killall ibmproxy</code> と入力します。</p>
ibmproxy -nbg	ctrl-c を入力します。
ibmproxy -r -other_config_file (AIX の場合)	stopsrc -s ibmproxy -p process_id を入力します。
ibmproxy -r -other_config_file (Linux の場合)	<ol style="list-style-type: none"><li>1. <b>ibmproxy</b> プロセス ID の検索: <code>ps aux   grep ibmproxy   grep process_id</code> を入力します。</li><li>2. <b>ibmproxy</b> プロセスの停止: <code>kill process_id</code> を入力します。</li></ol>

注: 透過プロキシを開始した場合には、Caching Proxy を停止した後で、透過プロキシのカーネル拡張、および関連したファイアウォール・ルールもアンロードします。 `root` として、次のコマンドを入力します。

```
ibmproxy -unload
```

サーバーを停止するには、`root` プロンプトで次のように入力します。

- AIX の場合: `stopsrc -s ibmproxy`
- HP-UX の場合: `/sbin/init.d/ibmproxy stop`
- Linux の場合: `/etc/rc.d/init.d/ibmproxy stop`
- Solaris の場合: `/etc/init.d/ibmproxy stop`

## 終了コマンドの制限

終了コマンドを使用する場合には、以下の制限があります。

- **AIX、HP-UX、および Linux**

AIX、HP-UX、および Linux システムでは、Caching Proxy システムを停止させるためのコマンドが、Caching Proxy プロセスのみを終了させることがあります。この動作の原因となる AIX コマンドは、**stopsrc -s ibmproxy** コマンドです。この動作の原因となる HP-UX および Linux コマンドは、**ibmproxy -stop** コマンドです。

LDAP サーバーで使用される PACD プロセスは、プロキシ・サーバーの終了後も稼働したまま残ることがあります。以下のような **kill** コマンドを使用することによって、PACD プロセスを安全に終了させることができます。

```
kill -15 PACD_process_ID
```

- **Solaris の場合**

Solaris システムで **ibmproxy -stop** コマンドを発行しても、他のオペレーティング・システムで発行された場合と同じ効果が生じません。Solaris コードの制限のため、**ibmproxy -stop** が Solaris プラットフォームで使用された場合は、サーバー終了プラグイン・ステップが実行されません。

この制限は、プロキシ・サーバー・ソフトウェアの場合と同様に、ユーザーがインプリメントするプラグインでも適用されます。

LDAP サーバーで使用される PACD プロセスは、プロキシ・サーバーの終了後も継続して稼働することが可能です。以下のような **kill** コマンドを使用することによって、PACD プロセスを安全に終了させることができます。

```
kill -15 PACD_process_ID
```

---

## Windows システムでの手動による終了

他の Windows プログラムを停止するのと同じ方法で、Caching Proxy サーバーを停止することができます。

プロキシをサービスとしてインストールする場合は、次のようにします。

1. 「スタート」->「設定」(Windows 2000 の場合) ->「コントロール パネル」をクリックします。
2. 「コントロール パネル」ウィンドウで、「管理ツール」->「サービス」をダブルクリックします。
3. 「サービス」ウィンドウで、「**Caching Proxy**」を強調表示します。
4. 「停止」をクリックして、Caching Proxy サービスを停止します。

プロキシがサービスとしてインストールされていない場合は、Caching Proxy を停止するために次のいずれかを行います。

- 右上隅の「x」アイコンをクリックして閉じます。
- 「ファイル」メニューから、「終了」をクリックします。
- **Alt + F4** を押します。

---

## 構成変更後の再始動

サーバーの構成を（「構成および管理」フォームを使用するか、または `ibmproxy.conf` ファイルを編集して）変更した後では、サーバーを再始動しなければその変更は有効となりません。ほとんどの場合、最初にサーバーを停止せずに再始動することができますが、再始動するだけではリフレッシュされない設定もあります。詳しくは、185 ページの表 6 を参照してください。

サーバーを最初に停止しないで再始動するには、任意の「構成および管理」フォームの「再始動」ボタンをクリックするか、またはコマンド `ibmproxy -restart` を入力します。

---

## 第 2 部 Caching Proxy プロセスの構成および調整

ここでは、Caching Proxy コンポーネントがオペレーティング・システム、コンピューター・ハードウェア、およびネットワークと対話する仕組みについて説明します。また、この対話を構成する手順についても説明します。プロキシ・サーバーを構成するこれらの要素は、通常、システム管理者が管理し、使用可能メモリーや CPU サイクルなどのシステム・リソースと同様、IP アドレスやホスト名などのネットワーク・リソースと入念に整合性をとる必要があります。

ここには、次の章が含まれます。

25 ページの『第 6 章 サーバーの定義』

29 ページの『第 7 章 プロセス所有権の確立』

31 ページの『第 8 章 接続の管理』

35 ページの『第 9 章 プロキシ・サーバー・プロセスの調整』





---

## 第 6 章 サーバーの定義

一般に Caching Proxy は、ネットワーク・サーバーとして機能するように構成されたホスト・コンピューター・システムにおいて、バックグラウンド・プロセスとして稼働します。このプロセスは、ホスト・コンピューター・システム上のアクティブな 1 つまたはすべてのインターネット・プロトコル (IP) アドレスに関連付けられて (バインドされて) います。プロセスは指定されたポートで FTP や HTTP などのさまざまなインターネット・プロトコルを listen し、動作構成に従ってこれらの要求についてアクションを実行します。(詳細については、39 ページの『第 3 部 Caching Proxy の動作の構成』を参照してください。)

デフォルトでは、Caching Proxy はホスト・コンピューター・システムの名前を想定しています。このデフォルトの動作は、プロキシ・サーバーのホスト名を故意に指定することにより、上書きできます。Caching Proxy を特定の IP アドレスにバインドするためには、プロキシ・サーバーのホスト名をその IP アドレスと同じに変更する必要があります。

**注:** プロキシ・サーバーが IP アドレスにバインドしようとする時にホスト名が使用可能な IP アドレスに設定されていない場合、バインドは失敗し、プロキシ・サーバーは使用できるすべての IP アドレスを listen します。

プロキシ・サーバーのホスト名は、クライアント・トラフィックの解決方法には影響しません。プロキシ・サーバーでは自身のホスト名を HTTP 要求のヘッダーにあるホスト名引き数の値と比較しません。プロキシ・サーバーのホスト名は、エラー・メッセージなど、動的に生成されたローカルなコンテンツ・ページに取り込まれる場合もあります。また、HTTP ヘッダーにある Via 引き数の値として要求クライアントに返されることもあります。

宛先サーバーに要求を引き渡す前に、要求元クライアントのホスト名をプロキシ・サーバーのホスト名で置換するようにプロキシ・サーバーを構成することができます。置換するように構成すると、宛先サーバー側ではクライアントとの直接接続が確立されずに、強制的にプロキシ・サーバーを経由した通信チャネルが維持されます。

プロキシ・サーバー・プロセスを定義するには、ホスト・コンピューター・システムにあるプロキシ・サーバー・ファイルの実際のロケーションとプロキシ・サーバー自身を表す名前、ServerRoot、Hostname、Port の各ディレクティブの値として listen するポートをそれぞれ指定します。ホスト側に複数の IP アドレスがある場合にプロキシ・サーバーを特定のアドレスにバインドするには、BindSpecific ディレクティブの値を On に設定し、Hostname ディレクティブの値をその IP アドレスと同じになるように設定します。

管理ポートは、「構成および管理」フォームへのアクセス・メソッドおよびサーバーの保守メソッドを提供します。管理ポート経由でプロキシ・サーバーにアクセスできるようにするには、AdminPort ディレクティブに値を指定します。管理ポートで受信した要求は、標準ポートで受信した要求のキューには入りません。ただ

し、マッピング・ルールを記述してこのポートからの「構成および管理」フォームへのアクセスを許可することができます。

`BindSpecific` ディレクティブが使用可能にされると、`Caching Proxy` は `Port` ディレクティブで指定されたポートと、`Hostname` ディレクティブの値から導き出した IP アドレスとにバインドされます。`AdminPort` ディレクティブに指定されたポートはシステムで利用可能なすべての IP アドレスにバインドされます。

`IBM-PROXY` または `IBM_HTTP_SERVER` など、実行中サーバーのデフォルト名を上書きするには、`HeaderServerName` ディレクティブに値を指定します。この値によって HTTP 応答サーバー・フィールドが設定されます。

プロキシ・パフォーマンスを向上させるには、`PureProxy` ディレクティブの値を `on` に設定します。これによって、すべてのキャッシング機能が使用不可になります。

---

## 関連するディレクティブ

以下のディレクティブでプロキシ・サーバー・プロセスを定義します。

- 239 ページの『`Hostname` - サーバーの完全修飾ドメイン・ネームまたは IP アドレスを指定する』
- 298 ページの『`ServerRoot` - サーバー・プログラムのインストール先ディレクトリを指定する』
- 238 ページの『`HeaderServerName` - HTTP ヘッダーに戻されるプロキシ・サーバーの名前を指定する』
- 200 ページの『`BindSpecific` - サーバーが 1 つまたはすべての IP アドレスのどちらにバインドするかを指定する』
- 271 ページの『`Port` - サーバーが要求を `listen` するポートを指定する』
- 196 ページの『`AdminPort` - 管理ページまたはフォームを要求するためのポートを指定する』
- 287 ページの『`PureProxy` - 専用プロキシを使用不可にする』

詳細については、13 ページの『第 4 章 `ibmproxy.conf` ファイルの手作業での編集』を参照してください。

---

## 「構成および管理」フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「基本設定」->「ホスト名」
- 「サーバー構成」->「基本設定」->「サーバー・ルート」
- 「サーバー構成」->「基本設定」->「デフォルト・ポート番号」
- 「サーバー構成」->「基本設定」->「管理者ポート番号」
- 「サーバー構成」->「基本設定」->「バインド・オプション」
- 「プロキシ構成」->「プロキシ・パフォーマンス」->「ピュア・プロキシとして実行」

注: 「構成および管理」フォームを使用して、HeaderServerName ディレクティブを編集することはできません。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。



---

## 第 7 章 プロセス所有権の確立

スーパーユーザー root 以外のユーザーが Caching Proxy を始動すると、そのユーザーがプロキシ・サーバーに関連するプロセスすべての所有権を維持します。ただし、スーパーユーザー root が Caching Proxy を始動すると、プロキシ・サーバー内に設定されたユーザー ID 機能によって ibmproxy.conf ファイルの UserId および GroupId ディレクティブが読み取られ、プロセス所有権は指定のユーザーおよびグループに変更されます。これは、ファイル・アクセスを制限し、コンピューター・システムを保護するために実施されます。UserId または GroupId ディレクティブを変更する場合は、ログ・ディレクトリーと、プロキシ・サーバーで使用するアクセス制御リスト (ACL) などのその他のファイルについて、所有権およびアクセス権を更新する必要があります。

プロキシ・サーバー・プロセスの所有権を確立するには、UserID、GroupID、および PidFile ディレクティブの値としてそのプロセス ID が記録されているユーザー ID、グループ識別、およびファイルの位置を指定します。

強制的に プロキシ・サーバー・プロセスをフォアグラウンド・プロセスとして実行するには、NoBG ディレクティブの値を on に設定します。

### Linux システムの場合:

Linux システムの場合は、接続を listen する役割のあるプロセスとスレッドについてのみ、所有権を変更します。ワークフロー内の他のアクティビティーを管理するプロセスおよびスレッドは、依然として root が所有します。すべてのプロセスとスレッドで、プロセス ID (PID) 番号を受け取ります。ps コマンドを使用すると、関連付けられているのがプロセスかスレッドかに関係なく、すべてのプロセス ID がリストされます。

注: いくつかの Linux カーネルの場合、Caching Proxy が、エラー・ログに以下のエラー・メッセージを生成することがあります。

```
Cannot init groups for user nobody, errno: 1
```

そのエラー・メッセージを無視することができます。Caching Proxy の通常操作に何の影響もないからです。Caching Proxy を開始する前に、以下の環境変数をエクスポートすることによって、エラー・メッセージを回避する次善策もあります。

```
export RPM_FORCE_NPTL=1
export LD_ASSUME_KERNEL=2.4.19:
```

---

### 関連するディレクティブ

以下のディレクティブでプロキシ・サーバー・プロセスの所有権を確立します。

- 311 ページの『UserId - デフォルトのユーザー ID を指定する』
- 238 ページの『GroupId - グループ ID を指定する』
- 261 ページの『NoBG - Caching Proxy プロセスをフォアグラウンドで実行する』

- 269 ページの『PidFile (Linux および UNIX 専用) - Caching Proxy のプロセス ID を保管するファイルを指定する』

詳しくは、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

---

## 「構成および管理」フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「基本設定」->「ユーザー ID」
- 「サーバー構成」->「基本設定」->「グループ ID」
- 「サーバー構成」->「基本設定」->「プロセス ID ファイル位置」

注: 「構成および管理」フォームを使用して、NoBG ディレクティブを編集することはできません。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。

## 第 8 章 接続の管理

Caching Proxy は新規スレッドを作成して各クライアント要求を処理します。使用可能なスレッドがない場合、プロキシ・サーバーは、使用可能なスレッドが出てくるまで要求を保留します。アクティブ・スレッドの数が増加するにつれて、プロキシ・サーバーはより多くのメモリーを消費します。アクティブ・スレッドの最大数を `MaxActiveThreads` ディレクティブの値として指定してください。

`listen` バックログは、サーバーが新規クライアントとの接続を拒否するまでに記録する、クライアント接続の保留要求数です。この設定は、サーバーが数秒間で処理できる要求の数に基づいて行います。クライアントの接続がタイムアウトになる前に、サーバーが接続に応答することが必要です。バックログに記録できる最大の接続数を `ListenBacklog` ディレクティブの値として指定してください。

プロキシ・サーバーにより、持続クライアント/サーバーの接続が維持されます。持続接続により、サーバーはクライアントから多重要求を受け入れ、同じ TCP/IP 接続により応答を送信します。持続接続機能を使用すると、クライアントの待機時間が短縮され、プロキシ・サーバー上の CPU の負荷が軽減されますが、低コストではあるものの、サーバー・メモリーはわずかに増加します。それぞれの要求や応答ごとに個別に TCP/IP 接続をサーバーが確立しなくてもよいときには、全体的なスループットが向上し、接続が持続的な場合は、TCP/IP 接続は最大限の効率で使用されます。

サーバー側接続プールによって、プロキシ・サーバーと起点サーバーとの間の既存の接続を再使用することにより、サーバー側で持続接続機能を活用できます。接続が再使用されるたびに、3 つの TCP パケット (接続をセットアップするための 2 つのスリーウェイ・ハンドシェイク・パケットとそのクローズ用に 1 つ) が保管されます。サーバー側接続プールの利点は、以下のとおりです。

- ネットワーク輻輳が少ない (接続のオープンとクローズを最小化することによって)
- ルーター、クライアント、およびサーバーで使用される CPU 時間が少ない
- クライアントとサーバーで使用されるメモリー量が少ない
- キャッシュ・ミスでは、プロキシの応答が (接続のオープンおよびクローズを回避することによって) より速い

**注:** 接続プールは、制御された環境でのみ使用することをお勧めします。起点サーバーが HTTP 1.1 に準拠していない場合には、パフォーマンスが低下する恐れがあります。起点サーバーが正しくセットアップされていることが重要であるということにも注意してください。次は、Apache 1.3.19 構成ファイルからの簡単な例です。

- `#KeepAlive`: 持続接続 (`#connection` 当たり複数の要求) を許可するかどうか。非活動にするにはオフに設定します `#`
- `KeepAlive On`

- `#MaxKeepAliveRequests`: 持続接続中に許可する要求の最大数。数に制限を設けない場合には、0 に設定します。最高のパフォーマンスを得るには、この数値を高くします #
- `MaxKeepAliveRequests` 0
- `#KeepAliveTimeout`: 同じ接続で同じクライアントからの次の要求を待機する秒数 #
- `KeepAliveTimeout` 240

こうした設定では、使用している限り Web サーバーへの接続がオープンされていて、起点サーバーでなくプロキシが接続を管理できるようになります。したがって、必要になる範囲に対してのみ接続がプールされます。

サーバー側の接続プールが使用可能になっていると、起点サーバーへの HTTP 接続がプールされます。プロキシの `SSLEnable` ディレクティブが `on` に設定されている構成では、SSL 接続もプールされます。

1 回にサーバー当たりで保持するアイドル・ソケットの最大数、アイドル状態の持続接続を終了するまでサーバーが待機する時間、およびガーベッジ・コレクション・スレッドが接続のタイムアウトを検査する時間間隔 (デフォルトは 2 分間) を指定して、接続プールが維持される方法を構成します。

各種接続がオープンした状態の時間を、`InputTimeout`、`OutputTimeout`、`PersistTimeout`、`ReadTimeout`、および `ScriptTimeout` の各ディレクティブの値として定義します。

---

## 関連するディレクティブ

以下のディレクティブで プロキシ・サーバー・プロセスとの接続を管理します。

- 256 ページの『`MaxActiveThreads` - アクティブ・スレッドの最大数を指定する』
- 216 ページの『`ConnThreads` - 接続管理に使用する接続スレッドの数を指定』
- 249 ページの『`ListenBacklog` - サーバーが持てる `listen` バックログ・クライアント接続の数を指定する』
- 285 ページの『`ProxyPersistence` - 持続接続機能を許可する』
- 257 ページの『`MaxPersistRequest` - 持続接続で受信する要求の最大数を指定する』
- 297 ページの『`ServerConnPool` - 起点サーバーへの接続のプールを指定する』
- 258 ページの『`MaxSocketPerServer` - サーバーのオープン・アイドル状態のソケットの最大数を指定する』
- 297 ページの『`ServerConnTimeout` - 最大活動停止中期間を指定する』
- 296 ページの『`ServerConnGCRun` - ガーベッジ・コレクション・スレッドを実行する間隔を指定する』
- 269 ページの『`PersistTimeout` - クライアントが別の要求を送信するのを待機する時間を指定する』
- 244 ページの『`InputTimeout` - 入力タイムアウトを指定する』
- 289 ページの『`ReadTimeout` - 接続の時間制限を指定する』
- 266 ページの『`OutputTimeout` - 出力タイムアウトを指定する』



- 295 ページの『ScriptTimeout - スクリプトのタイムアウト設定値を指定する』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

---

## 「構成および管理」フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「システム管理」->「パフォーマンス」->「最大アクティブ・スレッド数」
- 「サーバー構成」->「システム管理」->「パフォーマンス」->「listen バックログのサイズ」
- 「プロキシー構成」->「プロキシー・パフォーマンス」->「持続接続を許可」
- 「サーバー構成」->「システム管理」->「パフォーマンス」->「最大要求数」
- 「サーバー構成」->「システム管理」->「パフォーマンス」->「持続タイムアウト」
- 「サーバー構成」->「システム管理」->「タイムアウト」->「入力タイムアウト」
- 「サーバー構成」->「システム管理」->「タイムアウト」->「読み取りタイムアウト」
- 「サーバー構成」->「システム管理」->「タイムアウト」->「出力タイムアウト」
- 「サーバー構成」->「システム管理」->「タイムアウト」->「スクリプト・タイムアウト」
- 「サーバー構成」->「システム管理」->「タイムアウト」->「持続タイムアウト」

注:

1. 「構成および管理」フォームを使用して、ServerConnPool、MaxsocketPerServer、ServerConnTimeout、ServerConnGCRun の各ディレクティブを編集することはできません。
2. PersistTimeout は、「サーバー構成」->「システム管理」->「パフォーマンス」フォーム、または「サーバー構成」->「システム管理」->「タイムアウト」フォームから編集できます。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。



---

## 第 9 章 プロキシ・サーバー・プロセスの調整

システムの適切なセットアップと調整によって、Caching Proxy のパフォーマンスを著しく向上させることができます。セットアップおよび調整の改善については、以下の説明を参考にしてください。

---

### パフォーマンスに関連するディレクティブの設定

以下のディレクティブはプロキシ・サーバー・プロセスのパフォーマンスに大きく影響します。

- 287 ページの『PureProxy - 専用プロキシを使用不可にする』。この機能は、キャッシングを完全に使用不可にすることにより、システム・パフォーマンスを向上させます。
- 285 ページの『ProxyPersistence - 持続接続機能を許可する』。この機能によりクライアントおよびサーバーはオープン接続を維持できます。持続接続では、プロキシ・サーバーからの文書の要求に対するラグ時間は減少しますが、より多くのネットワーク帯域幅および接続ごとの専用サーバー・スレッドが必要となります。セットアップによって使用可能なスレッドの数が制限されている場合は、持続接続を許可することはできません。

以下の「構成および管理」フォーム・フィールドを使用して関連するディレクティブの値を編集します。

- 「プロキシ構成」->「プロキシ・パフォーマンス」:「ピュア・プロキシとして実行」
- 「プロキシ構成」->「プロキシ・パフォーマンス」:「持続接続を許可」

---

### 他のアプリケーションの調査

システム上で実行されているサービスまたはデーモンを調べ、使用可能メモリの増加や CPU サイクルの向上に必要なものを除去します。例えば、Web ページをわずかしき提供しない Web サーバーをシステムで稼働している場合、Caching Proxy を唯一の Web サーバーとして使用することを検討してみてください。次の手順を実行して、他の Web サーバーを使用不可にします。

- AIX の場合は、/etc/inittab を調べます。
- Linux の場合は、/etc/rc.d/rcx.d を調べて、システムのデフォルトの実行レベル（通常は 2）を確認します。
- HP-UX および Solaris の場合は、/etc/rcx.d を調べて、システムのデフォルト実行レベル（通常は 2）を確認します。
- Windows システムの場合:
  1. 「スタート」->「設定 (Windows 2000 の場合)」->「コントロール パネル」->「管理ツール」->「サービス」をクリックします。
  2. 必要ではないのに、「自動」に設定されているサービスを調べます。

3. このようなサービスの「スタートアップの種類」を、「自動」から「手動」に変更します。

---

## ページング・スペースの検査

適切なオペレーションに必要なだけのページング・スペースが、使用しているシステムにあることを確認してください。システムには、物理メモリーの 2 倍のページング・スペースがなければなりません。可能なら、ページング・スペースを複数の物理ドライブに分散してください。例えば、512 MB のメモリーと 5 台の SCSI ドライブを装備した Netfinity 5000 サーバーには、ドライブごとに約 200 MB ずつ、合計 1 GB のページング・スペースが必要です。

---

## ファイル・システムの調整

Caching Proxy は、オペレーション中に多数のファイルを作成したり破棄したりします。プロキシ・サーバーがアクセスを記録する場合 (アクセス・ログ、プロキシ・アクセス・ログ、またはキャッシュ・アクセス・ログを使用して)、そのログが予想外に大きくなったときに、別の機能 (例えば、キャッシュ) のためのスペースを使用しないよう、ログ固有のファイル・システムにそのログを送る必要があります。

---

## TCP/IP 構成の調整

Caching Proxy は、TCP/IP 構成の変更に敏感です。いずれかのオペレーティング・システムで TCP/IP 値を下げると、予期しない方法でプロキシ・サーバーが実行される原因となる場合があります。とりわけ、TCP/IP 値の設定が低すぎると、プロキシ・サーバーに接続されたクライアントによって、またはプロキシが接続される発信元サーバーによって、接続がリセットされる場合があります。これは特に、低帯域幅の接続 (56700 bps 以下) を介してプロキシ・サーバーに接続されるクライアントの場合に当てはまります。TCP/IP パラメーターを下げる必要がある場合は、注意して進めてください。

---

## 強度のロード環境 (HP-UX、Linux、Solaris、Windows) の場合の TCP 時間待ち間隔の調整

TCP 時間待ち間隔は、強制的にクローズするまでに、ソケットが送信側からの FIN パケットを待つ、時間の長さを指定します。強度のロード環境においては、大量のソケットが、その接続がクローズされた後も TIME\_WAIT 状態で保留のままになっている場合、プロキシ・サーバーが停止しているように見える場合があります。TCP 時間待ち間隔を減らすことで、保留にされるソケットの数が削減され、強度のロード環境では、プロキシ・サーバーが停止しているように見えるのを防止することができます。この間隔は 5 秒に設定することをお勧めします。

TCP 時間待ち間隔を 5 秒に設定するには、以下のようにします。

- HP-UX の場合:

以下のコマンドを実行します。

```
ndd /dev/tcp -set tcp_time_wait_interval 5000
```

「SAM」ユーティリティを使用し、カーネル・パラメーター `max_thread_proc` を少なくとも 2048 に設定します。

注: また、カーネル・パラメーター (`maxfiles`、`maxfiles_lim`、`maxproc`、`shmem`、`tcp_conn_request_max`、`tcp_ip_abort_interval`、`tcp_keepalive_interval`、`tcp_rexmit_interval_initial`、`tcp_rexmit_interval_max`、`tcp_rexmit_interval_min`、`tcp_xmit_hiwater_def`、`tcp_recv_hiwater_def`) も調整します。

- Linux の場合:

以下のコマンドを実行します。

```
echo "1024 61000" > /proc/sys/net/ipv4/ip_local_port_range
echo "5" > /proc/sys/net/ipv4/tcp_fin_timeout
```

- Solaris の場合:

以下のコマンドを実行します。

```
ndd /dev/tcp -set tcp_time_wait_interval 5000
```

`/etc/system` ファイルを編集して、以下のようになります。

```
set tcp:tcp_conn_hash_size=8129
```

- Windows の場合:

TCP 時間待ち間隔をセットする、レジストリー項目を作成する必要があります。詳しくは、Windows の資料を参照してください。

---

## Linux カーネルの調整

Linux カーネルの限界値の中には低いものがあり、変更することができます。 `/proc` ファイル・システムを介して変更されるものもあれば、カーネルの再コンパイルが必要なものもあります。

注: `/proc` ファイル・システムは仮想ファイル・システムです。つまり、ディスク上に実在するものではありません。 実在はしませんが、Linux カーネルへのインターフェースの役目を果たします。 実在しないので、リスタート時に入力値は失われます。したがって、`/proc` ファイル・システムに加える変更は、RedHat の `/etc/rc.d/rc.local` ファイルまたは SUSE の `/etc/rc.config` ファイルに入れてください。そうすると、リスタート時に必ず変更が有効になります。

次のことをお勧めします。

- ファイル記述子の最大値は、デフォルトでは 4096 です。 `rc.local` ファイルに次の行を追加すると、この値を変更できます。

```
echo 32768 > /proc/sys/fs/file-max
```

- inode の最大値は、デフォルトでは 16384 です。 `rc.local` ファイルに次の行を追加すると、この値を変更できます。

```
echo 65536 > /proc/sys/fs/inode-max
```

- TCP および UDP のポート範囲は、デフォルトでは 1024 ~ 4999 です。 `rc.local` ファイルに次の行を追加すると、この値を 32768 ~ 61000 に変更できます。

```
echo 32768 61000 > /proc/sys/net/ipv4/ip_local_port_range
```

- デフォルトでは、許可されるタスクの数は 512 です。多数のタスクが実行中の場合、これでは、処理スレッドの最大数に影響が出ます。この制限は、`YourKernelSource/include/linux/tasks.h` の `NR_TASKS` の値を変更して、2048 に増やすことができます。
- さらに、`MIN_TASKS_LEFT_FOR_ROOT` の値を 24 に変更します。この変更を有効にするには、カーネルを再コンパイルする必要があります。

カーネルの再作成を決めた場合は、確実に必要なオプションだけを使用可能にしてください。必要のないデーモンは、実行しないでください。

---

## AIX スレッド・チューニング変数の調整

AIX システムでは、システム・スコープ・スレッドを使用して、スレッドが複数のヒープを使用することを許可すると、Caching Proxy パフォーマンスを向上させることができます。パフォーマンスは、オペレーティング・システムのマルチプロセッシング機能、および根本的なオペレーティング・システムのスレッド・スケジューリングと関連しています。AIX スレッド・チューニング変数を次のように設定すると、パフォーマンスを改善することができます。

```
export AIXTHREAD_SCOPE=S
export SPINLOOPTIME=500
export YIELDLOOPTIME=100
export MALLOCMULTIHEAP=1
```

これらの環境変数は、`/usr/sbin/ibmproxy` を開始する前に設定するか、または **startsrc -s ibmproxy** を使用してプロキシ・サーバーを開始する場合には、`/etc/rc.ibmproxy` に追加することができます。これらのスレッド環境変数を調整すると、SMP システムにおけるパフォーマンスの向上がさらに顕著になります。ただし、単一プロセッサ・システム上でもパフォーマンスの向上が明白になる場合があります。

注: スレッド・チューニング変数の詳細については、ご使用の AIX オペレーティング・システムに付属の資料を参照してください。

---

## 第 3 部 Caching Proxy の動作の構成

ここでは、Caching Proxy コンポーネントがクライアントの要求に応える仕組みと、この動作の構成手順について説明します。これらのプロキシ・サーバー構成要素は、通常、Web 管理者が管理しており、同じネットワーク内のホスト・コンピューター・システム上でも、他のコンピューター・システム上でもその他のプロセスには影響しません。

ここには、次の章が含まれます。

41 ページの『第 10 章 要求処理の管理』

53 ページの『第 11 章 ローカル・コンテンツ送達の管理』

57 ページの『第 12 章 FTP 接続の管理』

61 ページの『第 13 章 サーバー処理のカスタマイズ』

71 ページの『第 14 章 ヘッダー・オプションの構成』

73 ページの『第 15 章 アプリケーション・プログラミング・インターフェースについて』





---

## 第 10 章 要求処理の管理

Caching Proxy はクライアント要求を受信すると、要求されたメソッドが使用可能であれば、URL フィールドに指定されたオブジェクト上のメソッド・フィールドで指定されたアクションを実行します。プロキシー・サーバーは管理者が定義したマッピング・ルール・セットに従って URL を変換します。これらのマッピング・ルールによって、Caching Proxy を Web サーバーとして機能させ、ローカル・ファイル・システムのオブジェクトを検索するように指示したり、あるいはプロキシー・サーバーとして機能させ、起点サーバーのオブジェクトを検索するように指示したりできます。

この章では、メソッドを使用可能にする方法、マッピング・ルールを定義する方法、および代理プロキシー・サーバーを構成する方法について説明します。

---

### HTTP/FTP メソッドを使用可能にする

クライアントがサーバーに送信する要求には、指定されたオブジェクトでサーバーが実行するアクションを示すメソッド・フィールドが含まれています。

以下に、プロキシー・サーバーがサポートするメソッドのリストを示し、メソッドが使用可能にされている場合にそのメソッドを含むクライアント要求に対して、どのように応答するかについて説明します。

**注:** メソッドの中には HTTP の場合でも FTP 要求の場合でも同じものがあります。これらのメソッドを HTTP に対して使用可能にすると、FTP に対しても使用可能になります。

#### CONNECT

CONNECT メソッドを使用すると、要求および応答をプロキシー・サーバー経由でトンネルすることができます。これは、フォワード・プロキシー構成にのみ適用されます。

Enable CONNECT メソッドのフォーマットおよび使用可能なオプションについては、128 ページの『SSL トンネリングの構成』を参照してください。

#### DELETE

プロキシー・サーバーが、URL によって識別されたオブジェクトを削除します。DELETE によってクライアントは Caching Proxy からファイルを消去することができます。サーバー保護セットアップを使用して、だれがどのファイルに DELETE を使用できるかを定義します。詳しくは、121 ページの『第 25 章 サーバー保護セットアップ』を参照してください。

**GET** プロキシー・サーバーは、URL によって識別されるすべてのデータを戻します。URL が実行可能なプログラムを参照すると、プロキシーはそのプログラムの出力を戻します。このメソッドは、持続接続の間、取り扱うことができます。

**HEAD** プロキシー・サーバーは URL によって識別される HTTP 文書ヘッダーだけを戻し、文書本文は戻しません。

## OPTIONS

プロキシ・サーバーは、URL によって識別される要求応答チェーンの通信オプションに関する情報を戻します。このメソッドを使用すると、クライアントは、オブジェクトを処理したり検索しなくても、そのオブジェクトと関連したオプションと要件、またはサーバーの能力を判別することができます。

**POST** 要求にはデータおよび URL が含まれます。プロキシ・サーバーは、要求に含まれているデータについて、そのデータを処理する URL で識別されたリソースの新しい従属データとして受け入れます。リソースとしては、データを受け入れるプログラム、他のプロトコルなどへのゲートウェイ、または注釈を受け入れる別個のプログラムが考えられます。

POST メソッドは、既存のリソースの注釈を扱うように設計されています。例には、掲示板、ニュースグループ、メーリング・リスト、または同様なリソースのグループに対するメッセージの送付、データ・ブロックの提供 (例えば、フォームからデータ処理プログラムへ)、あるいは付加操作によるデータベースの拡張が含まれます。Caching Proxy では、POST メソッドは「構成および管理」フォームを処理するために使用されます。

このメソッドは、持続接続の間、取り扱うことができます。

**PUT** 要求にはデータおよび URL が含まれます。プロキシ・サーバーは URL で識別されたリソースにデータを保管します。リソースがすでに存在していれば、PUT はそれを要求に含まれるデータで置き換えます。リソースが存在しない場合には、PUT はそれを作成して、要求に含まれるデータをそれに移植します。このメソッドは、持続接続の間、取り扱うことができます。

PUT メソッドを使用可能にすると、HTTP および FTP を使用して、Caching Proxy にファイルを書き込むことができます。PUT によってクライアントが Caching Proxy に書き込むことができるため、サーバー保護設定を使用して、PUT を使用できる担当者および PUT を使用できるファイルを定義する必要があります。(121 ページの『第 25 章 サーバー保護セットアップ』を参照してください)

## TRACE

プロキシ・サーバーは、クライアントが送信した要求メッセージをエコーします。このメソッドを使用すると、クライアントは、要求チェーンの他方の端で受信されているデータを見ることができ、そのデータをテストや診断に使用することができます。プロキシ応答のコンテンツ・タイプは message/http です。

## 関連するディレクティブ

以下のディレクティブで HTTP/FTP メソッドを使用可能にします。

- 226 ページの『Enable - HTTP メソッドを使用可能にする』
- 224 ページの『Disable - HTTP メソッドを使用不可にする』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

## 構成および管理フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「GET」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「HEAD」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「POST」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「PUT」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「DELETE」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「OPTIONS」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「TRACE」
- 「サーバー構成」->「要求処理」->「HTTP メソッド」->「CONNECT」

注: POST メソッドを使用不可にすると、「構成および管理」フォームを使用して Caching Proxy を構成することはできなくなります。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。

---

## WebDAV メソッド、MS Exchange メソッド、およびユーザー定義のメソッドを使用可能にする

これは、リバース・プロキシ構成にのみ適用されます。

Caching Proxy では、標準 HTTP メソッドのサポートがあるほか、RFC で定義されたメソッドや、なんらかのアプリケーションで使われるメソッドなど、その他のメソッドの転送もサポートしています。また、Caching Proxy は、カスタマー定義のメソッドもサポートしており、プロキシ・サーバー経由でそれらのメソッドを転送することもできます。

Web-based Distributed Authoring and Versioning (WebDAV) は、HTTP プロトコルへの拡張セットです。この拡張セットを使用すると、リモート Web サーバー上でファイルの編集と管理を円滑に行うことができます。Caching Proxy は、WebDAV メソッド、Microsoft Exchange Server で使われるメソッド、およびユーザー定義 (カスタマイズ) メソッドをサポートしています。

これらのメソッドは、ハードコーディングされており、Enable ディレクティブと Disable ディレクティブによって管理します。管理者は、PROTECT ディレクティブで定義された、対応するメソッド・マスクを使用して、これらのメソッドの使用を許可することもできます。

サポートされる WebDAV メソッド (RFC 2518):

PROPFIND、PROPPATCH、MKCOL、COPY、MOVE、LOCK、UNLOCK、SEARCH

サポートされる MS Exchange メソッド:

BMOVE、BCOPY、BDELETE、BPROPFIND、BPROPPATCH、POLL、NOTIFY、SUBSCRIBE、UNSUBSCRIBE、ACL、SUBSCRIPTIONS、X\_MS\_ENUMATTS

WebDAV メソッドまたは MS Exchange Server メソッドが有効であると、Caching Proxy は、要求をターゲット・サーバーにのみ転送し、要求本体にリソース・リンクを再書き込みすることはありません。

また、Caching Proxy は、ユーザー定義のメソッドをバックエンド・サーバーに転送することもできます。ibmproxy.conf ファイルで Enable ディレクティブの以下の構文を使用すると、カスタマイズ・メソッドを使用可能にできます。

```
Enable user-defined-method [WithBody | WithoutBody]
```

WithBody または WithoutBody の値を設定して、ユーザー定義メソッドが要求本体を必要とするかどうかをプロキシに指示します。

次の例では、ユーザー定義メソッド My\_METHOD を有効にしており、そのメソッドが要求本体を必要とすることをプロキシに指示しています。

```
Enable MY_METHOD WithBody
```

## 関連するディレクティブ

以下のディレクティブで、WebDAV メソッド、MS Exchange メソッド、およびユーザー定義メソッドを有効にします。

- 226 ページの『Enable - HTTP メソッドを使用可能にする』
- 224 ページの『Disable - HTTP メソッドを使用不可にする』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

---

## マッピング・ルールの定義

マッピング・ルールはクライアント要求を、例えば起点サーバーに (代行して) 渡す、リダイレクトする、拒否するなど、何らかの方法で Caching Proxy に処理させる構成ディレクティブです。Caching Proxy を適切に機能させるためには、マッピング・ルールを正しく設定することが重要です。マッピング・ルールが影響を与えるのは、以下の機能です。

- 基本プロキシ機能
- ブラウザー・ベースの「構成および管理」フォームへのアクセス
- サーブレット結果および他の動的生成コンテンツのキャッシング機能

マッピング・ルール・ディレクティブは次のような形式を使用します。

```
rule template target [IP_address | host_name]:[port]
```

指定のテンプレートと IP ポートの組み合わせに一致する要求だけが、そのルールの対象となります。テンプレートには、例えば https://\*\*/\*.asp のように、ワイルドカードを入れることができます。

構成ファイルでルールが現れる順序は、重要です。Map ディレクティブを除いて、要求はテンプレートに一致するとすぐに処理され、後のルールは評価されません。Map ディレクティブは、要求にある URL を置き換えます。この新規の要求が、引き続き残りのマッピング・ルールと比較されます。

## マッピング・ルール

以下のマッピング・ルールは、指定のテンプレートと一致するクライアント要求に適用されます。

- **Map、MapQuery** — 要求を再書き込みします。Map ルールおよび MapQuery ルールは、要求 URL (テンプレート) を別の URL スtring (ターゲット) で置き換えます。この置換が行われた後、新しいStringの入った要求は引き続き、残りのマッピング・ルールと比較されます。
- **RuleCaseSense** — 大/小文字を区別しないアプリケーション URL からのマッピング要求を許可します。RuleCaseSense ディレクティブを off に調整すると、プロキシが大/小文字を区別することなく、ibmproxy.conf ファイルに定義されているルールに対する要求をマップできるようになります。
- **Pass、Exec** — 要求をローカルで処理する。Pass および Exec ルールは、要求をプロキシ・サーバーで処理します。Pass ルールは、要求 URL (テンプレート) をプロキシ・サーバーから提供されるファイル (ターゲット) にマップします。Exec ルールは、要求 URL をプロキシ・サーバーで実行される CGI プログラムにマップします。
- **Fail** — 要求を拒否する。Fail ルールは、プロキシ・サーバーで要求 (テンプレート) を拒否します。Fail ルールのテンプレートに一致する要求はすべて、それ以上処理されません。Fail ルールにはターゲット引き数がありません。
- **Redirect** — 要求を転送する。Redirect ルールは、要求 (テンプレート) を別の Web サーバー (ターゲット) に転送します。このルールのターゲットは通信プロトコルを含む完全な URL であるため、このリダイレクト中にプロトコルを変更することができます。例えば、SSL 暗号を HTTP 要求に追加できます。リダイレクトでは、要求を満たす前にキャッシュを検査しません。
- **Proxy、ProxyWAS** — 要求を代行する。Proxy および ProxyWAS ルールは、要求 (テンプレート) を別のサーバー (ターゲット) に引き渡します。単純な Redirect ルールとは異なり、Proxy ルールではプロキシ・サーバーが、キャッシュを検査して要求を満たし、起点サーバーからコンテンツをキャッシュに入れ、拡張機能を使用可能にする HTTP ヘッダーを書き込むことができるようにします。起点サーバーが WebSphere Application Server の場合は、Proxy ルールではなく ProxyWAS ルールを使用してください。

以下のマッピング・ルールは、起点サーバーの応答に適用されます。

- **ReversePass** — 自動的にリダイレクトされた要求をインターセプトする。ReversePass ルールは、起点サーバーからの応答がクライアント側に渡される途中でプロキシ・サーバーに渡されると、その応答をテンプレートと突き合わせます。ReversePass ディレクティブは、クライアントを直接起点サーバーに接続させるリダイレクト状況コードを検出するように設計されています。クライアントはターゲット引き数で定義されたサーバーに接続するよう指示されます。

以下のマッピング・ルールは、API アプリケーションに適用されます。

- **nameTrans** — 要求を受け入れ、要求処理の名前変換ステップ時に置換ファイル・パスで定義された API アプリケーションを実行する。
- **service** — 要求を受け入れ、要求処理のサービス・ステップ時に置換ファイル・パスで定義された API アプリケーションを実行する。



## サロゲート・サーバーの構成

標準サロゲートを構成するには、次のようにします。

- プロキシ・サーバー・ポートを 80 に設定する。

Port 80

- ポート 80 で受信した起点サーバーに対するすべての要求を代行するというプロキシ・ルールを、他のすべてのルールより前に追加する。

Proxy /\* http://our.content.server.com/\* :80

- 管理ポートを 80 以外のポートで使用可能にする。

AdminPort 8080

これにより、起点サーバーに対する、ポート 80 のすべての HTTP トラフィックが代行されます。管理ポートに入ったトラフィックは、初期のワイルドカード・プロキシ・ルールとは突き合わされないため、影響を受けません。要求は、残りのマッピング・ルールを使用して処理されます。

## 関連するディレクティブ

以下のディレクティブでマッピング・ルールを定義します。

- 253 ページの『Map - ルールの突き合わせを行うため、要求パス・ストリングを使用して、マッチング要求を新規要求ストリングに変更する』
- 254 ページの『MapQuery - ルールの突き合わせを行うため、要求パスおよび照会ストリングを使用して、マッチング要求を新規要求ストリングに変更する』
- 294 ページの『RuleCaseSense - 大/小文字を区別しないアプリケーション URL からの要求をマップする』
- 267 ページの『Pass - 要求を受け入れるためのテンプレートを指定する』
- 230 ページの『Exec - 一致する要求に対して CGI プログラムを実行する』
- 290 ページの『Redirect - 別のサーバーに送信される要求のテンプレートを指定する』
- 282 ページの『Proxy - プロキシ・プロトコルまたはリバース・プロキシを指定する』
- 287 ページの『ProxyWAS - 要求が WebSphere Application Server に送信されることを指定する』
- 292 ページの『ReversePass - 自動的にリダイレクトされた要求をインターセプトする』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

## 構成および管理フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「要求処理」->「要求経路指定」

注: 「構成および管理」フォームでは、ポート番号引き数をサポートしていません。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。

---

## ジャンクション再書き込みの使用可能化 (オプション)

これは、リバース・プロキシー構成にのみ適用されます。

**JunctionRewrite** ディレクティブにより、**Caching Proxy** 内のジャンクション再書き込みルーチンは、発信元サーバーからの応答を再書き込みし、ジャンクションが使用された場合にサーバーに関連した URL が適切な発信元サーバーにマップされるようにします。ジャンクション再書き込みプラグインも使用可能にする必要があります。ジャンクションは、プロキシー・マッピング・ルールによって定義されます。

プロキシー・マッピング規則を使用してジャンクションを定義するときは、**JunctionPrefix** オプションを使用する場合も、しない場合も、**Proxy** ディレクティブを使用できます。

### **JunctionPrefix** オプションを使用しないジャンクションの定義

以下は、ジャンクション再書き込みルーチンによって行われる、有効なジャンクションの例です。

- **Proxy /shop/\* http://shopserver.acme.com/\***
- **Proxy /auth/\* http://authserver.acme.com/\***

以下は、ジャンクション再書き込みルーチンが作用しない、有効なジャンクションの例です。

- **Proxy /\* http://defaultserver.acme.com/\***

以下は、有効なジャンクションの例です。

- **Proxy /images/\*.gif http://imageserver.acme.com/images/\*.gif**
- **Proxy /cgi-bin/\* http://cgiserver.acme.com/cgi/perl/\***

これらのマッピング・ルールでは、shopserver、authserver、および b2bserver 用のジャンクションが作成されています。該当する HTML タグ内に、以下の URL が含まれた HTML 文書を shopserver が戻すものとします。

- **/index.html** (サーバー相対参照)
- **/images/shop.gif** (サーバー相対参照)
- **buy/buy.jsp** (ディレクトリー相対参照)
- **http://ebay.com** (絶対参照)

ジャンクション再書き込みルーチンは、以下のように、プロキシー・マッピング・ルールを使用して、サーバー相対参照を再書き込みします。

- **/shop/index.html** (変更)
- **/shop/images/shop.gif** (変更)
- **buy/buy.jsp** (未変更)
- **http://ebay.com** (未変更)

## JunctionPrefix オプションを使用するジャンクションの定義 (推奨される方法)

JunctionPrefix オプションを Proxy ディレクティブと併用するときは、Proxy 規則内の最初の URL パターンから JunctionPrefix を割り出すようにする代わりに、次のフォーマットを用いて、Proxy 規則内でジャンクションの接頭部を宣言することができます。

```
Proxy url_pattern1 url_pattern2 JunctionPrefix:url_prefix
```

JunctionPrefix を使用しているときは、最初の URL パターンのフォーマットに制限はありません。JunctionPrefix オプションを使用していないときにジャンクションの再書き込みをサポートするには、プロキシ URL は次のフォーマットを持つ必要があります。Proxy /market/\* http://b2bserver/\*。しかし、JunctionPrefix の使用時は、次の Proxy 規則がジャンクション再書き込みにおいて有効です。

```
Proxy /market/partners/*.html http://b2bserver.acme.com/*.html  
junctionprefix:/market/partners
```

ジャンクション再書き込みルーチンは、以下のタグに影響を与えます。

表 3. ジャンクション再書き込みルーチンによって影響されるタグ

タグ	属性
!—	URL
a	href
applet	archive、codebase
area	href
base	href
body	background
del	cite
embed	pluginspage
form	action
input	src
frame	src、longdesc
iframe	src、longdesc
ilayer	src、background
img	src、usemap、lowsrc、longdesc、dynsrc
layer	src、background
link	href
meta	url
object	data、classid、codebase、codepage
script	src
table	background
td	background
th	background
tr	background



注: ジャンクション再書き込みルーチンは、JavaScript またはブラウザー内のプラグインによって生成されたタグには影響を与えません。

## 関連するディレクティブ

以下のディレクティブは、ジャンクション再書き込みルーチンおよびプラグインを使用可能にするために使用されます。

- 297 ページの『ServerInit - サーバー初期化ステップをカスタマイズする』
- 308 ページの『Transmogriifier - データ操作ステップをカスタマイズする』
- 245 ページの『JunctionRewrite - URL 再書き込みを使用可能にする』
- 246 ページの『JunctionRewriteSetCookiePath - JunctionRewrite プラグインとの併用時に Set-Cookie ヘッダーのパス・オプションを再書き込みする』
- 245 ページの『JunctionReplaceUrlPrefix - JunctionRewrite プラグインと併用時に、接頭部を挿入する代わりに URL を置き換える』
- 246 ページの『JunctionSkipUrlPrefix - JunctionRewrite プラグインと併用時に接頭部を既に含んでいる URL の再書き込みをスキップする』

詳細については、13 ページの『第 4 章 ibmpoxy.conf ファイルの手作業での編集』を参照してください。

## 構成および管理フォーム

以下の「構成および管理」フォームを使用して、ジャンクション再書き込みプラグインを使用可能にすることができます。

- 「サーバー構成」->「要求処理」->「API 要求処理」

注: 「構成および管理」フォームは、JunctionRewrite ディレクティブをサポートしていません。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。

## JunctionRewrite に代わる UseCookie

次のように、Cookie を使用して、バックエンド・サーバー情報を保管できます。Cookie は、クライアント・ブラウザーに送信されます。ブラウザーが HTML ページのリソースに要求を送信すると、Cookie が添付されるため、Caching Proxy は要求を正しいバックエンド・サーバーに転送します。

JunctionRewrite の代わりに Cookie を使用する場合は、ibmpoxy.conf ファイルを次のように変更します。

1. 「JunctionRewrite on」を「JunctionRewrite on UseCookie」に変更します。
2. JunctionRewrite プラグインをコメント化します。

以下は、JunctionRewrite プラグインと Cookie のインプリメンテーションの比較です。

- JunctionRewrite プラグイン
  - HTML ページが再書き込みされます。

- Transmogrifier プラグインを使用している場合を除いて、スクリプト言語とアプレットの再書き込みはサポートされません。『JunctionRewrite 機能を拡張する Transmogrifier プラグインのサンプル』を参照してください。
- パフォーマンスが低下します。
- バックエンド・サーバー構成には制限がありません。クライアントはセッション内のバックエンド・サーバーに相互アクセスできます。
- Cookie のインプリメンテーション
  - HTML ページは、再書き込みされません。Cookie はクライアントに送信されます。
  - クライアント・ブラウザは、Cookie をサポートしている必要があります。
  - パフォーマンスが向上します。
  - バックエンド・サーバー構成には、いくつかの制限があります。クライアントがセッション内でバックエンド・サーバーにアクセスする場合のみ使用できます。

注: JunctionRewrite を UseCookie オプションと併用する際に既知の制限があります。この組み合わせでは、Cookie がホストのサブディレクトリーの 1 つに対してのみ適用されるにも関わらず、すべての要求に対して誤った URL 変換が行われます。以下に、ROOT の下にありジャンクションを必要としない URL を正しく処理するための方法を 2 つ挙げます。

- ibmproxy.conf ファイルで、プロキシ規則を JunctionRewrite ディレクティブの前に置く。(JunctionRewrite ディレクティブの前にあるプロキシ規則はすべて、再書き込みされることはありません。)
- ワイルドカード (\*) を使用する代わりに、それぞれの URL を明示的にマップする。例えば、次のとおりです。

```
Proxy /no-junction.jpg http://login-server/no-junction.jpg
```

## JunctionRewrite 機能を拡張する Transmogrifier プラグインのサンプル

HTML ファイル内の JavaScript™ (SCRIPT) およびアプレット (APPLET) タグ・ブロックを再書き込みおよび構文解析する、カスタマイズ可能なサンプル・コードが提供されています。JunctionRewrite プラグインは、JavaScript 内または Java™ のパラメーター値へのリソース・リンクを単独で処理することはできません。

Caching Proxy をインストールした後は、同じコードをコンパイルおよび構成し、JunctionRewrite で実行することができます。

次のサンプル・ファイルは、フィックスパックをダウンロードしたディレクトリー内の ...samples/cp/ サブディレクトリーに配置されています。

- Makefile (このサンプル・プラグイン用の Makefile)
- junctionRewrite2.h (カスタマイズされたパーサー・ハンドラー用のインターフェース)
- junctionRewrite2.c (上記のインターフェース用のインプリメンテーション)
- scriptHandler.c (JavaScript 再書き込みハンドラーのサンプル)
- appletHandler.c (Applet ブロック・ハンドラーのサンプル)

- junctionRewrite2.def (Windows プラグインの def ファイル)
- junctionRewrite2.exp (Linux および UNIX プラグインの export ファイル)



---

## 第 11 章 ローカル・コンテンツ送達の管理

Pass および Exec マッピング・ルールは、ローカル・コンテンツを要求元クライアントに送達するために使用されます。デフォルトでは、ワイルドカード・テンプレート付きの Pass ルールが最後のマッピング・ルールとして指定されます。このルールによって、直前のテンプレートに一致しないすべての要求を対象に、通常は文書ルート・ディレクトリーを示すターゲット・ディレクトリーからファイルを検索するよう指示します。

ファイル名を含まない URL を受信した場合、Caching Proxy は指定ディレクトリーがある場合はそのディレクトリー、指定ディレクトリーがない場合は文書ルート・ディレクトリーを対象に、構成ファイルで指定されたウェルカム・ページのリストに一致するファイルがないかどうかを検索して、要求を処理します。ウェルカム・ページより多くのページが定義されている場合、プロキシー・サーバーは定義された順序でページを検索します。最初に検出されたウェルカム・ページが提供されます。

サーバー・ホーム・ページは、サーバーがディレクトリー名またはファイル名なしでサーバーの URL のみを含む要求を受け取るときにデフォルトで送達する Web ページです。前に説明したように、デフォルトのワイルドカード・マッピング・ルールでは、サーバー・ホーム・ページが文書ルート・ディレクトリーに保管されていること、およびホーム・ページのファイル名が定義されたウェルカム・ページと一致していることが必要です。

**注:** Web ブラウザーの中には、ブラウザーを開始すると、そのブラウザーがロードする最初のページに ホーム・ページ という用語を使用するものがあります。この文書でこの用語を使用する場合は、サーバー・ホーム・ページのみを示します。

この章では、文書ルート・ディレクトリーとウェルカム・ページを定義する方法について説明します。

---

### 文書ルート・ディレクトリーの定義

デフォルトの文書ルート・ディレクトリーは次のようになります。

- Linux および UNIX の場合: /opt/ibm/edge/cp/server\_root/pub/lang/
- Windows の場合: drive:¥Program Files¥IBM¥edge¥cp¥server\_root¥pub¥lang¥、またはインストール時に HTML ディレクトリーとして指定したディレクトリー

### 関連するディレクティブ

以下のディレクティブで文書ルート・ディレクトリーを定義します。

- 267 ページの『Pass - 要求を受け入れるためのテンプレートを指定する』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

## 「構成および管理」フォーム

「構成および管理」フォームで文書ルート・ディレクトリーを変更するには、次の手順を使用してください。

1. 「サーバー構成」->「要求処理」->「要求経路指定」を選択します。
2. 要求経路指定テーブルで、「要求テンプレート」列にストリング /\* が入っている行を見つけます。これは文書ルート・ディレクトリーを表します。テーブルの下「索引」ボックスで、その行の「索引」列の数字に対応する数字をクリックします。
3. 「置き換え」をクリックします。
4. 「アクション」ドロップダウン・リストで、「Pass」をクリックします。
5. 「URL 要求テンプレート」フィールドに /\* と入力します。
6. 「置換ファイル・パス」フィールドに、新規文書ルート・ディレクトリーを入力します。
7. 「実行依頼」をクリックします。
8. 変更が受け入れられた後に、上部のフレームにある「サーバー再始動」アイコン (I) をクリックします。

再始動後、サーバーは新規文書ルート・ディレクトリーの使用を開始します。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。

---

## デフォルト・ウェルカム・ページの定義

サーバーは、文書ルート・ディレクトリー内でホーム・ページを探しますが、そのサーバーが戻す特定のファイルはウェルカム・ページのリストで定義されます。

### ウェルカム・ページについて

サーバーは、ファイル名を指定しない URL 要求を受け取ると、そのサーバーの構成ファイルで設定されたウェルカム・ページのリストに従って、要求を満たそうとします。このリストは、デフォルトのホーム・ページとして使用するファイルを定義します。サーバーは、ウェルカム・ページのリストを文書ルート・ディレクトリー内のファイルに突き合わせて、ユーザーのホーム・ページを判別します。サーバーが検出する最初に一致するファイルが、ユーザーのホーム・ページとして戻されるファイルです。一致するファイルが見つからないと、サーバーは文書ルート・ディレクトリーのリストを表示します。

特定のファイルをサーバーのホーム・ページとして使用し、要求がディレクトリーまたはファイル名を指定しないときに戻るように設定するには、そのファイルを文書ルート・ディレクトリーに入れ、その名前がウェルカム・ページのリストにリスト表示されているいずれかのファイル名と一致するようにする必要があります。

デフォルトの構成ファイルは、次の順序で、ウェルカム・ページとして使用するこれらのファイル名を定義します。

1. welcome.html または welcome.htm
2. index.html または index.htm

### 3. Frntpage.html

サーバーは、リストにあるファイル名と一致する最初のファイルを戻します。ユーザーが `welcome.html` または `index.html` ファイルを作成し、そのファイルを文書ルート・ディレクトリーに入れるまで、サーバーは `Frntpage.html` をユーザーのホーム・ページとして使用します。

例えば、ユーザーがデフォルト構成を使用していて、文書ルート・ディレクトリーには `welcome.html` というファイルは含まれていないが、`index.html` と `FrntPage.html` というファイルが含まれている場合には、`index.html` ファイルをユーザーのホーム・ページとして使用します。

ホーム・ページが見つからない場合は、文書ルート・ディレクトリーの内容がディレクトリーとして表示されます。

## 関連するディレクティブ

以下のディレクティブでウェルカム・ページを定義します。

- 313 ページの『Welcome - ウェルカム・ファイルの名前を指定する』

詳細については、13 ページの『第 4 章 `ibmproxy.conf` ファイルの手作業での編集』を参照してください。

## 「構成および管理」フォーム

以下の「構成および管理」フォームでウェルカム・ページを定義します。

- 「サーバー構成」->「ディレクトリーおよびウェルカム・ページ」->「ウェルカム・ページ」

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。





---

## 第 12 章 FTP 接続の管理

これは、フォワード・プロキシー構成にのみ適用されます。

Caching Proxy は、適切な FTP サーバーに対する FTP URL の要求を代行しますが、FTP クライアントからの要求を代行するのには使用できません。これがサポートするのは、(ftp:// プロトコル方式を使用する) HTTP クライアントから受信した FTP 要求に限られます。

FTP ファイルの要求に対してサポートされるのは、GET、PUT、DELETE の各メソッドだけです。FTP ディレクトリー・リスト作成の要求に対しては、GET メソッドのみがサポートされます。デフォルトでは、PUT および DELETE は Caching Proxy では使用不可です。詳細については、41 ページの『HTTP/FTP メソッドを使用可能にする』を参照してください。

この章では、FTP ファイルを保護する方法と、FTP サーバー・ログイン、ディレクトリー・パス、チェーニングを管理する方法について説明します。

---

### FTP ファイルの保護

FTP ファイル・アップロードの PUT メソッド、または FTP ファイル削除の DELETE メソッドが使用可能になっている場合は、少なくとも PUT 要求と DELETE 要求に対する FTP プロキシー保護を定義して、FTP サーバーで許可されないファイル更新が行われるのを防ぐ必要があります。

FTP 要求のプロキシーを保護するには、「構成および管理」フォームで、「サーバー構成」->「文書保護」を選択します。FTP ファイル要求に保護セットアップを作成するには、要求テンプレートの先頭に ftp:// を入れます。例えば、exams というディレクトリー内のファイルを保護するには、テンプレート ftp://exams/\* を使用します。

保護セットアップ作成の詳細については、121 ページの『第 25 章 サーバー保護セットアップ』を参照してください。

---

### FTP サーバー・ログインの管理

要求 URL にユーザー ID もパスワードも指定されていない場合には、Caching Proxy は、要求された FTP サーバーに (ユーザー ID ANONYMOUS を使用して) 無名でログインしようとします。FTP サーバーの多くは、無名 FTP のパスワードとして、電子メール・アドレスを要求します。FTP サーバーが無名ログインに対するパスワードを要求する場合は、Caching Proxy は、構成ファイルの WebmasterEmail ディレクティブで指定する電子メール・アドレスを送信します。

「構成および管理」フォームで Webmaster 電子メール・アドレスを設定するには、「サーバー構成」->「システム管理」->「SNMP MIB」を選択します。また、電子メール・アドレスは、WebmasterEmail ディレクティブを使用することにより設定す

ることもできます。詳しくは、313 ページの『WebMasterEMail - 選ばれたサーバー報告書を受け取るための電子メール・アドレスを設定する』の該当するセクションを参照してください。

要求 URL の FTP サーバーが、ログイン時に特定のユーザー ID とパスワードを要求する場合は、ユーザーは要求された URL に、例えば次のようにユーザー ID とパスワードを入力することができます。

```
ftp://userid:password@ftpserverhost/
```

要求 URL で FTP ユーザー ID のパスワードを指定したくない場合、ユーザーは URL : `ftp://userid@ftpserverhost` にユーザー ID のみを入力することができます。Caching Proxy はまず、パスワードのない指定ユーザー ID を使用して、FTP サーバーにログインしようとします。パスワードのないログインがうまくいかない場合、ブラウザーはプロンプトを出して、指定ユーザー ID に関連したパスワードを入力するよう求めます。

無名ログイン以外のログインの場合、URL に少なくともユーザー ID を指定しなければなりません。ユーザー ID が指定されていない場合には、無名ログインが行われるので、クライアントにユーザー ID を求めるプロンプトが出されることはありません。

---

## FTP ディレクトリー・パスの管理

FTP URL のパス名をユーザーの作業ディレクトリーに関するものと解釈するか、ルート・ディレクトリーに関するものと解釈するかを Caching Proxy に対して指定しなければなりません。例えば、FTP サーバーにログインしたユーザーが `/export/home/user1` と呼ばれるデフォルトの作業ディレクトリーを持っていて、`test` と呼ばれるサブディレクトリーから `test1.exe` と呼ばれるファイルを検索したい場合には、プロキシは (FTP URL がどのように解釈されるかによって) 次の URL を使用して FTP サーバーからファイルを検索します。

- 絶対 パス名に設定されている場合:  
`ftp://user1:user1pw@FTPhost/export/home/user1/test/test1.exe`
- 相対 パス名に設定されている場合:  
`ftp://user1:user1pw@FTPhost/test/test1.exe`

相対 FTP URL パスが設定されている場合でも、ユーザーは最初のスラッシュ文字 (/) をルート・ディレクトリーを示す `%2F` で置き換えるという規則を使用して、絶対パス名を指定することができます。例えば、作業ディレクトリーが `/export/home/user1` の `user1` が `user2` の作業ディレクトリー `/export/home/user2` のファイルにアクセスしたい場合には、要求 `ftp://user1:user1pw@FTPhost/%2Fexport/home/user2/file` は、相対 FTP URL パス名が選択されていても、ルート・ディレクトリー / に対する URL として (すなわち、絶対パス名として) 正しく解釈されます。

FTP URL の解釈方法を指定するには、「構成および管理」フォームで、「プロキシ構成」→「プロキシ・パフォーマンス」を選択します。フォームの下側部分の、「FTP URL パス」の下で、サーバーのルート・ディレクトリーを指定する場合は「絶対パス」を、パスの開始としてユーザーの作業ディレクトリーを指定する場合は「相対パス」を選択するようにしてください。

この設定は、プロキシー構成ファイルでも変更することができます。詳しくは、235 ページの『FTPUrlPath - FTP URL をどう解釈するかを指定する』を参照してください。

---

## FTP チェーニングの管理

複数の Web プロキシー・サーバーを一緒にチェーニングしている場合は、FTP URL を含む要求が FTP サーバーに直接送信されずに、チェーニングされている Web プロキシー・サーバーに送信されるように指定することができます。FTP 要求に対してチェーニングされたプロキシー・サーバーを指定するには、「構成および管理」フォームで、「**プロキシー構成**」->「**プロキシー・チェーニングおよび非プロキシー・ドメイン**」を選択します。ftp:// プロトコル方式の要求をチェーニングしている場合でも、チェーニングされたプロキシーの URL を指定するために http:// プロトコル方式が使用されます。

プロキシー構成ファイルを使用して FTP チェーニングを構成するには、235 ページの『ftp\_proxy - FTP 要求のための別のプロキシー・サーバーを指定する』の解説セクションを参照してください。



---

## 第 13 章 サーバー処理のカスタマイズ

この章では、サーバー側インクルードを使用して、CGI プログラムおよびクライアントに引き渡される HTML 文書に情報を挿入する方法について説明します。サーバーのエラー・メッセージおよびリソース・マッピングのカスタマイズについても説明しています。

---

### サーバー側インクルード

サーバー側インクルードを使用すると、サーバーが起点サーバーとして機能する(つまり代理オブジェクトでもキャッシュされたオブジェクトでもない)場合に、サーバーが CGI プログラムおよびクライアントに送信する HTML 文書に情報を追加することができます。クライアントに送信できる情報の種類としては、現在日付、ファイルのサイズ、ファイルに加えられた最終変更日付などがあります。このセクションでは、サーバー側インクルードのためのコマンド形式について記述し、サーバー側インクルード・コマンドを CGI プログラムおよび HTML 文書で機能させる方法について説明します。サーバー側インクルードを使用して、エラー・ページをカスタマイズすることもできます。

### サーバー側インクルードに関する考慮事項

サーバー側インクルードをサーバーで使用する前に、パフォーマンス、セキュリティ、およびリスクなどの問題を考慮します。

- ファイルを送信しながらサーバーがそれを処理しているときは、パフォーマンスが大きく損なわれる可能性があります。
- サーバー上で一般のユーザーにコマンドを実行させる場合、セキュリティ面を妥協する必要があります。サーバー側インクルードを入れるディレクトリおよび **exec** コマンドを入れるディレクトリの決定は、慎重に行ってください。  
**exec** コマンドを使用可能にしなければ、セキュリティ・リスクを最小限に抑えることができます。
- サーバー側インクルードの使用によって、いくつかの問題が生じる場合があります。例えば、ファイルを再帰的に参照することはできません。ファイル `sleepy.html` を実行してプログラムが `<-- !#include file="sleepy.html" -->` を検出した場合には、サーバーはエラーを検出しないで失敗することがあります。(他のファイル内での非再帰的なファイル参照は問題ではありません)

### サーバー側インクルードの構成

サーバー側インクルードを使用可能にするには、「構成および管理」フォームで、「サーバー構成」->「基本設定」を選択します。このフォームを使用して、以下のサーバー側インクルード・タイプの中から受け入れるタイプを指定します。

- CGI スクリプト
- ファイル
- **exec** コマンドを使用する CGI スクリプト以外のすべて
- なし

また、このフォームを使用して、他のファイル・タイプに加えて、テキストまたは HTML 文書のサーバー側インクルード処理を実行するかどうかも指定します。

さらに、include に使用するファイル拡張子が認識されるようにします。そのためには、「構成および管理」フォームで、「サーバー構成」->「MIME タイプおよびエンコード」を選択し、「MIME タイプ」フォームを使用します。shtml および html の拡張子がデフォルトで認識されることに注意してください。

プロキシ構成ファイル内のディレクティブを編集してサーバー側インクルードのためにサーバーを構成するには、次のディレクティブのそれぞれの該当するセクションを参照してください。

- 194 ページの『AddType - 特定の接尾部を持つファイルのデータ・タイプを指定する』
- 242 ページの『imbeds - サーバー側インクルード処理が使用されるかどうかを指定する』

## サーバー側インクルードの形式

インクルード・コマンドは、コメントとして HTML 文書または CGI プログラムに組み込む必要があります。そのコマンドの形式は、以下のとおりです。

```
<!--#directive tag=value ... -->
または
<!--#directive tag="value" ... -->
```

値を囲む引用符は任意指定ですが、値にスペースが含まれる場合は必要です。

## サーバー側インクルードのディレクティブ

このセクションでは、サーバー側インクルードのためにサーバーによって受け入れられるディレクティブについて説明します。(これらのディレクティブを、プロキシ構成ファイルのディレクティブ (185 ページの『付録 B. 構成ファイル・ディレクティブ』で説明している) と混同しないようにしてください)

### config — ファイル処理を制御する

このディレクティブを使用して、ファイル処理のある局面を制御します。有効なタグは、cmntmsg、errmsg、sizefmt、および timefmt です。

#### cmntmsg

このタグを使用して、他のディレクティブにより追加されるコメントの先頭より前にくるメッセージを指定します。ディレクティブの指定と "-->" の間にテキストを含むディレクティブの場合、そのテキストがコメントとして扱われ、サーバーがクライアントに送信するファイルに追加されます。

例:

```
<!--#config cmntmsg="[This is a comment]" -->
<!-- #echo var=" " extra text -->
```

結果: <!--[This is a comment] extra text -->

デフォルト: [the following was extra in the directive]

**errmsg**

このタグを使用して、ファイルの処理中にエラーが起きた場合にクライアントに送信されるメッセージを指定します。 このメッセージは、サーバーのエラー・ログに記録されます。

例:

```
<!-- #config errmsg="[An error occurred]" -->
```

デフォルト: "[An error occurred while processing this directive]"

**sizefmt**

このタグを使用して、ファイル・サイズが表示されるとき形式を指定します。次の例では、bytes はバイト数を表示するために使用される値で、abbrev は K バイト数または M バイト数を表示するために使用される値です。

例 1:

```
<!--#config sizefmt=bytes -->
<!--#fsize file=foo.html -->
```

結果: 1024

例 2:

```
<!--#config sizefmt=abbrev -->
<!--#fsize file=foo.html -->
```

結果: 1K

デフォルト: "abbrev"

**timefmt**

このタグを使用して、日付を提供するために使用される形式を指定します。

例:

```
<!--#config timefmt="%D %T" -->
<!--#flastmod file=foo.html -->
```

結果: "10/18/95 12:05:33"

デフォルト: "%a, %d %b %Y %T %Z"

以下の strftime() 形式が、timefmt タグで有効です。

指定子	意味
%%	% で置換する。
%a	省略された曜日名で置換する。
%A	完全な (省略なし) 曜日名で置換する。
%b	省略された月名で置換する。
%B	完全な (省略なし) 月名で置換する。
%c	日時で置換する。
%C	世紀数 (100 で割って切り捨てた数字) で置換する。
%d	日付で置換する (01 ~ 31)。
%D	日付を %m/%d/%y として挿入する。

指定子	意味
%e	月数を 10 進数で挿入する (01 ~ 12) (C POSIX に限っては、2 文字の、右寄せされ、ブランクで埋められるフィールドです)。
%E[cCxyY]	代替日付 / 時刻形式が使用できない場合、%E 記述子は、非拡張の対応する記述子にマップされる (例えば、%EC は %C にマップされる)。
%Ec	代替日時表示で置換する。
%EC	代替表示の基本となる年 (期間) の名前で置換する。
%Ex	代替日付表示で置換する。
%EX	代替時間表示で置換する。
%Ey	代替表示の %EC (年のみ) からのオフセットで置換する。
%EY	完全な (省略なし) 代替年表示で置換する。
%h	省略された月名で置換する (%b と同じ)。
%H	10 進数 (00 ~ 23) の時刻 (24 時間時計) で置換する。
%I	10 進数 (00 ~ 12) の時刻 (12 時間時計) で置換する。
%j	年間通算日で置換する (001 ~ 366)。
%m	月で置換する (01 ~ 12)。
%M	分で置換する (00 ~ 59)。
%n	改行で置換する。
%O[deHImMSUwWy]	代替日付 / 時刻形式が使用できない場合、%O 記述子は、非拡張の対応する記述子にマップされる (例えば、%Od は %d にマップされる)。
%Od	代替数値シンボルを使用して、月の日付で置換する。必要に応じて、ゼロを示す代替シンボルがある場合は先行ゼロ、ない場合は先行スペースによって埋められる。
%Oe	代替数値シンボルを使用して、必要であれば先行スペースで埋めて、月の日付で置換する。
%OH	代替数値シンボルを使用して、時間 (24 時間時計) で置換する。
%OI	代替数値シンボルを使用して、時間 (12 時間時計) で置換する。
%Om	代替数値シンボルを使用して月で置換する。
%OM	代替数値シンボルを使用して分で置換する。
%OS	代替数値シンボルを使用して秒で置換する。
%OU	代替数値シンボルを使用して、年間通算週数 (日曜日を週の第 1 日とし、規則は %U と同じ) で置換する。
%Ow	代替数値シンボルを使用して、曜日 (日曜日 = 0) で置換する。
%OW	代替数値シンボルを使用して、年間通算週数で置換する (月曜日が最初の日)。
%Oy	代替表示の年 (%C からのオフセット) で、代替数値シンボルを使用して置換する。
%p	AM または PM と同等のもので置換する。
%r	%l:%M:%S %p と同等のストリングで置換する。



指定子	意味
%R	24 時間表記の時刻で置換する (%H:%M)。
%S	秒で置換する (00 ~ 61)。
%t	タブで置換する。
%T	%H:%M:%S と同等のストリングで置換する。
%u	10 進数の曜日で置換する (1 ~ 7)。1 は月曜日を表す。
%U	年間通算週数 (00 ~ 53) で置換する。日曜日を週の最初の日とする。
%V	年間通算週数 (01 ~ 53) で置換する。月曜日を週の最初の日とする。
%w	曜日で置換する (0 ~ 6)。日曜日は 0 になる。
%W	年間通算週数 (00 ~ 53) で置換する。月曜日を週の最初の日とする。
%x	適切な日付表示で置換する。
%X	適切な時間表示で置換する。
%y	2 桁の年数値と世紀で置換する。
%Y	完全な 4 桁の年数値で置換する。
%Z	時間帯の名前で置換する。時間帯が分からない場合は、文字を何も指定しない。

オペレーティング・システムの構成によって、月名と年が完全表記であるかが省略表記であるかが決まります。

## echo — 可変値を表示する

このディレクティブは、`var` タグを使用して指定された環境変数の値を表示するために使用します。変数が検出されない場合は、(None) が表示されます。**echo** は、**set** または **global** ディレクティブによって設定された値を表示することもできます。以下のような環境変数を表示することができます。

### DATE\_GMT

グリニッジ標準時による現在の日付および時刻。この変数の形式設定は、**config timefmt** ディレクティブを使用して定義されます。

### DATE\_LOCAL

現在の日付と地方時。この変数の形式設定は、**config timefmt** ディレクティブを使用して定義されます。

### DOCUMENT\_NAME

最上位の文書の名前です。HTML が CGI によって生成された場合、この変数には CGI の名前が入ります。

### DOCUMENT\_URI

照会ストリングのない、クライアントが要求した完全 URL。

### LAST\_MODIFIED

現在の文書が最後に変更されたときの日付と時刻。この変数の形式設定は、**config timefmt** ディレクティブを使用して定義されます。

## QUERY\_STRING\_UNESCAPED

クライアントによって送信された検索照会。これは、HTML が CGI によって生成された場合を除いて、未定義です。

## SSI\_DIR

SSI\_ROOT に対応する現在のファイルのパス。現行ファイルが SSI\_ROOT にある場合、この値は "/" になります。

## SSI\_FILE

現行ファイルのファイル名。

## SSI\_INCLUDE

現行ファイルを検索した include コマンドで使用された値。これは、最上位ファイルには定義されていません。

## SSI\_PARENT

SSI\_ROOT に関連して、現行ファイルを検索した include コマンドが入っているファイルのパスとファイル名。

## SSI\_ROOT

最上位ファイルのパス。すべてのインクルード要求は、このディレクトリーまたはこのディレクトリーの子ディレクトリーになければなりません。

例:

```
<!--#echo var=SSI_DIR -->
```

## exec — CGI プログラムを指定する

このディレクティブを使用して、CGI プログラムの出力を組み込みます。exec ディレクティブは、CGI が出力するすべての HTTP ヘッダーを廃棄します。ただし、以下のものは除きます。

## CONTENT-TYPE

他の includes の出力の本文を構文解析するかを示します。

## CONTENT-ENCODING

EBCDIC から ASCII への変換を行う必要があるかを判別します。

## LAST-MODIFIED

現在の値が指定の値よりも新しくない限り、Last-Modified ヘッダーの値を置換します。

## cgi — CGI プログラム URL を指定する

このディレクティブを使用して、CGI プログラムの URL を指定します。

この例では、**program** は実行される CGI プログラムで、**path\_info** および **query\_string** は環境変数としてプログラムに渡される 1 つまたは複数のパラメータを表します。

```
<!--#exec cgi="/cgi-bin/program/path_info?query_string" -->
```

次の例は、変数の使用を示すものです。

```
<!--#exec cgi="%path;%cgiprogram;%pathinfo;%querystring;" -->
```

## flastmod — 文書が最後に変更された日付および時刻を表示する

このディレクティブを使用して、文書が最後に変更された日付および時刻を表示します。この変数の形式設定は、**config timefmt** ディレクティブによって定義されます。**file** および **virtual** タグは、このディレクティブで有効であり、その意味は次のように定義されます。

ディレクティブ形式:

```
<!--#flastmod file="/path/file" -->
<!--#flastmod virtual="/path/file" -->
```

**file** このタグを使用して、ファイルの名前を指定します。**flastmod**、**fsize**、および **include** で、前に **'/'** が付いている場合、**file** は **SSI\_ROOT** に対応するものと想定されます。そうでなければ、**SSI\_DIR** と対応します。指定されたファイルは、**SSI\_ROOT** またはそのサブディレクトリーのいずれかに存在していなければなりません。例えば、次のとおりです。

```
<!--#flastmod file="/path/file" -->
```

**virtual** このタグを使用して、文書への仮想パスの URL を指定します。**flastmod**、**fsize**、および **include** の場合、**virtual** は常にサーバーのマッピング・ディレクティブを介して渡されます。例えば、次のとおりです。

```
<!--#flastmod virtual="/path/file" -->
```

例:

```
<!--#flastmod file="foo.html" -->
```

結果: 12May96

### **fsize** — ファイル・サイズを表示する

このディレクティブを使用して、指定したファイルのサイズを表示します。この変数の形式設定は、**config sizefmt** ディレクティブによって定義されます。**file** および **virtual** タグは、このディレクティブで有効で、その意味は前に **flastmod** ディレクティブに定義したものと同じです。

例:

```
<!--#fsize file="/path/file" -->
<!--#fsize virtual="/path/file" -->
```

結果: 1K

### **global** — グローバル変数を定義する

このディレクティブを使用して、このファイルまたは任意のインクルード・ファイルによって後からエコーすることができる、グローバル変数を定義します。

例:

```
<!--#global var=VariableName value="SomeValue" -->
```

例えば、仮想境界にわたる親文書を参照するために、グローバル変数 **DOCUMENT\_URI** を設定する必要があります。子文書内のグローバル変数も参照する必要があります。次の例は、親文書に挿入する必要がある HTML コーディングを示したものです。

```
<!--#global var="PARENT_URI" value=&DOCUMENT_URI; -->
```

次の例は、子文書に挿入する必要がある HTML コーディングを示したものです。

```
<!--#flastmod virtual=&PARENT_URI; -->
```

### include — 文書を出力に組み込む

このディレクティブを使用して、文書からのテキストを出力に組み込みます。**file** および **virtual** タグは、このディレクティブで有効であり、その意味は **flastmod** ディレクティブの場合に上記に定義したものと同じです。

### set — 変数がエコーされるように設定する

このディレクティブを使用して、このファイルだけが後からエコーすることができる変数を設定します。

例:

```
<!--#set var="Variable 2" value="AnotherValue" -->
```

ディレクティブを定義している間に、**value** 内にストリングをエコーすることができます。例えば、次のとおりです。

```
<!--#include file="&filename;" -->
```

変数: サーバー側 **set** ディレクティブは、一般的に **echo** ディレクティブが続いているので、**set** 変数を検索し、その変数が見つかったら、エコーして、その機能を実行します。これには、変数に対する複数参照を含めることができます。サーバー側 **set** によって、すでに設定されている変数をエコーすることもできます。**set** 変数が見つからない場合は、何も表示されません。

サーバー側 **set** は、サーバー側 **include** ディレクティブ内部の変数参照が見つかったら、サーバー側でこれを解決しようとします。次の例の 2 行目では、サーバー側変数 **&index;** がストリング **var** とともに使用され、変数名 **var1** が構成されます。その後で **&ecirc;** 中の **&** をエスケープして変数 **&var1;** に値が割り当てられるので、変数として認識されなくなります。代わりに、値 **fr&ecirc;d** または **e** の上に曲折アクセント記号のついた **fred** を作成するためのストリングとして使用されます。変数 **&ecirc;** はクライアント側変数です。

```
<!--#set var="index" value="1" -->
<!--#set var="var&index;" value="fr¥&ecirc;d" -->
<!--#echo var="var1" -->
```

エスケープできる文字 (エスケープ変数と呼ばれる文字) は、前に円記号 (¥) が付き、次のようなものがあります。

文字	意味
¥a	アラート (ベル)
¥b	バックスペース
¥f	用紙送り (改ページ)
¥n	改行
¥r	改行復帰
¥t	水平タブ
¥v	垂直タブ

文字	意味
¥'	単一引用符
¥"	二重引用符
¥?	疑問符 (?)
¥¥	円記号
¥-	ハイフン
¥.	ピリオド
¥&	アンバーサンド

## エラー・メッセージのカスタマイズ

Caching Proxy から戻されるエラー・メッセージをカスタマイズして、特定のエラー状態に対応する特定のメッセージを定義することができます。「構成および管理」フォームで、「サーバー構成」->「エラー・メッセージのカスタマイズ」を選択します。このフォームを使用して、エラー状態を選択し、その状態に使用する特定の HTML ファイルを指定します。

プロキシ構成ファイル中のディレクティブを編集してエラー・メッセージをカスタマイズするには、ディレクティブの該当するセクション（228 ページの『ErrorPage - 特定のエラー条件にカスタマイズされたメッセージを指定する』）を参照してください。

## Real Time Streaming Protocol (RTSP) リダイレクト

これは、リバース・プロキシ構成にのみ適用されます。

WebSphere Application Server, バージョン 6.1 では、RTSP リダイレクターという形でストリーミング・メディアのサポートを導入しています。RTSP により Caching Proxy は、メディア・プレイヤーとの最初の接点として働き、メディア・プレイヤーの要求を該当するプロキシ・サーバーへ、または要求されたメディア・コンテンツを提供するコンテンツ・サーバーへリダイレクトすることができます。

RTSP (Real Time Streaming Protocol) は RFC 2326 に定義されています。これは、データ・ストリームを制御するためのインターネット標準プロトコルです。これには、ストリームを送達するテクノロジーは含まれていませんが、ビデオやオーディオの再生とは無関係なデータ・ストリームを制御するために使用できる十分な柔軟性があります。

### RTSP のリダイレクトについて

RTSP リダイレクト機能によって、Caching Proxy は、RTSP で制御されるすべてのストリーミング・メディア・セッションについて要求をリダイレクトできます。そのセッションには、以下のタイプのメディアが含まれます。

- RealNetworks レコード・オーディオ
- RealNetworks レコード・ビデオ
- RealNetworks ライブ・ストリーム (オーディオおよびビデオ)
- Microsoft Media Player ファイル

- Apple Quicktime メディア・ファイル

RTSP ポート (通常は 554) でプロキシー・サーバーに接続するよう構成できるどのプレイヤーも、Caching Proxy でこのフレームワークを使用して、その要求を RTSP リダイレクターに処理させることができます。

RTSP リダイレクターは、プロキシー・メディア・プレゼンテーションをキャッシュに格納したり送信したりしません。RTSP リダイレクターをサード・パーティー製のストリーミング・メディア・サーバーと一緒に使用し、これらの機能のいずれかまたは両方を提供する必要があります。RTSP リダイレクターを備えた Caching Proxy は、1 つまたは複数の RTSP プロキシー・サーバーにネットワーク・アクセスができる必要があります。

## RTSP の制限

この機能には、以下の制約事項があります。

現在、RealNetworks のテクノロジーだけがサポートされています。これらのテクノロジーには、RealProxy プロキシー・サーバー、RealServer 起点サーバー、RealPlayer メディア・プレイヤーなどがあります。

## RTSP の機能強化

従来、RTSP リダイレクターには、どの URL についても同じ起点サーバーに向けられたすべての要求が同じ方法でリダイレクトされるという制限がありました。要求された URL のファイル名またはその他の部分を基にしたリダイレクトは不可能でした。この制限はもはや適用されません。RTSP リダイレクターは、Caching Proxy 構成ファイルで設定されたしきい値 (rtsp\_proxy\_threshold) とともに、受信した要求からの完全 URL を使用して、クライアント要求を起点サーバーとプロキシー・サーバーのどちらにリダイレクトするかを判別するようになりました。同じ起点サーバーへの要求は、個別に処理されるようになりました。

## RTSP リダイレクトの構成

RTSP リダイレクトを制御するために、以下の構成ファイル・ディレクティブを使用します。これらのディレクティブの設定値は、サーバーを再始動してもリフレッシュされません。これらのディレクティブの変更を有効にするには、サーバーを完全に停止したあと、再始動する必要があります。

- 293 ページの『RTSPEnable - RTSP リダイレクトを使用可能にする』
- 293 ページの『rtsp\_proxy\_server - リダイレクト用のサーバーを指定する』
- 294 ページの『rtsp\_proxy\_threshold - キャッシュへのリダイレクトまでの要求の数を指定する』
- 294 ページの『rtsp\_url\_list\_size - プロキシー・メモリー中の URL の数を指定する』

---

## 第 14 章 ヘッダー・オプションの構成

文書を要求するとき、Web クライアントはブラウザや要求に関する追加の情報を提供するヘッダーを送信します。ヘッダーは、要求の送信時に自動的に生成されます。

Caching Proxy では、ヘッダー情報を宛先サーバーから隠しておくようにカスタマイズするための、いくつかのオプションを使用できます。実際のヘッダーをより一般的なヘッダーに置き換えることにはクライアントの匿名性を高める利点がありますが、一部の Web ページに書き込まれるヘッダー・ベースのページのカスタマイズが使用不可になるという欠点もあります。

ヘッダーは通常、次のような形式を使用します。

```
User-Agent: Mozilla 2.02/OS2
Client-IP: 45.37.192.3
Referer: http://www.bigcompany.com/WebTrafficExpress/main.html
```

このヘッダーには、次のようなフィールドが含まれています。

- **User-Agent:** ブラウザーおよびオペレーティング・システムについての情報を提供する。
- **Client-IP:** URL を要求しているクライアントの IP アドレスを提供する。
- **Referer:** 宛先サーバーに、このページへの参照リンクの URL を提供する。

多くのヘッダーは、適切なプロキシ構成設定によってブロックできます。しかし、一部のヘッダー・フィールドは起点サーバーに必要で、それらをブロックすると、Web ページが正しく表示されないことがあります (例えば、ある場合には "host" のヘッダー・フィールドをブロックすると、間違った Web ページが表示されることがあります)。ヘッダー・フィールドについて詳しくは、HTTP バージョン 1.1 の仕様書を参照してください。

---

### 関連するディレクティブ

プロキシ構成ファイルを編集してヘッダー・オプションを変更するには、次のディレクティブの解説セクションを参照してください。

- 263 ページの『NoCacheOnRange - Range 要求でキャッシングなしを指定する』
- 264 ページの『NoProxyHeader - ブロックするクライアント・ヘッダーを指定する』
- 285 ページの『ProxyFrom - From: ヘッダーのクライアントを指定する』
- 285 ページの『ProxyIgnoreNoCache - 再ロード要求を無視する』
- 286 ページの『ProxySendClientAddress - Client - IP: ヘッダーを生成する』
- 286 ページの『ProxyUserAgent - User Agent ストリングを変更する』
- 287 ページの『ProxyVia - HTTP ヘッダーの形式を指定する』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。



---

## 「構成および管理」フォーム

ヘッダー・オプションを指定するために、2 つの「構成および管理」フォームを使用できます。

- 「プロキシ構成」→「プライバシー設定」を選択します。「プライバシー設定」フォームで、以下のものを設定します。

- 宛先サーバーへのクライアントの IP アドレスの転送

要求クライアントの IP アドレスを宛先 (コンテンツ) サーバーに転送したい場合は、このボックスにチェックを入れます。このボックスにチェックがない場合は、宛先サーバーは、プロキシ・サーバーの IP アドレスを受信します。このボックスにチェックを入れないと、Web サーフィン時のクライアントの匿名性が高くなります。

- ユーザー・エージェント・ストリング

宛先サーバーにヘッダーで送信するストリングを入力して、クライアントが使用しているブラウザとオペレーティング・システムのタイプを置き換えます。例えば、「Caching Proxy 4.0」を指定すると、以下のヘッダー中で Mozilla 2.02/OS2 が置き換えられます。

```
Content-Type:MIME
User-Agent: Mozilla 2.02/OS2
Referer: http://www.ics.raleigh.ibm.com/WebTrafficExpress/main.html
Pragma:no-cache
```

- 発信元:

宛先サーバーが "From:" ヘッダーを解析するときに読み取る電子メール・アドレスを入力します。プロキシ管理者は、すべての問題の報告を受け取る必要があるので、プロキシ管理者の電子メール・アドレスを指定することをお勧めします。

- 「実行依頼」をクリックして、構成ファイルを変更します。

- 「プロキシ構成」→「プロキシ・ヘッダー (見出し) のフィルター」を選択します。このフォームを使用して、ブロックする HTTP ヘッダーをリストします。

1. 「追加」または「除去」をクリックして、ブロックされるヘッダーの索引位置を示します。
2. ブロックするクライアントの HTTP ヘッダーを入力します。(詳細なリストとヘッダーの説明については、HTTP 1.1 の仕様書を参照してください。)
3. 「実行依頼」をクリックして、構成ファイルを変更します。

詳細については、7 ページの『第 2 章 「構成および管理」フォームの使用』を参照してください。



---

## 第 15 章 アプリケーション・プログラミング・インターフェースについて

アプリケーション・プログラミング・インターフェース (API) の詳細な説明は、*Edge Components プログラミング・ガイド* に記載されています。構成ファイル内の API ディレクティブによって、要求処理ワークフロー内の特定ステップで呼び出されるプラグイン・ルーチンが使用可能になります。これらのプラグイン・ルーチンは、組み込みルーチンを置き換えることも、組み込みルーチンに追加して実行することもできます。

---

### 関連するディレクティブ

API ディレクティブは、次のとおりです。

- 199 ページの『Authentication - 認証ステップをカスタマイズする』
- 199 ページの『Authorization - 許可ステップをカスタマイズする』
- 227 ページの『Error - エラー・ステップをカスタマイズする』
- 250 ページの『Log - ログ・ステップをカスタマイズする』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 260 ページの『NameTrans - 名前変換ステップをカスタマイズする』
- 265 ページの『ObjectType - オブジェクト・タイプ・ステップをカスタマイズする』
- 272 ページの『PostAuth - PostAuth ステップをカスタマイズする』
- 272 ページの『PostExit - PostExit ステップをカスタマイズする』
- 273 ページの『PreExit - PreExit ステップをカスタマイズする』
- 297 ページの『ServerInit - サーバー初期化ステップをカスタマイズする』
- 298 ページの『ServerTerm - サーバー終了ステップをカスタマイズする』
- 299 ページの『Service - サービス・ステップをカスタマイズする』
- 308 ページの『Transmogriifier - データ操作ステップをカスタマイズする』
- 309 ページの『TransmogriifiedWarning - 警告メッセージをクライアントへ送信する』

詳細については、13 ページの『第 4 章 ibmproxy.conf ファイルの手作業での編集』を参照してください。

---

### 「構成および管理」フォーム

以下の「構成および管理」フォームを使用して関連するディレクティブの値を編集します。

- 「サーバー構成」->「要求処理」->「API 要求処理」

詳細については、7 ページの『第 2 章 「構成および管理」 フォームの使用』を参照してください。

---

## 第 4 部 プロキシ・サーバー・キャッシングの構成

ここでは、プロキシ・キャッシュとその構成方法について説明します。キャッシュは、ファイルをメモリー内に保管 (メモリー・キャッシュ) または 1 つ以上のストレージ・デバイスに保管 (ディスク・キャッシュ) するように設定できます。キャッシュ・リフレッシュ・エージェントを構成して、頻繁に要求されるファイルをキャッシュにプリロードし、また、各種の URL フィルターをキャッシングに適用することができます。またここでは、リモート・キャッシュ・アクセスまたはインターネット・キャッシング・プロトコル (ICP) プラグインを使用し、キャッシュのガーベッジ・コレクションによって廃止ファイルを除去し、生成ファイルを動的にキャッシングする、キャッシュの共用方法についても説明します。

ここには、次の章が含まれます。

- 77 ページの『第 16 章 プロキシ・サーバー・キャッシングの概説』
- 81 ページの『第 17 章 基本的なキャッシュの構成』
- 87 ページの『第 18 章 キャッシュ対象の制御』
- 91 ページの『第 19 章 キャッシュ・コンテンツの保守』
- 99 ページの『第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成』
- 107 ページの『第 21 章 共用キャッシュの使用』
- 111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』
- 115 ページの『第 23 章 プロキシ・サーバー・キャッシュの調整』



---

## 第 16 章 プロキシ・サーバー・キャッシングの概説

キャッシングとは、プロキシ・サーバーがクライアントに要求されたファイルのローカル・コピーを保管して、同一のクライアントや別のクライアントから再び要求されたときに、キャッシュからそのファイルを迅速に提供できるようにする機能です。

Caching Proxy は HTTP 1.1 に準拠しているため、通常 HTTP 1.1 プロトコルに従って、文書をキャッシュに入れ、その新しさを判別します。

この章では、プロキシ・サーバーキャッシュの機能についてそのいくつかを説明します。構成可能な機能については、適切な値の設定方法が以降の章で詳細に説明されています。

---

### キャッシュ・ストレージ

プロキシ・サーバーは、物理ストレージ・デバイスまたはシステム・メモリーにキャッシュを格納することができます。ユーザーのシステムに適したキャッシュ・ストレージのタイプは、使用するハードウェアの性能によって、またキャッシュの応答速度とキャッシュに格納する項目数ではどちらの方が重要かによって異なります。一般に、メモリー・キャッシュの応答時間はディスク・キャッシュの場合より高速ですが、メモリー・キャッシュのサイズはプロキシ・サーバー・マシンの RAM 容量によって制限されます。ディスク・キャッシュのサイズは、ストレージ・デバイスのサイズによって制限されますが、これは通常、RAM 容量よりかなり大きなサイズになります。

ディスク・キャッシュの場合、Caching Proxy はロー・ディスク・キャッシュを使用します。これは、プロキシ・サーバーがオペレーティング・システムの読み取りおよび書き込みプロトコルを使用せずに、直接キャッシュ・デバイスに書き込むという意味です。 **htcformat** コマンドを使用して、ディスク・キャッシュ用のストレージ・デバイスを準備しておく必要があります。**htcformat** の詳細は、81 ページの『第 17 章 基本的なキャッシュの構成』のセクションにあります。

---

### キャッシュ索引

メモリー・キャッシュとディスク・キャッシュのどちらの場合でも、Caching Proxy はシステム・メモリー・スペースを使用してキャッシュの索引も保持します。これにより、キャッシュされたファイルの検索に要する処理時間が短縮されます。

この Caching Proxy と他のプロキシ・サーバーではキャッシュ・ディレクトリー構造と索引付けメソッドが異なります。Caching Proxy は、キャッシュ内のファイルに関する情報を含む索引をメモリー内で保守します。ディスクまたはその他のメディアの代わりに索引用に RAM を使用すると、ファイルの索引付けと検索がより速くなります。

索引には、キャッシュに格納されたオブジェクトの URL、キャッシュの場所、および有効期限情報が含まれています。このため、索引を入れるのに必要なメモリーの量は、キャッシュ内のオブジェクトの数に比例します。

クライアントから要求を受信すると、プロキシーは、メモリー内のキャッシュ索引を調べてその URL を検索します。

- 該当のファイルが索引内にない場合は、宛先サーバーに要求します。
  - 次にその URL を調べて、検索されたファイルをキャッシュに格納できるかどうかを判別します。可能な場合、プロキシー・サーバーは検索されたファイルをキャッシュに格納します。
  - 次に、新たにキャッシュされたオブジェクトの URL、場所、および有効期限情報により、キャッシュ索引が更新されます。
- 索引内に該当のファイルがある場合には、次のようになります。
  - 有効期限情報を調べて、キャッシュ・ファイルが新しいものかどうかを判別します。
    - オブジェクトの有効期限が切れている場合は、宛先サーバーに接続し、新たに検索した文書にで期限切れのオブジェクトを置き換えます。キャッシュ索引内の有効期限情報が更新されます。
    - オブジェクトの有効期限が切れていない場合は、プロキシー・キャッシュから文書が呼び出されます。

---

## FTP キャッシュ

プロキシーは、要求をキャッシュに入れるように構成されると、FTP ファイル要求および HTTP ファイル要求をキャッシュに入れることができます。ただし、FTP ファイルには HTTP ファイルと同じタイプのヘッダー情報が含まれていないため、キャッシュ FTP ファイルの有効期限は他のキャッシュ・ファイルとは異なる方法で計算されます。

FTP サーバーにファイル検索の要求を出すと、プロキシーはまず、FTP サーバーにそのファイルの LIST 要求を送信して、そのファイルに関する FTP ディレクトリー情報を入手します。FTP サーバーが LIST 要求に肯定的な完了応答とそのファイルのディレクトリー情報で応答すると、プロキシーは、FTP ディレクトリー情報から解析した日付で HTTP の Last-Modified ヘッダーを作成します。次に、プロキシーのキャッシュ機能は、この Last-Modified ヘッダーを、構成ファイルの `CacheLastModifiedFactor` ディレクティブに設定された値と共に使用して、期限が切れるまでに FTP ファイルがキャッシュに残る時間を決定します。

"Last-Modified" ヘッダーと `CacheLastModifiedFactor` ディレクティブを使用して、ファイルがキャッシュに残る時間を決定する方法の詳細については、91 ページの『第 19 章 キャッシュ・コンテンツの保守』を参照してください。

無名ログインではなく、特定のユーザー ID で検索される FTP ファイルは、プライベートなファイルと考えられ、キャッシュに格納されません。

---

## DNS キャッシュ

Web コンテンツのキャッシングに加えて、プロキシ・サーバーはドメイン・ネーム・サーバー (DNS) のキャッシュも実行します。例えば、クライアントが `www.myWebsite.com` から URL を要求すると、プロキシは、ホスト名 `www.myWebsite.com` を IP アドレスに変換するよう、その DNS サーバーに求めます。その後、この IP アドレスがキャッシュに格納され、そのホスト名に対する以降の要求に必要な応答時間が改善されます。DNS キャッシュは自動的に行われ、再構成することはできません。

---

## キャッシュの除外

ファイルおよび文書の中には決してキャッシュに格納されないものもあります。そのようなファイルまたは文書とは、以下のようなものです。

- POST や PUT など、GET 以外の HTTP メソッドを使用する要求から戻されるファイル。
- 起点サーバーによってキャッシングが特に許可される場合を除き、認証を必要とする文書。
- CGI スクリプトの動的出力 (要求されるたびに異なるため)。動的キャッシング機能を使用可能な場合は、IBM WebSphere Application Server によって実行されたサーブレットおよび JavaServer Pages (JSP) からの、動的に生成された結果をキャッシュに入れることができます。詳細については、111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』を参照してください。
- SSL トンネリング接続で渡される情報 (プロキシがこの接続を介して渡されるデータを暗号化解除することができないため)。
- 照会キャッシュが特に許可される場合を除き、疑問符 (?) が含まれている URL から戻されるすべてのファイル。(照会結果のキャッシュの構成については、87 ページの『第 18 章 キャッシュ対象の制御』を参照してください。)

キャッシュ・フィルターを設定すると、キャッシュに入れる項目をさらに制限することができます。例えば、プロキシ・サーバーで プロキシからローカルで提供されたファイルをキャッシュに入れないようにすることができます。詳細については、87 ページの『第 18 章 キャッシュ対象の制御』を参照してください。

---

## キャッシュの管理

キャッシュの管理には、多くの要因が含まれます。サーバー管理者として、次を指定できます。

- キャッシュされる対象文書 (詳細は 87 ページの『第 18 章 キャッシュ対象の制御』参照)。
- キャッシュできる文書の数 (詳細は 81 ページの『第 17 章 基本的なキャッシュの構成』参照)。
- キャッシュされる文書が現行文書と見なされる期間 (詳細は 91 ページの『第 19 章 キャッシュ・コンテンツの保守』参照)。
- キャッシュがパーシステント (ガーベッジ・コレクション) される頻度と保持されるファイルのタイプ (詳細は 91 ページの『第 19 章 キャッシュ・コンテンツの保守』参照)。

- キャッシュされる文書の索引付けされる方法 (詳細は 81 ページの『第 17 章 基本的なキャッシュの構成』参照)。
- キャッシュがリフレッシュされる時期 (詳細は 99 ページの『第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成』参照)。
- リモート・キャッシュ・アクセス (詳細は 107 ページの『第 21 章 共用キャッシュの使用』参照)。
- ログを保存およびアーカイブする方法 (詳細は 81 ページの『第 17 章 基本的なキャッシュの構成』参照)。

さらに、Caching Proxy の全体的なパフォーマンスを改善するために、キャッシュ構成の調整を行うことができます。パフォーマンス調整の詳細については、115 ページの『第 23 章 プロキシ・サーバー・キャッシュの調整』を参照してください。



---

## 第 17 章 基本的なキャッシュの構成

Caching Proxy をインストールするために Edge components 製品セットアップ・プログラムでデフォルト設定を使用した場合、キャッシュは使用可能になり、メモリーにキャッシュが格納されます。次の基本的なキャッシュ設定を調整して、システムの要件に応じてキャッシュをカスタマイズできます。

セットアップ・プログラムを使用しなかった場合は、これらの設定を構成すると、キャッシュが使用可能になります。

キャッシュを構成するのに必要な基本ステップは次のとおりです。

1. キャッシュを使用可能にする。
2. キャッシュ・ストレージを構成する。

基本的なキャッシュ設定を構成したら、次の機能の設定を追加または変更できます。

- キャッシュ・メモリーをカスタマイズする
- キャッシュ・メモリーをディスクに保管またはロードする
- URL フィルターを使用してキャッシュ対象を制限する
- 照会結果または動的に生成されたファイルのキャッシュを可能にしてキャッシュ対象を拡大する
- キャッシュ・ファイルの有効期限とガーベッジ・コレクションを構成する
- 自動キャッシュ・リフレッシュおよびプリロードを構成する
- リモート・キャッシュ・アクセス (RCA) またはインターネット・キャッシング・プロトコル (ICP) とのキャッシュ共用を構成する
- ログ記録を構成する

これらの各設定の変更手順については、本章で説明しています。

---

### 1. キャッシュを使用可能にする

キャッシングを使用可能にするには、Caching ディレクティブをオンに設定するか、または「キャッシュ構成」->「キャッシュ設定」構成フォームの「プロキシー・キャッシングを使用可能にする」ボックスにチェックマークを付けます。キャッシュ・デバイスを指定しないと、キャッシュはメモリーに保管されます。ディスク・キャッシュを作成するには、『2. キャッシュ・ストレージを構成する』の手順に従ってください。

---

### 2. キャッシュ・ストレージを構成する

キャッシュ・ストレージを構成するタスクは、メモリー・キャッシュを使用するか、ディスク・キャッシュを使用するかによって異なります。

メモリー・キャッシュを使用するには、キャッシュのコンテンツを保持するのに十分なメモリーが組み込まれるように「キャッシュ・メモリー」の設定をカスタマイ

ズします。キャッシュ・メモリーの推奨サイズについては、83 ページの『キャッシュ・メモリーを設定する』を参照してください。

ディスク・キャッシュを使用するには、以下の手順を実行することが必要です。

1. キャッシュを入れるストレージ・デバイスを準備します。

キャッシュには、特別にフォーマットされたデバイスが必要です。デバイスまたはディスク区画全体をキャッシュ専用にすることをお勧めします。キャッシュの最小サイズは 16392KB です。

キャッシュ・デバイスをフォーマットするには、次のようにします。

- a. キャッシュを入れるデバイスを選択します。他のプログラムでそのストレージ・スペースが使用されていないこと、およびロー（つまり、文字形式の）デバイスとしてそのデバイスにアクセスできることを確認します。
- b. **htcformat** コマンドを使用してデバイスをフォーマットします。その構文は、次のとおりです。

```
htcformat raw_device_path [-blocksize block_size]
                        [-blocks number_of_blocks]
```

-blocksize 引き数と -blocks 引き数はオプションです。デフォルトのブロック・サイズは 8192 バイトです。ブロックの数を指定しない場合、ディスク区画には、収容できるだけの数のブロックが格納されます。

デバイス・パスを指定するときには、必ずロー・デバイスのパスを指定してください。

- AIX プラットフォームでは、/dev/lv02 と定義された論理ボリュームのロー・デバイスのパスは /dev/rlv02 となります
- Linux プラットフォームでは、最初に **raw** コマンドを実行してから、以下のような **htcformat** を実行して、ロー・デバイスのパスを実際の SCSI ドライブ **sdb1** に関連付ける必要があります。

```
raw /dev/raw/raw1 dev/sdb1
```

- HP-UX および Solaris プラットフォームでは、/dev/dsk/c0t0d0s0 と定義された区画のロー・デバイスのパスは /dev/rdsk/c0t0d0s0 となります
- Windows プラットフォームでは、e: と定義されたデバイスのロー・デバイスのパスは ~~¥¥.¥¥~~e: となります

ロー・デバイスへのアクセスに関する追加情報については、使用中のファイル・システムの参考資料を参照してください。

2. CacheDev ディレクティブまたは「**キャッシュの設定**」構成フォームを使用することにより、キャッシュ・デバイスを指定します。複数のデバイスを指定できます。

注意:

Windows システムでは、`htcformat` コマンドによって、キャッシュ・デバイスに書き込み不可のマークが自動的に付けられることはありません。

オペレーティング・システムがキャッシュ・デバイスへの書き込みを試みた場合、キャッシュ・データが消失する可能性があります。これを避けるために、Windows の **Disk Manager** ユーティリティーを使用して、`htcformat` コマンドを使用する前にディスクの準備を行うことができます。ディスクの準備を行うには、このディスク・ユーティリティーを使用して、使用するデバイスまたは区画を削除してから、デバイスまたは区画をフォーマットしないで再作成してください。このようにすると、システムは、そのデバイスがシステム・ストレージ用に選択不可能であると見なします。

---

## オプション・カスタマイズ

### キャッシュ・メモリーを設定する

CacheMemory ディレクティブ内 (または「**キャッシュ設定**」構成フォームの「**キャッシュ・メモリー**」フィールド内) の値を、以下の原理に従って設定します。この値に設定されたメモリーは、キャッシュ索引を含むキャッシュ・インフラストラクチャーのサポートに使用され、また、メモリー・キャッシュが構成されていれば、キャッシュのコンテンツを格納するのに使用されます。

#### 最小値

ディスク・キャッシュのパフォーマンスを最適にするためには、キャッシュ索引を含むキャッシュ・インフラストラクチャーのサポートには、キャッシュ・メモリーの値を最小 64 MB にすることをお勧めします。キャッシュ・サイズが増えると、キャッシュ索引が増加し、索引を保管するためにさらにキャッシュ・メモリーが必要になります。64 MB のキャッシュ・メモリー値は、キャッシュ・インフラストラクチャーのサポートを提供し、約 6.4 GB までのディスク・キャッシュ用のキャッシュ索引を保管するために十分な大きさです。もっと大きなディスク・キャッシュの場合、キャッシュ・メモリーは、キャッシュ・サイズの 1% にすべきです。

メモリー・キャッシュの場合、キャッシュ・メモリーの値は、キャッシュ・インフラストラクチャーのサポートとキャッシュそれ自体のために確保するメモリーの量になります。最小 64 MB のキャッシュ・メモリー値をお勧めします。

#### 最大値

メモリー・キャッシュ用にあまりにも多すぎる物理メモリーが割り振られると、「メモリー不足」エラーやプロキシ・サーバー障害などの、望ましくない動作が発生する可能性があります。キャッシュ・メモリーの制限値は、32 ビット・アプリケーションの制限に依存します。Caching Proxy は 32 ビット・アプリケーションなので、最大 2 GB のメモリーを使用可能です。

Caching Proxy によって、CacheMemory ディレクティブで定義されたメモリーが割り振られ、オブジェクトを格納するキャッシュとして使用されます。メモリー・キャッシュまたはロー・ディスク・キャッシュのいずれの場合でも、追加のメモリーを割り振る必要があります。このメモリーは、キャッシュのデータ構造、ネットワーク入出力および接続用のバッファ、セッション用バッファ、およびメイン・

プロセスや他のすべてのスレッド用のメモリーなどに使用されます。さらに、クライアントからの要求の中には、デフォルトよりも大きなメモリー・プール・ブロックを割り振る必要が生じる場合もあります。したがって、CacheMemory ディレクティブが 2 GB に近い値に設定されている場合、特に要求負荷が高い状況では、Caching Proxy で操作に十分なメモリーを確保できない可能性もあります。

CacheMemory ディレクティブの値を 1600 MB 以下に設定することをお勧めします。1600 MB を超える値を設定すると、Caching Proxy が通常の操作で必要とするメモリーが不足し、不都合な副次作用が発生する原因となります。一般にこれらの副次作用は、増大する CPU 使用率 (場合によっては使用率 100% まで)、メモリー不足エラー、およびパフォーマンス低下などだけではありません。全体的に大きなキャッシュ・サイズが必要な場合は、キャッシュ・デバイスを使用するか、または RCA または ICP を使用した共用キャッシュ構成をインプリメントしてください。

## キャッシュ・メモリーをディスクに保管またはロードする

キャッシュの内容をダンプ・ファイルからインポートしたり、ダンプ・ファイルへエクスポートしたりすることができます。再始動時にキャッシュ・メモリーが破損したり、同じキャッシュを複数のプロキシに配置する場合などに役立つ機能です。

## キャッシュ・フィルターを設定する

フィルターでは、URL 要求の形式に一致させることによって、キャッシュ対象を制限できます。フィルター設定について詳しくは、87 ページの『第 18 章 キャッシュ対象の制御』を参照してください。

## 照会結果および動的に生成されたファイルのキャッシュを構成する

オプションで、照会要求の結果をキャッシュに入れるようにプロキシ・サーバーを構成できます。デフォルトでは、疑問符 (?) が含まれている URL はキャッシュに入れられません。詳細については、88 ページの『照会応答のキャッシュ』を参照してください。

もう 1 つのオプションでは、IBM WebSphere Application Server からサーブレットまたは JSP を実行した結果をキャッシュに入れることができます。詳細については、111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』を参照してください。

## ファイルの有効期限とガーベッジ・コレクションを構成する

キャッシュ内のファイルの有効期限が切れた場合の構成、および廃止ファイルの除去方法について詳しくは、91 ページの『第 19 章 キャッシュ・コンテンツの保守』を参照してください。

## 自動プリロードを構成する

頻繁にアクセスされるファイルのほとんどを、リフレッシュ要求が出される前に自動的に毎日リフレッシュするように、キャッシュを構成することができます。詳しくは、99 ページの『第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成』を参照してください。

## キャッシュ共用を構成する

特定の環境下では、共用キャッシュを使用すると、要求されたファイルがキャッシュで検出される可能性が高くなります。詳しくは、107 ページの『第 21 章 共用キャッシュの使用』を参照してください。

## ログ記録を構成する

ログを簡潔かつ正確に保つことは、Caching Proxy を管理する上で重要なことです。157 ページの『第 6 部 Caching Proxy のモニター』に、プロキシー・サーバー・ログの構成と使用方法に関する情報が記載されています。



---

## 第 18 章 キャッシュ対象の制御

Caching Proxy は、キャッシュされるファイル、文書、およびその他のオブジェクトを制御するために、いくつかのフィルター操作メソッドを提供します。メソッドには次の機能が含まれます。

- URL ベースのキャッシュ・フィルター
- 照会応答キャッシュ
- ローカル提供ファイルのキャッシュ
- 部分 URL ベースのキャッシュ
- 要求 URL の一部を基にしたファイルのキャッシュ
- 動的に生成されたファイルのキャッシュ — 111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』を参照

注: 「キャッシュ構成」→「キャッシュの動作」構成および管理フォームには、「着信 URL に基づくキャッシュ」というラベルのオプションがあります。(対応する構成ファイル・ディレクティブは `CacheByIncomingURL` というファイルです。) このディレクティブは、キャッシュされたファイルのファイル名を参照します。キャッシュに入れるファイルのファイル名を着信 URL に基づいたものにするには、このボックスにチェック・マークを付けてください。このボックスにチェック・マークを付けない場合は、ファイル名は、発信 URL に基づきます。

---

### URL ベースのキャッシュ・フィルターの構成

ファイルをキャッシュに入れるかどうかを判別するために、URL テンプレートに対する要求を比較するようにプロキシ・サーバーを構成することができます。この機能は、ファイルが常にキャッシュに入れられる要求のテンプレートと、ファイルを決してキャッシュに入れてはならない要求の個別のテンプレートを設定することによって、構成されます。複数のテンプレートを使用できます。

これと類似したシステムが、照会応答キャッシュを使用可能にする時にも使用されます。詳しくは、88 ページの『照会応答のキャッシュ』を参照してください。

`ibmproxy.conf` ファイルを編集して URL キャッシュ・フィルターを設定するには、209 ページの『CacheOnly - テンプレートと一致する URL を持つファイルだけをキャッシュに入れる』および 262 ページの『NoCaching - URL がテンプレートと一致したファイルはキャッシュに入れないことを指定する』を参照してください。

構成および管理フォームで URL キャッシュ・フィルターを設定するには、「キャッシュ構成」→「キャッシュの動作」の「URL によりキャッシュをフィルターに掛ける」フィールドを使用します。常にキャッシュに入れるファイルのある URL を指定するとき、または決してキャッシュに入れないファイルのある URL を指定するときに、このセクションを使用します。2 つのリスト、つまり、常にキャッシュに入れるファイルのリストと決してキャッシュに入れないファイルのリストを指定



するために、どちらか一方のリストを作成してから「**実行依頼**」をクリックし、そのあとでもう一方のリストを作成してください。

---

## 照会応答のキャッシュ

照会 (疑問符を含む URL 要求) から戻された応答を、キャッシュ・フィルターを使用してキャッシュに入れることができます。多くのクライアントが同じ照会要求を行う場合、この機能はリバース・プロキシ (代理) シナリオに役立ちます。

照会キャッシュを構成するには、ibmproxy.conf 構成ファイルで CacheQueries ディレクティブを編集します。CacheQueries ディレクティブには、次のオプションがあります。

- **Always** — テンプレートと一致するホストからのすべての応答は、HTTP 1.1 標準でキャッシュ可能であれば、キャッシュに入れられます。
- **Public** — テンプレートと一致するホストからの応答は、"Cache-control: public" ヘッダーまたは強制再検査ヘッダーを含み、さらに HTTP 1.1 標準でキャッシュ可能であれば、キャッシュに入れられます。

これらのオプションについての追加情報は、210 ページの『CacheQueries - 疑問符 (?) を含む URL へのキャッシュ応答を指定する』に示されています。

構成および管理フォームで照会応答キャッシュを構成するには、「**キャッシュ構成**」→「**キャッシュの動作**」の「**URL によりキャッシュ照会応答をフィルターに掛ける**」フィールドを使用します。2 つのリストを指定するには、どちらか一方のリストを作成してから「**実行依頼**」をクリックし、そのあとでもう一方のリストを作成してください。

## 照会応答キャッシュの追加要件

照会応答をキャッシュ可能にするには、照会キャッシュの構成の他に、以下の設定を適切に構成するようにしてください。「構成および管理」フォームを使用してこれらのオプションを設定する方法については、94 ページの『キャッシュの新しさの構成』を参照してください。

- **CacheTimeMargin** — このディレクティブにより、最小有効期限時間が指定されます。この最小値未満の有効期限時間のファイルは、キャッシュに入れられません。照会応答は時には非常に短い有効期限時間を持つため、このディレクティブを低い設定値にすると、より多くの照会応答がキャッシュに入れられます。211 ページの『CacheTimeMargin - ファイルをキャッシングする場合の最小存続時間を指定する』を参照するか、または 94 ページの『キャッシュの新しさの構成』に説明されている「**キャッシュ有効期限設定**」フォームを使用してください。
- **CacheDefaultExpiry** — このディレクティブにより、有効期限時間の計算に使用される明示的な有効期限も最終変更日時もないファイルについて、有効期限時間が指定されます。この設定値を HTTP 要求に対してデフォルトの 0 から増やすことにより、より多くの照会応答がキャッシュに入れられます。ただし、この方法で設定値を変更すると、失効したコンテンツがキャッシュから供給されるリスクも増えます。203 ページの『CacheDefaultExpiry - デフォルトのファイル有効期限時間を指定する』を参照するか、または 94 ページの『キャッシュの新しさの構成』に説明されている「**キャッシュ有効期限設定**」フォームを使用してください。



- **CacheLastModifiedFactor** — このディレクティブは、最終変更日時はあるが明示的な有効期限のないファイルについて、有効期限を計算するのに使用されます。  
HTTP ファイルの係数を高い値に設定すると、HTTP ファイルが妥当性の再検査なしにキャッシュに留まる時間が増えます。この方法で設定値を変更すると、失効したコンテンツがキャッシュから供給されるリスクも増えます。205 ページの『CacheLastModifiedFactor - 有効期限を決定する値を指定する』を参照するか、または 94 ページの『キャッシュの新鮮さの構成』に説明されている「**最終変更係数**」フォームを使用してください。
- オプションで、**SignificantUrlTerminator** ディレクティブおよび **AggressiveCaching** ディレクティブを設定します。300 ページの『SignificantURLTerminator - URL 要求の終了コードを指定する』および 197 ページの『AggressiveCaching - キャッシュ不可能ファイルのキャッシュを指定する』を参照してください。

---

## ローカル提供ファイルのキャッシュ

一般的に、プロキシー・サーバーから提供されるファイルをキャッシュに入れるのは非効率的なので、サーバーのローカル・ドメイン起源のファイルは、デフォルトではキャッシュに入れられません。サーバーのローカル・ドメイン起源のオブジェクトをキャッシュに入れるには、「**キャッシュ構成**」→「**キャッシュの動作**」構成および管理フォームの「**ローカル・ドメイン・ファイルをキャッシュに入れる**」ボックスにチェックマークを付けます。または、プロキシー構成ファイルで **CacheLocalDomain** ディレクティブを **on** に設定します。

---

## 部分 URL によるファイルのキャッシュ

項目は、完全な URL でなく、着信 URL の指定部分 (重要な部分) に基づいてキャッシュできるようになりました。これは、着信要求の URL の重要な部分が同一であるときにさまざまな着信要求に対してしばしば同一の応答が返されるので、トランザクション・モデルの Web サービスや動的キャッシングの場合に便利です。

部分 URL に基づいてキャッシュを指定するために、「構成および管理」フォームを使用することはできません。代わりに、プロキシー構成ファイル内の **SignificantUrlTerminator** ディレクティブを使用して URL 要求の終了コードを指定してください。この指定により、**Caching Proxy** では、要求の処理時に終了コードの前の文字だけを評価して、要求されたファイルがキャッシュされているかどうかを判別されます。複数の終了コードが定義されている場合には、**Caching Proxy** は着信 URL を **ibmproxy.conf** ファイルに定義されている順序での終了コードと比較します。詳しくは、300 ページの『SignificantURLTerminator - URL 要求の終了コードを指定する』を参照してください。

---

## 関連する構成ファイル・ディレクティブ

プロキシー構成ファイルを直接編集してキャッシュ・フィルターを設定するには、次のディレクティブの該当するセクションを参照してください。

- 262 ページの『NoCaching - URL がテンプレートと一致したファイルはキャッシュに入れないことを指定する』
- 209 ページの『CacheOnly - テンプレートと一致する URL を持つファイルだけをキャッシュに入れる』

- 210 ページの『CacheQueries - 疑問符 (?) を含む URL へのキャッシュ応答を指定する』
- 206 ページの『CacheLocalDomain - ローカル・ドメインをキャッシュに入れるかどうかを指定する』
- 300 ページの『SignificantURLTerminator - URL 要求の終了コードを指定する』

キャッシュに格納できない文書の詳細については、77 ページの『第 16 章 プロキシ・サーバー・キャッシングの概説』を参照してください。

---

## 第 19 章 キャッシュ・コンテンツの保守

キャッシュには、提供されたファイルのコピーを作成し保管する作業が含まれるので、キャッシュが正しく機能するためにはルーチン保守が必要です。キャッシュに入れられたファイルは、新しさについて検査して、起点サーバー上のファイルとの整合性がすでになくなっている場合は無効にする必要があります。このようなファイルの満了処理については、『ファイルの有効期限』で説明しています。また、無効にしたファイルまたは未使用のファイルは、新しいファイル用のスペースを作るために、キャッシュから除去する必要があります。このようなキャッシュ・ページ処理については、96 ページの『ガーベッジ・コレクション』で説明しています。

---

### ファイルの有効期限

キャッシュに格納されたオブジェクトとコンテンツ・サーバー上の元のオブジェクトとの整合性を保持することを、キャッシュの新しさを維持すると言います。

Caching Proxy は、キャッシュに入れる文書またはその他のオブジェクトごとに、オブジェクトの有効期限が切れる時刻を計算します。

HTTP ページの場合、コンテンツ・サーバーが生成する文書のヘッダーには、有効期限情報が含まれています。

FTP プロトコルには同等の有効期限情報が含まれていないので、Caching Proxy は、FTP ファイルに対して上記で説明しているように、それぞれのファイルの FTP ディレクトリー情報に基づいて独自の Last-Modified ヘッダーを生成して、その情報を有効期限の計算に使用します。プロキシー・サーバーが FTP サーバーからファイルのディレクトリー情報を取得できない場合は、FTP URL と一致するデフォルト値が使用されます。また、FTP サーバーには標準の日付形式がないので、Caching Proxy は一部の FTP サーバーが送信した日付および時刻を認識できない場合があります。その場合、プロキシー・サーバーのデフォルトの有効期限時刻値が使用されます。この手順によって、プロキシーは HTTP ページと FTP ファイルのキャッシングを同様の方法で管理することができます。

有効期限は、以下の方法（優先順位の順）のいずれかでコンテンツ・サーバーにより指定できます。

1. Cache-control: s-maxage= $n$  というヘッダーをコンテンツ・サーバーが指定します。このヘッダーにより、オブジェクトの受信後  $n$  秒間はそれが新しいオブジェクトであることをプロキシーに知らせます。
2. Cache-control: max-age= $n$  というヘッダーをコンテンツ・サーバーが指定します。このヘッダーにより、オブジェクトの受信後  $n$  秒間はそれが新しいオブジェクトであることをプロキシーに知らせます。
3. Expire:  $n$  というヘッダーをコンテンツ・サーバーが指定します。このヘッダーにより、 $n$  で指定した時刻まではオブジェクトが新しいオブジェクトであることをプロキシーに知らせます。
4. ヘッダー "Last-Modified:  $n$ " を使用して、文書が最後に変更された日時をコンテンツ・サーバーが示します。プロキシー・サーバーは、文書が最後に変更さ

れたとき以降の経過時間を計算し、この時間に、プロキシー構成ファイル内に設定されているキャッシュ最終変更係数を乗算し、文書がその時間だけ有効であると想定します。例えば、文書が最後に変更されたのは 1 週間 (7 日) 前であるとコンテンツ・サーバーによって示された場合に、キャッシュ最終変更係数が 0.14 であれば、プロキシー・サーバーは、その文書は約 1 日有効であると想定します。キャッシュ最終変更係数の設定の詳細については、94 ページの『キャッシュの新しさの構成』を参照してください。

5. 上記の情報のいずれもコンテンツ・サーバーによって指定されない場合、Caching Proxy は、現行の URL と一致するキャッシュ・デフォルト有効期限設定値を検索し、これを有効期限時間に使用します。キャッシュ・デフォルト有効期限値の設定の詳細については、94 ページの『キャッシュの新しさの構成』を参照してください。

前述の詳細情報を使用して有効期限時間が計算された後、Caching Proxy では、この URL に適用される最小保持時間の値があるかどうか調べられます。この値があるときに、指定された時間が計算した有効期限時間より長い場合は、最小保持時間値で指定された時間がオブジェクトの有効期限時間として使用されます。このことは、Caching Proxy が文書の有効期限時間を 0 分と計算した場合にも該当します。したがって、失効したコンテンツがサービスされるのを回避するために、最小保持時間設定値の使用には注意してください。(最小保持時間値を設定するには、CacheMinHold ディレクティブまたは「**キャッシュ構成**」→「**キャッシュ有効期限設定**」の「**URL 有効期限**」の設定を使用してください。詳しくは、94 ページの『キャッシュの新しさの構成』を参照してください。)

最後に有効期限時間の値は、「時間マージン」の設定値と照合されます。有効期限時間が「時間マージン」値より大きい場合には、文書はキャッシュに格納され、そうでない場合には、キャッシュに追加されません。(「時間マージン」値を設定するには、CacheTimeMargin ディレクティブを使用するか、または 94 ページの『キャッシュの新しさの構成』の指示に従ってください。)

文書がキャッシュ内にあるが、有効期限が切れている場合には、Caching Proxy は、コンテンツ・サーバーに対して *if-modified-since* 要求という特別の要求を出します。この要求によって、文書がプロキシーに最後に受信された後に変更された場合のみ、コンテンツ・サーバーは文書を送信することになります。文書が変更されていない場合には、コンテンツ・サーバーはそのことを示すメッセージを送って、そのページは再送信しません。この場合、プロキシーは、キャッシュに格納された文書をサービスします。FTP ファイルの場合、プロキシー・サーバーは、この *if-modified-since* プロセスをシミュレートします。ファイルが FTP サーバーで変更されなかったと判別すると、キャッシュからファイルがサービスされます。そうでない場合には、FTP サーバーから新しいバージョンを入手します。

## キャッシュの新しさに関する追加情報

- ほとんどすべての静的 Web 文書には (動的に生成される文書とは異なり)、Last-Modified ヘッダーが含まれています。これは、プロキシーが文書の有効期限時間を計算するための最も一般的な方法であり、Caching Proxy が FTP ファイルについて最初に試みる方法です。この方法に失敗すると、プロキシーは、デフォルト有効期限値を参照します。

- Cache-control: s-maxage、Cache-control: max-age、または Expires: ヘッダーを使用する文書は、きわめて少数です。
- キャッシュできない場合が多い、動的に生成されるページには、Expires: 0 または Cache-control: no-cache というヘッダーが含まれていることがあります。これらのヘッダーは、文書の有効期限が即時に切れることを意味します。IBM WebSphere Application Server から動的に生成されたファイルのキャッシュについて詳しくは、111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』を参照してください。
- HTTP: 構文を使用している URL に対して 0 分以外の値をデフォルト有効期限値に設定するときには、注意が必要です。多くの動的生成ページには有効期限ヘッダーは含まれていないので、それらはデフォルトの有効期限値に従います。デフォルト有効期限を 0 分より大きい値に設定すると、プロキシがこのようなオブジェクトをキャッシュに入れることができるようになります。しかし、このことは、ユーザーが古いコンテンツ (または CGI プログラムあるいはサプレットから予期しない結果) を得るということを意味する場合があります。
- 以下の状況では、キャッシュされた文書の有効期限が切れたかどうかにかかわらず、プロキシ・サーバーによって、すべての要求に対してサーバーで文書の妥当性の再検査が行われます。
  - 文書に次のヘッダーのいずれかが含まれている。
    - Cache-control: s-maxage
    - Cache-control: must-revalidate
    - Cache-control: proxy-revalidate
  - 文書には、ユーザー認証が必要であるが、サーバーによりキャッシュに入れることが許可されている。
  - 文書には、Cache-Control: no-cache ヘッダーが含まれているが、それでもキャッシュされる (アグレッシブ・キャッシングにより)。

## FTP での日付について

これは、フォワード・プロキシ構成にのみ適用されます。

FTP プロトコルは HTTP プロトコルほど厳格に日付や時刻を定義しないので、FTP ファイルに対してプロキシが生成する Last-Modified ヘッダーは、いくつかの要因によって実際のファイル日付とわずかに異なることがあります。このような要因として、以下のものがあります。

- HTTP プロトコルと異なり、FTP プロトコルには、戻された日付がグリニッジ標準時 (GMT) でなければならないという指定はありません。FTP サーバーが戻す日付は、たいていの場合、その FTP サーバーの地方時です。プロキシには FTP サーバーがどの時間帯で稼働しているかを判別する方法がないので、その時刻をプロキシの時間帯における時刻と解釈します。例外は Windows FTP サーバーで、このサーバーは GMT で日付を戻します。プロキシは、FTP サーバーが Windows システム上で稼働していることを検出した場合、ディレクトリーの日付が GMT であると考えます。
- FTP サーバーの一部は、戻されたディレクトリー情報の日付を 月、日、年 の形式でのみ指定し、指定された日付に実際の時間や分の情報を組み込みません。FTP サーバーがファイルの時刻情報を戻さない場合、プロキシは、そのファイルの最終変更時刻を、FTP サーバーが戻した日付の可能な限り最も遅い時刻であ



ると判断します。例えば FTP サーバーが、あるファイルの最終変更日が 1998 年 10 月 13 日であることを示すディレクトリー情報を戻したものの、そこに時刻に関する情報が含まれていなかったとすれば、プロキシーは、そのファイルの変更日時を、1998 年 10 月 13 日、午後 11 時 59 分 59 秒と判断します。そして、FTP サーバーが Windows FTP サーバー以外の場合、プロキシーはこの日付を自身の時間帯から、対応する GMT に変換します。

FTP ファイルのキャッシュ有効期限が切れた場合、プロキシーは、その FTP ファイル用に、HTTP の if-modified-since 再検査プロセスをシミュレートします。これは、要求されたファイルに対して FTP LIST コマンドを発行し直して、ファイル日付を FTP サーバーが戻した応答から解析し、その日付をファイルの最初の検索時に Last-Modified ヘッダー用にプロキシー・サーバーが生成した日付と比較します。ファイル日付が変更されていない場合には、プロキシー・サーバーはキャッシュされた FTP ファイルを再検査済みとマーク付けしてそのファイルに新しい有効期限を設定し、FTP サーバーからファイルを再検索しないで、キャッシュからファイルをサービスします。この 2 つのファイル日付が一致しない場合には、プロキシーは FTP サーバーからファイルを再検索して、新しいファイル日付で新しいコピーをキャッシュに入れます。

FTP サーバーからファイルのディレクトリー情報を入手することが、いつでも可能というわけではありません。プロキシーが FTP ファイルの日付を判別できない場合は、そのファイルの "Last-Modified" ヘッダーを生成することはありません。代わりに、URL に一致する、CacheDefaultExpiry ディレクティブ用に指定された値を使用して、そのファイルをキャッシュに保持しておく時間を決定します。この時間枠が期限切れになると、プロキシーはいつでも、FTP サーバーからファイルを再検索します。キャッシュ内の特定の FTP ファイルがしばしば CacheDefaultExpiry ディレクティブを使用して、頻繁に検索している (大量のネットワーク・トラフィックを生成している) ように見える場合には、その特定のファイルについてきめ細かい CacheDefaultExpiry の値の指定を考慮してください。これを行うことで、それらはより長時間キャッシュで保持されます。

構成および管理フォームでキャッシュ有効期限設定を指定するには、「**キャッシュ構成**」→「**キャッシュ有効期限設定**」→「**キャッシュ・ファイルの時間制限**」フォームを使用します。キャッシュ・ファイルの有効期限の設定に関する詳細は、91 ページの『ファイルの有効期限』を参照してください。

## キャッシュの新しさの構成

キャッシュ・ファイルの有効期限時間を指定するには、「構成および管理」フォームで、「**キャッシュ構成**」→「**キャッシュ有効期限設定**」を選択します。次のフォームが役立ちます。

### URL ベースの有効期限

このフォームでは、URL に基づいてファイルがキャッシュに保持される最小時間を設定します。このフォームでは、各種の URL 要求テンプレートに対してさまざまなキャッシュ動作を指定できます。

プロキシー構成ファイルを編集して URL ベースのファイル有効期限を設定するには、

185 ページの『付録 B. 構成ファイル・ディレクティブ』の次のディレクティブについての解説セクションを参照してください。

- 208 ページの『CacheMinHold - ファイルを使用可能に保つ期間を指定する』

## デフォルトの有効期限の設定

「キャッシュ有効期限設定」フォームを使用して、使用済みファイルまたは未使用ファイルのデフォルトの有効期限設定値を指定します。別々の値を HTTP、FTP、および Gopher ファイルに設定することができ、使用済みファイルまたは未使用ファイルにも別々の値を設定することができます。

このフォームには、次のような別のファイル有効期限オプションもあります。

- **キャッシュ・ファイルの有効期限検査を使用可能にする。**このチェック・ボックスはデフォルトで選択されます。一般に、失効したコンテンツをサーバーが送信しないように、このオプションを選択することが望ましいです。
- **リモート・サーバーからのファイル検索を使用不可にする。**サーバーにリモート・サーバーからファイルを検索させない場合は、このオプションを選択してください。
- **有効期限が切れる直前のファイルをキャッシュに入れない。**短時間で有効期限が切れるファイルをキャッシュに入れるのを避けるには、このオプションで時間枠を指定します。デフォルトでは、10 分以内に有効期限が切れるファイルはキャッシュに入れられません。

プロキシ構成ファイルを編集してデフォルトの有効期限設定値を設定するには、以下のディレクティブについての解説ページを参照してください。

- 203 ページの『CacheDefaultExpiry - デフォルトのファイル有効期限時間を指定する』
- 204 ページの『CacheExpiryCheck - サーバーが有効期限切れファイルを戻すかどうかを指定する』
- 211 ページの『CacheTimeMargin - ファイルをキャッシングする場合の最小存続時間を指定する』
- 212 ページの『CacheUnused - 未使用キャッシュ・ファイルの保持期間を指定する』
- 209 ページの『CacheNoConnect - スタンドアロン・キャッシュ・モードを指定する』

## 最終変更係数の設定

「最終変更係数」フォームを使用して、ヘッダーに有効期限のないキャッシュ・ファイルの有効期限をプロキシで計算するために使用する値を設定します。各種の要求テンプレートと一致するファイルに異なる値を設定できます。有効期限を計算するために、最初に一致したテンプレートが使用されます。

プロキシ構成ファイルを直接編集して最終変更係数を設定するには、205 ページの『CacheLastModifiedFactor - 有効期限を決定する値を指定する』を参照してください。

## キャッシュの時間制限

「キャッシュ・ファイルの時間制限」構成フォームでは、ファイルがキャッシュ内に残っていることのできる最長時間を設定します。時間制限は、要求テンプレートに基づいて設定され、時間制限が期限切れになったときにそのファイルを破棄するか、または妥当性を再検査するかを指定できます。この設定は、有効期限が無効になっていたり、有効期限時間が非常に長いファイルを保守するために使用することができます。

プロキシ構成ファイルを編集してキャッシュ・ファイルの最長有効期限時間の制限を設定するには、以下を参照してください。

- 207 ページの『CacheMaxExpiry - キャッシュ・ファイルの最大存続時間を指定する』
- 202 ページの『CacheClean - キャッシュされたファイルの保持期間を指定する』

---

## ガーベッジ・コレクション

頻繁にアクセスされる URL をキャッシュして、システム・リソースの使用量を最小にする努力の一環として、Caching Proxy は、ガーベッジ・コレクション と呼ばれるクリーンアップ・プロセスを実行します。このプロセスでは、古いファイルや使われないファイルがキャッシュから取り除かれて、より新しいファイル用のスペースが作られます。

ガーベッジ・コレクション・プロセスでは、キャッシュ・ディレクトリー内のファイルを調べ、有効期限が切れたファイルの除去を試みて、キャッシュのサイズを削減して新しいファイル用にスペースを作り出します。ガーベッジ・コレクションは自動的に実行されますが、一部の設定値を構成し、要件に合わせてこのプロセスを調整することができます。

## ガーベッジ・コレクションの設定

ガーベッジ・コレクションを構成するには、「構成および管理」フォームで、「キャッシュ構成」->「ガーベッジ・コレクション設定」を選択します。このフォームを使用して、「最高水準点」および「最低水準点」を設定することにより、ガーベッジ・コレクションの開始時と停止時を決定します。キャッシュ内の使用済みスペースが最高水準点に設定したパーセンテージ以上になると、ガーベッジ・コレクションが開始されます。キャッシュ内の使用済みスペースのパーセンテージが最低水準点に設定した値以下になるまで、ガーベッジ・コレクションは継続されます。

2 つのガーベッジ・コレクション・アルゴリズムから選択することができます。

**responsetime** アルゴリズムでは、大きいファイルを優先的にキャッシュから除去することによってユーザーへの応答に必要な時間を最適化します。**bandwidth** アルゴリズムでは、小さいファイルを優先的にキャッシュから除去することによってネットワーク帯域幅の使用を最適化します。いずれかを選択するか、または両方を混合して使用してください。

プロキシ構成ファイルを編集してガーベッジ・コレクションを構成するには、以下のディレクティブについての解説セクションを参照してください。

- 236 ページの『Gc - ガーベッジ・コレクションを指定する』



- 237 ページの『GcHighWater - ガーベッジ・コレクションをいつ開始するかを指定する』
- 237 ページの『GcLowWater - ガーベッジ・コレクションをいつ終了するかを指定する』
- 202 ページの『CacheAlgorithm - キャッシュ・アルゴリズムを指定する』



---

## 第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成

ほとんどの Caching Proxy サーバーは、ユーザーがファイルを要求した後でそのファイルをキャッシュに入れるだけです。Caching Proxy には、自動キャッシュ・プリロードを行うキャッシュ・エージェントがあります。キャッシュ・エージェントが、指定された URL または最も頻繁にアクセスされる URL、あるいはその両方を自動的に検索し、これらの URL が要求される前にこれらをキャッシュに入れるように指定することができます。

場合によっては、プロキシー・サーバーのホスト名を設定して、キャッシュをプリロードする前にキャッシュ・アクセス・ログを識別する必要があります。キャッシュ・エージェントを構成するには、「構成および管理」フォームで、「**キャッシュ構成**」を選択し、「**キャッシュ・プリロード**」および「**キャッシュ・リフレッシュ**」フォームを使用してください。照会の結果を表すファイル (つまり、その URL に疑問符文字 (?) が含まれているファイル) がキャッシュに入れられるのは、照会キャッシュが使用可能な場合だけであることに注意してください。

自動キャッシュ・リフレッシュおよびプリロードには、以下の利点があります。

- ユーザーがページを要求する前に、指定した URL にキャッシュが適用される。
- サーバーがユーザー活動で使用中になる前にキャッシュが取り込まれる。
- 現行ファイルが最初の要求で取り出された場合よりも迅速に、キャッシュからユーザーに提供される。

欠点としては次のものがあげられます。

- ユーザー活動が少ない時間中でも、プロキシー・サーバーはページのキャッシングでビジー状態になる。
- 自動的にロードすべきページの制御を用いなければならない。リンクされたファイルを Web 索引や検索サイトなどの上位ページからロードすると、多数のページに対する要求が生成されることがある。

効率を最適にするために、キャッシュ・エージェントを、サーバーの活動が少ないときや、クライアントの要求でビジーになる前に実行するように設定してください。その場合、ユーザーがはじめてファイルを要求したときに高速に実行できるように、キャッシュ内でファイルが使用可能状態になっています。デフォルトでは、キャッシュ・エージェントは、毎日、地方時の午前 3 時に開始します。

### リバース・プロキシー構成の特別な考慮事項

リバース・プロキシー構成を使用するときは、デフォルトで Proxy http:\* ルールを無効にしてください。これは、セキュリティ上の理由で必要となります。(つまり、このルールは ibmproxy.conf ファイル内でコメント化されます。) ただし、このルールを無効にすると、キャッシュ・エージェントは、要求の送信や Caching Proxy のキャッシュ内容のリフレッシュを、正常に行うことができなくなります。エラー・ログに「403 Forbidden By Rule Error」が示され、キャッシュのリフレッシュが完了しません。

この問題を回避するには、cacheAgentService を使用します。これは、Caching Proxy が提供している内部サービスです。このサービスを有効にするには、ibmproxy.conf ファイルのその他のマッピング・ルールの前に、次の Service ディレクティブを書き込みます。

```
Service    /any-valid-string*  INTERNAL:cacheAgentService
```

any-valid-string 変数は、ibmproxy.conf ファイルの他のマッピング・ルールと競合しない、有効な任意のストリングです。

Caching Proxy およびキャッシュ・エージェントの両方が、このサービス・ディレクティブを基にして URI の解析を行います。URI を直接 Caching Proxy に送信する代わりに、キャッシュ・エージェント・ユーティリティーは、このサービス・ディレクティブで URI の前に /any-valid-string パターンを付加します。

たとえば、キャッシュ・エージェントは、

```
http://www.ibm.com/
```

この URI を、次のように変換します。

```
/any-valid-string/http://www.ibm.com/
```

キャッシュ・エージェントは、この接頭部の付いた URI を Caching Proxy に送信します。Caching Proxy は、この要求を受け取ると、接頭部 /any-valid-string/ を除去します。残りの URI が完全修飾されたユニットになっていれば、Caching Proxy は、他のルールに対して URI をマッピングすることなく、この要求にサービスを直接提供します。

さらに、キャッシュ・エージェントは、関連する URI を Caching Proxy に送信することもできます。たとえば、ibmproxy.conf ファイルで前に参照したサービス・ディレクティブを使用して、LoadURL /abc/ を追加すると、キャッシュ・エージェントはそれを /any-valid-string/abc/ に変換し、Caching Proxy に送信します。Caching Proxy は URL を受け取ると、接頭部を除去し、他のマッピング・ルールに対して /abc/ をマップし、一致がある場合はその要求を処理します。

Service ディレクティブについては、299 ページの『Service - サービス・ステップをカスタマイズする』を参照してください。

---

## サーバーのホスト名の設定

Linux および UNIX プラットフォームでは、キャッシュがプリロードまたはリフレッシュされるプロキシ・サーバーのホスト名を指定します。Windows プラットフォームでは、リフレッシュされるプロキシ・サーバーがローカル・マシン上にない場合だけホスト名を指定してください。(ローカル・キャッシュ・エージェントはリモート・サーバーのキャッシュ・アクセス・ログにアクセスしないので、頻繁にアクセスするファイルに基づいてリモート・サーバーのキャッシュをリフレッシュすることはできないことに注意してください。)

プロキシ・サーバーのホスト名を設定するには、「構成および管理」フォームで、「キャッシュ構成」->「キャッシュ・リフレッシュ」の「キャッシュ宛先サーバーの識別」を選択します。

---

## キャッシュへの特定のファイルのプリロード

キャッシュに特定の URL に保管されているコンテンツをプリロードするには、「構成および管理」フォームで、「**キャッシュ構成**」→「**キャッシュ・プリロード**」を使用します。このフォームで、キャッシュ・エージェントがロードする URL を指定できます。プロキシは、そのページが以前にキャッシュされていたかどうかに関係なく、キャッシュ・エージェントの開始時にそのページを検索します（これらの URL は、LoadURL ディレクティブによってプロキシ構成ファイルに指定されます）。また、このフォームは、そのコンテンツをキャッシュに入れない URL を定義する場合にも使用できます。このタイプのキャッシュ・プリロードでは、キャッシュ・アクセス・ログへのアクセスは不要です。

「**キャッシュ・プリロード**」フォームを使用して、次のオプションを構成します。

- **毎日キャッシュをリフレッシュする** — キャッシュ・エージェントに毎晩キャッシュをリフレッシュさせる場合は、このボックスにチェック・マークを付けます。キャッシュ・エージェントを開始させたくない場合には、このボックスにチェックが入っていないことを確認してください。
- **キャッシュ・リフレッシュの時刻** — キャッシュ・エージェントが地方時の午前 3 時以外の時刻に実行されるようにするには、キャッシュ・エージェントを開始させる時刻を指定します。
- **キャッシュのコンテンツ** — 「**URL または IP アドレス**」フィールドで、ロードする URL を指定します。URL のプリロードを除外するには、URL を指定し、「**キャッシュ状況**」ボックスで「**無視**」をクリックします。

---

## キャッシュへの頻繁にキャッシュされるファイルのプリロード

最もアクセス頻度の高いページを自動的にプリロードするには、「**キャッシュ構成**」→「**キャッシュ・リフレッシュ**」フォームを使用します。この機能には、プロキシ・サーバーのキャッシュ・アクセス・ログが必要です。（ログの位置と名前は変更可能です。157 ページの『第 6 部 Caching Proxy のモニター』を参照してください。）頻繁にアクセスされる URL は、キャッシュ・アクセス・ログから自動的に判別されます。管理者は、キャッシュにプリロードする頻繁にアクセスされるページの数指定することもできます。（この数は LoadTopCached ディレクティブによってプロキシ構成ファイルに指定されます。）

「**キャッシュ・リフレッシュ**」フォームを使用して、次のオプションを構成します。

- **毎日キャッシュをリフレッシュする** — キャッシュ・エージェントに毎晩キャッシュをリフレッシュさせる場合は、このボックスにチェック・マークを付けます。キャッシュ・エージェントを開始させたくない場合には、このボックスがクリアにされていることを確認してください。
- **キャッシュ・リフレッシュの時刻** — キャッシュ・エージェントを午前 3 時以外の時刻に実行するには、キャッシュ・エージェントを開始させる時分 (HHMM) を指定します。
- **キャッシュ宛先サーバーの識別** — ローカル・マシン以外のサーバーをリフレッシュする場合は、このオプションを使用します。（特定のファイルへのアクセス頻度に基づいてリモート・サーバーをリフレッシュすることはできないことに注意してください。）

- **頻繁にアクセスされる URL のキャッシュ** — 前夜のキャッシュ・アクセス・ログからキャッシュに入れる URL の数を指定します。
- **リンク・ページのロード** — この設定は、探求機能の構成に使用します (探求機能の詳細については、次のセクションを参照してください)。探求するレベルの数を設定し、すべてのページを探求するか (**always**)、どのページも探求しないか (**never**)、管理者が指定したページだけを探求するか (**admin**)、または頻繁にアクセスされるページだけを探求するか (**topn**) について設定します。また、ホスト間を探求するかどうか、要求と要求の間に遅らせるかどうか、およびインライン・イメージをキャッシュに入れるかどうかについても指定します。
- **スレッドの数** — キャッシュ・リフレッシュに使用するスレッドの最大数を設定します。
- **作業キューの最大長** — 要求する URL 用のキューの最大長を設定します。
- **要求する URL の最大数** — ロードするページの最大数を設定します。ページ探求の検索が開始される前に、この数が検査されます。
- **最長の時間** — キャッシュ・エージェントを実行する最長の時間を設定します。この時間を 0 時間 0 分に設定すると、キャッシュ・エージェントは、完了するまで実行されます。

---

## 探求

探求 とは、自動キャッシュ・リフレッシュ機能のうちのオプション部分です。ほとんどの Web ページには、関連情報を含んでいる他のページへのリンクがあり、ユーザーは、通常、あるページから別のページへとリンクしたり、あるサイトから別のサイトへとリンクするパスをたどります。探求は、このような論理情報パスをキャッシュに入れる手段となります。探求機能では、キャッシュ・エージェントは、ロードするページの指定されたレベルのハイパーテキスト (HTML) リンクをたどって、そのリンク・ページすべてをキャッシュに入れます。リンクされたページは、ソース・ページと同じホストに常駐する場合でも、他のホストに常駐する場合でもかまいません。103 ページの図 1 でこのことについて説明します。

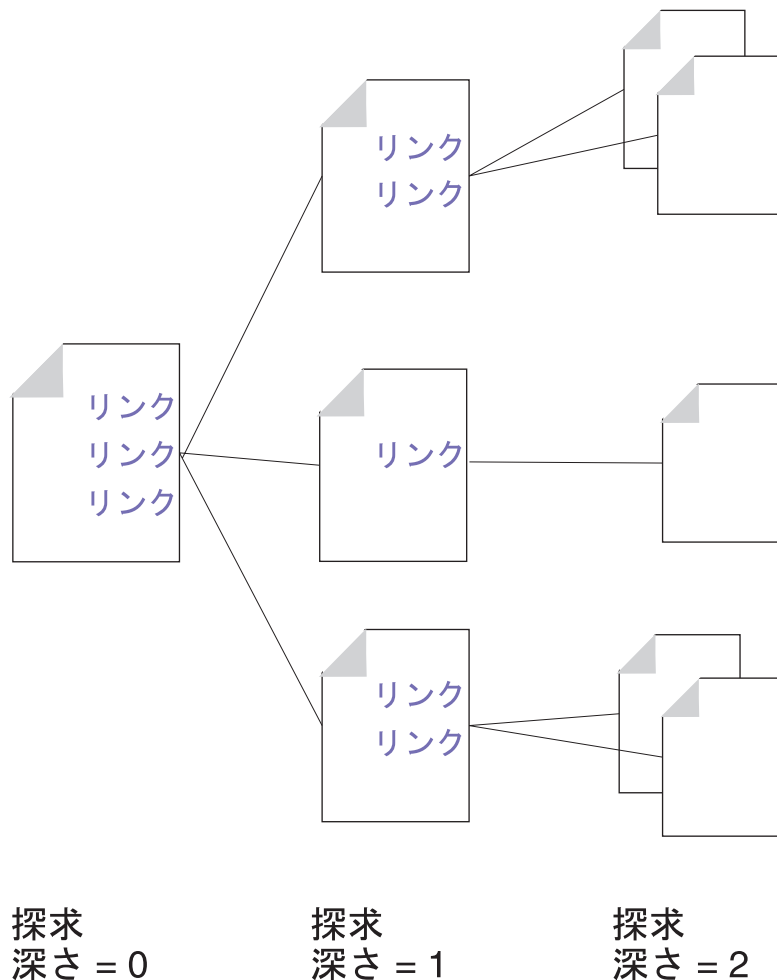


図 1. 探求

探求プロセスを制御するために、管理者はキャッシュ・エージェントにロードできる URL の最大数 (デフォルトの設定は 2000)、実行可能な最長時間 (デフォルトの設定は 2 時間)、および使用できるスレッドの最大数 (デフォルトの設定は 4) を指定します。管理者は追加の制御も構成することができます。デフォルトでは、探求機能は 2 レベルの階層で使用可能になっていますが、ホスト間では許可されません。さらに、要求間に遅延を挿入します。これらの設定値を変更するには、104 ページの『関連したプロキシー構成ファイル・ディレクティブ』を参照してください。

キャッシュ・エージェントは、以下の順序でキャッシュをロードしたあと、リフレッシュします。

1. 管理者が指定した特定のページをロードします。
2. キャッシュ・アクセス・ログの一般的な (頻繁にアクセスされる) ページをロードします。
3. この時点で最大ページ数に達していない場合には、探求機能によって追加のページがロードされます。

キャッシュ・エージェントは、リンク間の探求を開始するまではページの最大数にすでに達しているかどうかについて検査しません。ページ数の最大値 (プロキシー構成ファイル内で **MaxURL** と呼ばれる) がステップ 1 および 2 で検索されるページの数より小さい場合は、リンクされたページはまったく検索されません。

以下の例は、指定された URL の最大数に関連して、キャッシュ・エージェントがキャッシュ・リフレッシュの優先順位と探求をどのように処理するかを示しています (以下の例すべてに探求機能が構成されているものとします)。

構成ファイルの設定	結果
LoadURL http://www.getthis.com/main.html LoadURL http://www.getmetoo.com/welcome.htm LoadTopCached 30 MaxURLs 50	キャッシュ・アクセス・ログに 31 以上の固有の URL がある場合、キャッシュ・エージェントは、main.html と welcome.htm を検索し、キャッシュ・アクセス・ログに基づいた上位 30 の要求された URL を検索します。MaxURL 値にはまだ達していないので、キャッシュ・エージェントは、リンクされた URL を最大 18 まで、すでにキャッシュに格納されているページから検索してロードします。
LoadURL http://www.joesmith.edu/favorites.html LoadURL http://www.janesmith.edu/dislikes.html LoadTopCached 30 MaxURLs 25	キャッシュ・アクセス・ログに 31 以上の固有の URL がある場合、キャッシュ・エージェントは、favorites.html と dislikes.html を検索し、キャッシュ・アクセス・ログから上位 30 の要求された URL を検索します。MaxURL の値をすでに超えているので、その他のファイルは検索されません。
LoadURL http://www.hello.com/hi.htm LoadURL http://www.ballyhoo.com/index.html LoadTopCached 20 MaxURLs 25	キャッシュ・アクセス・ログに 21 以上の固有の URL がある場合、キャッシュ・エージェントは、hi.htm と index.html を検索し、キャッシュ・アクセス・ログから上位 20 の要求された URL、および以前のページから最高 3 つまでのリンクされた URL を検索します。MaxURL の値にすでに達しているので、その他のファイルは検索されません。

## 関連したプロキシー構成ファイル・ディレクティブ

キャッシュ・エージェントは、プロキシー構成ファイルで適切なディレクティブを直接編集して構成することもできます。キャッシュ・エージェントに関連したプロキシー構成ファイルのディレクティブについては、185 ページの『付録 B. 構成ファイル・ディレクティブ』の以下の解説ページを参照してください。

- 200 ページの『AutoCacheRefresh - キャッシュ・リフレッシュを使用するかどうかを指定する』
- 201 ページの『CacheAccessLog - キャッシュ・アクセス・ログ・ファイルのパスを指定する』
- 211 ページの『CacheRefreshTime - キャッシュ・エージェントをいつ開始するかを指定する』



- 219 ページの『DelayPeriod - 要求間の一時停止を指定する』
- 220 ページの『DelveAcrossHosts - ドメイン間のキャッシュへの格納を指定する』
- 220 ページの『DelveDepth - キャッシュへの格納中にリンクをどこまで追跡するかを指定する』
- 220 ページの『DelveInto - キャッシュ・エージェントがリンクをたどるかどうかを指定する』
- 242 ページの『IgnoreURL - リフレッシュしない URL を指定する』
- 249 ページの『LoadInlineImages - 組み込みイメージのリフレッシュを制御する』
- 249 ページの『LoadTopCached - リフレッシュを実行する頻繁にアクセスされるページの数指定する』
- 250 ページの『LoadURL - リフレッシュする URL を指定する』
- 259 ページの『MaxUrls - リフレッシュする URL の最大数を指定する』

---

## キャッシュ・エージェントの手動による開始

自動キャッシュ・リフレッシュが使用可能になっていると、キャッシュ・エージェントは指定された時刻に自動的にリフレッシュ操作を実行します。しかし、キャッシュ・エージェントは、任意の時刻にコマンド行から実行することもできます。

実行可能ファイルは、次のとおりです。

- Linux および UNIX プラットフォームの場合: `usr/sbin/cacheagt`
- Windows プラットフォームの場合: `server_root¥bin¥cacheagt.exe`

ここでは、`server_root` は、Caching Proxy をインストールしてあるドライブおよびディレクトリー (たとえば、`C:¥Program Files¥IBM¥edge¥cp`) になります。

Linux および UNIX プラットフォームでは、**cron** デーモンを使用することにより、さまざまな時刻にキャッシュ・エージェントを自動的に実行できます。**cron** で制御されるジョブは、システムの `crontab` ファイルに行を追加することによって指定します。次に、Linux および UNIX 上のコマンド・ファイルの項目の例を示します。

```
45 16 * * * /usr/sbin/cacheagt
```

このコマンド例は、地方時で毎日の午後 4 時 45 分にキャッシュ・エージェントを開始します。複数の項目を使用して、希望すれば、複数回キャッシュ・エージェントを実行することができます。詳しくは、使用しているオペレーティング・システムの資料で **cron** デーモンについて参照してください。

**cron** デーモンを使用してキャッシュ・エージェントを実行する場合には、「**キャッシュ構成**」→「**キャッシュ・リフレッシュ**」構成フォームを使用するか、またはプロキシ構成ファイルを編集して、自動リフレッシュ・オプションをオフにすることを忘れないでください。そうしないと、キャッシュ・エージェントが毎日複数回実行されることになります。



---

## 第 21 章 共用キャッシュの使用

Web 上のある地点で単一のサーバーでは処理しきれない量のトラフィックが発生することはよくあることです。解決策の 1 つは、単にサーバーをさらに追加することです。しかし、複数の Caching Proxy サーバーを使用すると、あるキャッシュのコンテンツが他のキャッシュのコンテンツと重複することがよくあります。記憶域に不要な冗長部分が生じるだけでなく、帯域幅の節約を最大化することができなくなります。これは、キャッシュ・ファイルの取り出し要求が、独自のキャッシュにそのファイルを持たないプロキシ・サーバーに対して出された時に、そのファイルが起点サーバーから再び取り出されるためです。階層チェーンのプロキシ・サーバーを使用すると重複キャッシュを最小限に抑えることができますが、この方法でも特定のサーバーを経由するトラフィックが増える結果となり、チェーン内のリンクが追加されるたびに待ち時間が長くなります。

キャッシュ共用では、各キャッシュが他のキャッシュと共にそのコンテンツを共用できるので、このような問題が解消されます。帯域幅は、以下の理由から節減されます。

- オブジェクトの複数回取り出しが行われません。
- 結合された論理キャッシュが大きければ大きいほど、ヒット率が高くなります。

複数のキャッシュを 1 つの論理キャッシュであるかのように使用するには、以下の 2 とおりの方法があります。

- リモート・キャッシュ・アクセス (RCA) は、メンバー・キャッシュの配列を定義する Caching Proxy の機能です。ファイルは内部論理に基づいて、これらのキャッシュの 1 つだけに保管されます。
- Caching Proxy プラグインでは、プロキシ・サーバーがインターネット・キャッシング・プロトコル (ICP) を使用できます。Caching Proxy マシンと Caching Proxy 以外のキャッシュ間でデータを共用したい場合は、RCA の代わりに ICP プラグインを使用することもできます。

RCA と ICP は併用できます。

---

### リモート・キャッシュ・アクセス

RCA の計画では、以下の勧告を考慮に入れてください。

- 関係する プロキシ・サーバー は、互いに近くにあって、高帯域幅リンク (例えば、FDDI、SP2 バス) で接続する必要があります。
- RCA 配列のメンバーシップは、構成を可能な限り継続的にするために、長期にする必要があります。
- プロキシ・サーバーは、同様な能力 (例えば、CPU、メモリー・サイズ、キャッシュ・サイズ) を備えている必要があります。
- ネットワーク停止はほとんどが起らないようにする必要があります。
- どの配列でも、メンバーは 100 未満でなければなりません。

- 配列のすべてのメンバーは、同じバージョンの Caching Proxy ソフトウェアを使用する必要があります。

注: RCA 配列のプロキシが異なる Linux オペレーティング・システム (例えば、SUSE および Red Hat) を使用している場合には、“nobody” ユーザーがすべてのピアで同じ UID を持っていることを確認してください。各コンピューターの /etc/ ディレクトリー内のパスワードとグループ・ファイルの項目を調べて、同じ UID を “nobody” に割り当ててください。

これらの条件のいずれかに違反しているか、あるいは別の組織がこの配列のメンバーである別のサーバーを管理している場合には、リモート・キャッシュ・アクセスは適切ではありません。

## リモート・キャッシュ・アクセスの構成

リモート・キャッシュ・アクセスを構成するには、「構成および管理」フォームで、「キャッシュ構成」→「リモート・キャッシュ・アクセス」を選択します。このフォームには、1 つの論理キャッシュを共用する名前付き配列を定義するフィールドがあります。配列のメンバーごとに、必要な情報を入力してください。

プロキシ構成ファイルを編集してリモート・キャッシュ・アクセスを構成するには、185 ページの『付録 B. 構成ファイル・ディレクティブ』の以下のディレクティブについての解説セクションを参照してください。

- 198 ページの『ArrayName - リモート・キャッシュ配列を指定する』
- 259 ページの『Member - 配列のメンバーを指定する』

---

## インターネット・キャッシング・プロトコル・プラグインの構成

インターネット・キャッシング・プロトコル・プラグインによって Caching Proxy は、HTML ページおよびその他のキャッシュ可能リソースの検索で、ICP に準拠したキャッシュを照会できます。プロキシ・サーバーは HTTP 要求を受け取ると、独自のキャッシュでそのリソースを検索します。ローカル・キャッシュにそのリソースが見つからず、ICP プラグインが使用可能な場合には、プロキシ・サーバーは ICP 照会パケット内の URL 要求をカプセル化して、識別されたすべての ICP PEER キャッシュにこのパケットを配布します。PEER キャッシュがリソースを持っていると応答すると、プロキシ・サーバーはその PEER のキャッシュからリソースを検索します。複数の PEER が肯定的な応答をした場合は、最初の応答が処理されます。ヒットで応答する PEER がない場合には、元のサーバーはそのワークフローに従って要求の処理を続行します。例えば、プロキシ・サーバーがもう 1 つのプラグインを呼び出し、リモート・キャッシュ・アクセス・ルーチンへ継続するか (RCA が使用可能な場合)、または要求されたリソースそのものを取り出すことができます。

## ICP プラグインの構成

ICP プラグインは、プロキシ構成ファイル `ibmproxy.conf` を編集することによって、活動化され、構成されます。ICP プラグインを使用するためには、`ServerInit` ディレクティブまたは `PreExit` ディレクティブ、あるいはその両方を構成ファイルの

API ディレクティブ・セクションに追加する必要があります。どちらのディレクティブが使用されるかは、ICP システム内で Caching Proxy に割り当てられている役割に応じて異なります。

- Caching Proxy を ICP サーバーとして機能させるには、ServerInit ディレクティブを使用して、icpServer モジュールを呼び出します。
- Caching Proxy を ICP クライアントとして機能させるには、PreExit ディレクティブを使用して、icpClient モジュールを呼び出します。
- Caching Proxy を ICP クライアントと ICP サーバーの両方として機能させるには、両方のディレクティブを使用します。
- プラグインが使用する設定を構成するには、ディレクティブ icpAddress、icpMaxThreads、icpPeer、icpPort、および icpTimeout を使用します。

これらのディレクティブを作成するには、ibmproxy.conf ファイルを手作業で編集するか、あるいはプロキシ・サーバーがすでに稼働中の場合には「構成および管理」フォームの「サーバー構成」->「要求処理」->「API 要求処理」に接続します。

ibmproxy.conf ファイルの API セクションには、(コメントの形で) プロトタイプ・ディレクティブが追加されていることに注意してください。この API ディレクティブは目的別の順序で配列されています。API ディレクティブを追加して新しい機能やプラグイン・モジュールを使用できるようにするには、各ディレクティブを構成ファイルのプロトタイプ・セクションに示されているように配列してください。あるいは、必要に応じて API ディレクティブをアンコメントして編集し、それぞれ必要な機能やプラグインに対するサポートを組み込んでください。

ServerInit および PreExit ディレクティブには 2 つの引き数があります。(1) 共用ライブラリーの完全修飾パスと (2) 関数呼び出しです。これらの引き数はコロン (:) で区切られます。最初の引き数はシステム特有で、プラグイン・コンポーネントのインストール場所によって決まります。2 番目の引き数は共用ライブラリーにハードコーディングされるので表示されたとおりに入力する必要があります。

プロキシ構成ファイルでは各ディレクティブは単一行にある必要があります。

```
ServerInit path_of_shared_library:icpServer
```

Linux および UNIX の例:

```
ServerInit /opt/ibm/edge/cp/internet/lib/plugins/icp/libicp_plugin.so:icpServer
```

Windows の例:

```
ServerInit C:\Program Files\IBM\edge\cp\Bin\plugins\icp\icpplugin.dll:icpServer
```

```
PreExit path_of_shared_library:icpClient
```

Linux および UNIX の例:

```
PreExit /opt/ibm/edge/cp/internet/lib/plugins/icp/libicp_plugin.so:icpClient
```

Windows の例:

```
PreExit C:\Program Files\IBM\edge\cp\Bin\plugins\icp\icpplugin.dll:icpClient
```

プラグインの設定を構成するには、プロキシ構成ファイルで提供されている ICP\* ディレクティブを追加または変更します。詳しくは、次のディレクティブの説明を参照してください。

- 240 ページの『ICP\_Address - ICP 照会用の IP アドレスを指定する』
- 240 ページの『ICP\_MaxThreads - ICP 照会用の最大スレッド数を指定する』
- 240 ページの『Occupier - ICP クラスターのメンバーを指定する』
- 241 ページの『ICP\_Port - ICP 照会用のポート番号を指定する』
- 241 ページの『ICP\_Timeout - ICP 照会に対する最大待機時間を指定する』
- 273 ページの『PreExit - PreExit ステップをカスタマイズする』
- 297 ページの『ServerInit - サーバー初期化ステップをカスタマイズする』

---

## 第 22 章 動的に生成されたコンテンツのキャッシング

これは、リバース・プロキシ構成にのみ適用されます。

動的キャッシング機能により、Caching Proxy は IBM WebSphere Application Server によって生成された JavaServer Pages (JSP) およびサーブレットからの応答の形式で、動的に生成されたコンテンツをキャッシュできます。Caching Proxy アダプター・モジュールは、応答がアプリケーション・サーバーの動的キャッシュにキャッシュされるのに加えてプロキシ・サーバーでもキャッシュされるように、アプリケーション・サーバーで応答を変更するのに使用されます。この機能を使用すると、動的に生成されたコンテンツがネットワークのエッジでキャッシュされるため、複数のクライアントが同じコンテンツを要求する場合に、コンテンツ・ホストはアプリケーション・サーバーに要求を繰り返すことから解放されます。

**注:** 動的キャッシュ機能では、URL 照会からの結果をキャッシュに入れるためにプロキシ・サーバーが使用可能になりません。照会結果をキャッシュに入れるには、キャッシュ・フィルターを構成します。キャッシュ・フィルターについては、87 ページの『第 18 章 キャッシュ対象の制御』、および 210 ページの『CacheQueries - 疑問符 (?) を含む URL へのキャッシュ応答を指定する』のディレクティブ解説セクションを参照してください。IBM WebSphere Application Server でない起点サーバーからの照会の結果は、キャッシュに入れることができます。

動的キャッシュ機能を機能させるには、場合によっては、照会キャッシュを使用可能にする必要があります。例えば、サーブレットが照会のフォームで URL を使用する場合などです。プロキシ・サーバーでは、疑問符 (?) が含まれている URL を照会として扱います。

動的に生成されたコンテンツのキャッシングにより、以下のような利点が得られます。

- コンテンツ・ホストの負荷を軽減します。
- Application Server の負荷を軽減します。
- 要求されたリソースをエンド・ユーザーに迅速に送達します。
- サーバー間の帯域幅使用量を軽減します。
- 動的に生成されるコンテンツを作成または供給する Web サイトのスケーラビリティが向上します。

アプリケーション・サーバーでは、プロキシ・キャッシングの場合、完全に構成された公開ページだけがエクスポートされます。プライベート・ページはプロキシによってキャッシュされません。例えば、公開サイトから動的に生成され、現在の天気予報をリストしたページは、IBM WebSphere Application Server によってエクスポートし、Caching Proxy によりキャッシュすることができます。しかし、動的に生成され、ユーザーのショッピング・カートのコンテンツをリストするページをプロキシ・サーバーによってキャッシュすることはできません。また、動的に



生成されたページをキャッシュするには、そのページのサブコンポーネントもすべてキャッシュ可能でなければなりません。

キャッシュされた動的ファイルは、通常のファイルのように有効期限が切れることはありません。これらは、これらを生成したアプリケーション・サーバーによって無効にすることができます。

次の状況では、動的キャッシュの項目が無効化されます。

- 動的キャッシュ・ガーベッジ・コレクターがキャッシュの輻輳（ふくそう）によって項目を除去した場合。
- サブレット項目 (servletcache.xml) またはプロキシの ExternalCacheManager ディレクティブで設定されたタイムアウトが満了した場合。
- 外部エージェントまたはアプリケーションが動的キャッシュ API を起動してキャッシュ項目を無効にした場合。

動的キャッシュ項目の無効化は、Caching Proxy 動的キャッシング・プラグインの特定のインスタンスに対する無効化 (Invalidate) メッセージの生成によって行われます。Caching Proxy は、/WES\_External\_Adapter リソース・ロケーターへの通知として無効化 (Invalidate) メッセージを受信します。Caching Proxy は、次に、無効な項目をそのキャッシュから消去します。

動的キャッシングには次の構成ステップが必要です。

- IBM WebSphere Application Server 構成
  - ローカル動的キャッシングを行うようにアプリケーション・サーバーを構成する。
  - 外部キャッシュ・アダプターを使用するようにアプリケーション・サーバーを構成する。
  - キャッシュ可能なサブレットおよび JSP ファイルごとに、使用できる外部キャッシュを指定する。
- Caching Proxy 構成
  - Caching Proxy が動的キャッシング・プラグインを使用できるようにする。
  - 動的コンテンツがキャッシュされる送信元を指定する。

---

## プロキシ・キャッシングのための IBM WebSphere Application Server の構成

### アプリケーション・サーバーでの動的キャッシングの構成

ローカル動的キャッシュ（動的フラグメント・キャッシュとも呼ばれる）を使用するようにアプリケーション・サーバーを構成する場合は、IBM WebSphere Application Server 資料の説明に従ってください。動的フラグメント・キャッシュは Application Server Caching Proxy の外部キャッシュと相互作用します。

### アプリケーション・サーバー・アダプターの構成

IBM WebSphere Application Server は、Application Server と共にインストールされる外部キャッシュ・アダプター (External Cache Adapter) と呼ばれるソフトウェア・モジュールを使用して、Caching Proxy と通信します。



注: 動的キャッシュの構成に関するテクニカル・ノートについては、IBM WebSphere Application Server サポートの Web サイトを参照してください。

## 動的キャッシングのための Caching Proxy の構成

動的に生成されたコンテンツ (サーブレットおよび JSP からの結果) をプロキシ・サーバーでキャッシュ可能にするには、プロキシ構成ファイル `ibmproxy.conf` に 2 つの変更を行う必要があります。最初の変更は、動的キャッシング・プラグイン・モジュールを使用可能にすることで、2 番目の変更は、キャッシュ可能な動的コンテンツの送信元をこのモジュールが認識するように構成することです。

### 動的キャッシング・プラグインを使用可能にするための Service ディレクティブの設定

Service ステップの API ディレクティブは、動的キャッシング・プラグインを使用可能にするのに使用されます。このディレクティブを作成するには、`ibmproxy.conf` ファイルを手作業で編集するか、あるいはプロキシ・サーバーがすでに実行中であれば、「構成および管理」フォームを使用して、「サーバー構成」->「要求処理」->「API 要求処理」を選択します。ディレクティブの内容は、このセクションの後の方にある例で示されます。

動的キャッシングを使用可能にする Service ディレクティブのプロトタイプは、`ibmproxy.conf` ファイルの API セクションにコメントとして存在しています。そこには JSP Plug-in という見出しがあります。プロトタイプの API ディレクティブは、目的別の順序で配列されていることに注意してください。API ディレクティブを追加して新しい機能やプラグイン・モジュールを使用できるようにするには、各ディレクティブを構成ファイルのプロトタイプ・セクションに示されているように配列してください。オプションで、コメント文字をプロトタイプ API ディレクティブから除去し、必要に応じてそれらを編集して、所要のそれぞれの機能またはプラグインのサポートを組み込むことができます。

次の例に示されるように、Service ディレクティブを設定します。(プロキシ構成ファイルでは、それぞれのディレクティブは単一行になければならないことに注意してください。次の例では、読みやすくするために改行されている場合があります。)

- AIX の場合:

```
Service /WES_External_Adapter /opt/ibm/edge/cp/lib/plugins/  
dynacache/libdyna_plugin.o:exec_dynacmd
```

- Solaris の場合:

```
Service /WES_External_Adapter /opt/ibm/edge/cp/lib/plugins/  
dynacache/libdyna_plugin.so:exec_dynacmd
```

- Linux の場合:

```
Service /WES_External_Adapter /usr/lib/libdyna_plugin.so:exec_dynacmd
```

- Windows の場合:

```
Service /WES_External_Adapter C:%Program Files%IBM%edge%cp%bin%plugins%  
dynacache%dyna_plugin.dll:exec_dynacmd
```

Caching Proxy ソフトウェアをデフォルト以外のディレクトリーにインストールしている場合は、これらの例のパスについて、インストール・パスを置き換えてください。

## ファイル・ソースを指定するための ExternalCacheManager ディレクティブの設定

各 Caching Proxy は動的に生成されるファイルの送信元を認識するように構成する必要があります。この プロキシ・サーバー で動的に生成されたコンテンツをキャッシュに入れるアプリケーション・サーバーごとに、ExternalCacheManager ディレクティブを `ibmproxy.conf` ファイルに追加してください。このディレクティブでは、プロキシで結果をキャッシュに入れる WebSphere Application Server が指定され、そのサーバーからのコンテンツについての最大有効期間時間が設定されます。詳細は、232 ページの『ExternalCacheManager - IBM WebSphere Application Server からの動的キャッシング用の Caching Proxy の構成』に示されています。

ExternalCacheManager ディレクティブで使用されるサーバー ID は、アプリケーション・サーバーの `dynacache.xml` ファイルの外部キャッシュ・グループ・スタンザで使用されるグループ ID と一致していなければなりません。

前述の例では、以下の項目を個々のプロキシの `ibmproxy.conf` ファイルに追加します。

```
ExternalCacheManager IBM-edge-cp-XYZ-1 20 seconds
```

Caching Proxy は、IBM WebSphere Application Server からそのグループ ID が `ibmproxy.conf` ファイルの ExternalCacheManager 項目と一致するコンテンツだけをキャッシュします。

---

## 第 23 章 プロキシ・サーバー・キャッシュの調整

キャッシュが使用可能な場合、キャッシュ・ストレージの速度は Caching Proxy のパフォーマンスにとって重要なことです。このセクションでは、キャッシュ・ストレージの種類を選択し、最高のパフォーマンスを得られるようにキャッシュ・ストレージを設定する方法について、提案を示します。

---

### キャッシュ・ストレージ・メディアの選択

Caching Proxy では、キャッシュ・ストレージの以下の 2 種類のメディアを使用できます。

- メモリー
- ロー・ディスク区画

メモリー・キャッシュでは、ファイルの検索速度は最高になりますが、メモリー・キャッシュのサイズはプロキシ・サーバー・マシンで使えるメモリーの容量によって制限されます。1 つまたは複数のロー・ディスク区画からなるディスク・キャッシュは、メモリー・キャッシュより低速ですが、ほとんどの場合、より大きいキャッシュ・サイズが可能になります。

---

### ディスク・キャッシュのパフォーマンスの最適化

ディスク・キャッシュに使用するデバイス区画は、キャッシュ専用にする必要があります。つまり、これらの物理ディスクを、他のファイル・システムを含めるために使用したり、プロキシ・キャッシュを格納する以外の目的で使用したりしないでください。また、パフォーマンスを低下させるので、プロキシ・キャッシュ用に使用されるディスクではデータ圧縮を使用しないでください。

キャッシュ・ストレージ (ディスクまたはファイルのいずれでも) ごとにプロキシ・サーバー上でメモリー・オーバーヘッドが発生します。一般に、1 つの物理ディスク全体を 1 つのキャッシュ・デバイスとして使用すると、最高のパフォーマンスが得られます。RAID またはその他の機構を使用し、複数の物理ディスクを結合して 1 つの論理ディスクにすると、逆効果となることがあります。複数のディスクを使用する場合は、「**キャッシュの設定**」構成フォームを使用して、またはプロキシ構成ファイル内の CacheDev ディレクティブを編集して、それらのディスクを複数のキャッシュ・デバイスとして指定します。この方式では、プロキシ・サーバーは、複数のディスクに対する読み取りと書き込みの並列性を制御することができ、オペレーティング・システムやディスク・サブシステムのパフォーマンスに依存しなくなります。

---

## キャッシュのガーベッジ・コレクション

プロキシー・サーバーのキャッシュ・ガーベッジ・コレクションでは、有効期限の切れたファイルがキャッシュから廃棄され、新規要求のファイルをキャッシュに入れるためにスペースが解放されます。キャッシュ内の使用スペースの量が最高水準点と呼ばれる管理者指定の限度に達すると、ガーベッジ・コレクションが自動的に起動され、使用スペースの量が最低水準点に達するまで処理が継続されます。

ガーベッジ・コレクション・ルーチンが使用する CPU リソースは最小限であり、キャッシュに格納後まだ有効期限切れになっていないものの可用性が影響を受けることはないので、ガーベッジ・コレクションが特定の時刻に実行されるよう設定する必要はありません。

ガーベッジ・コレクションのパフォーマンスを向上させるために、最高水準点と最低水準点を設定できます。また、ガーベッジ・コレクションのために使用されるアルゴリズムの種類を設定することもできます。ガーベッジ・コレクションの設定の変更方法の詳細については、96 ページの『ガーベッジ・コレクション』を参照してください。

---

## プラットフォーム固有の最適化

以下に示すのは、プラットフォームごとのキャッシュのパフォーマンス最適化に関するその他の提案です。

### AIX

なるべく使用可能なすべての物理区画 (PP) を使用して、1 つのディスク上に単一の論理ボリュームを作成してください。例えば、9 GB のディスクの場合、`cpcache1` という 9 GB の論理ボリュームを作成します。これをフォーマットし、そのロー論理ボリューム `/dev/rcpcache1` を使用するプロキシー・キャッシュ・デバイスとして指定します。

### HP-UX および Solaris

ディスクのサイズ全体を使用する単一の区画 (またはスライス) をキャッシュ・デバイス上に作成します。例えば、9 GB のディスク上に `c1t3d0s0` という 9 GB の区画を作成します。これをフォーマットし、そのロー・デバイス `/dev/rdisk/c1t3d0s0` を使用するプロキシー・キャッシュ・デバイスとして指定します。

### Windows

ディスクのサイズ全体を使用して単一の区画を作成します。例えば、9 GB のディスク上に `i:` という 9 GB の区画を作成します。これをフォーマットし、そのロー・デバイス `\\\\.\\i:` を使用するプロキシー・キャッシュ・デバイスとして指定します。

プロキシー・サーバーのキャッシュの構成方法、およびキャッシュ・デバイスをフォーマットして指定する方法については、75 ページの『第 4 部 プロキシー・サーバー・キャッシングの構成』を参照してください。

---

## 第 5 部 Caching Proxy セキュリティーの構成

ここには、Caching Proxy で SSL を使用する場合、暗号ハードウェアを使用できるようにする場合、IBM Tivoli® Access Manager (以前の Tivoli Policy Director) プラグイン を使用する場合、および PAC-LDAP 許可モジュールを使用する場合の、基本的なセキュリティに関する情報があります。

ここには、次の章が含まれます。

119 ページの『第 24 章 プロキシ・サーバーのセキュリティ』

121 ページの『第 25 章 サーバー保護セットアップ』

125 ページの『第 26 章 Secure Sockets Layer (SSL)』

143 ページの『第 27 章 暗号ハードウェアのサポートの使用可能化』

145 ページの『第 28 章 Tivoli Access Manager プラグインの使用』

147 ページの『第 29 章 PAC-LDAP 許可モジュールの使用』



---

## 第 24 章 プロキシ・サーバーのセキュリティー

インターネットからアクセス可能なサーバーでは、そのサーバーが稼働しているシステムに対して望ましくない行為を受ける危険性があります。許可されない人々が、パスワードの推測、ファイルの更新、ファイルの実行、あるいは機密データの読み取りを試みるかもしれません。Web の魅力の 1 つは、それが開かれていることです。しかし、Web は正当な使用も不正な使用も可能です。

以下の項では、Caching Proxy サーバー上のファイルにアクセスするユーザーを管理する方法について説明します。

Caching Proxy は Secure Sockets Layer (SSL) 接続をサポートしています。この SSL 接続では、暗号化と暗号化解除を行うセキュア伝送がクライアント・ブラウザーと宛先サーバー (コンテンツ・サーバーまたは代理サーバーのいずれか) の間で確立されます。

Caching Proxy は、代理として構成されていると、クライアント、コンテンツ・サーバー、またはその両方とセキュア接続を確立することができます。SSL 接続を使用可能にするには、「構成および管理」フォームで、「**プロキシ構成**」->「**SSL 設定**」を選択します。このフォームで、「**SSL を使用可能にする**」チェック・ボックスを選択し、鍵リング・データベースと鍵リング・データベース・パスワード・ファイルを指定します。

Caching Proxy は、フォワード・プロキシ・サーバーとして構成されていると、SSL トンネリングというパススルー・プロトコルに従って、クライアントとコンテンツ・サーバーの間で暗号化された要求の受け渡しを行います。プロキシ・サーバーはトンネル伝送の要求の暗号化を解除しないため、暗号化された情報はキャッシュには入れられません。フォワード・プロキシのインストールでは、SSL トンネリングが使用可能にされています。これを使用不可にするには、「構成および管理」フォームで、「**プロキシ構成**」->「**プロキシ設定**」を選択し、このフォーム上で「**SSL トンネリング**」チェック・ボックスをクリアします。

システムを保護するために基本的ないくつかの予防措置を講じることができます。

- 公開アクセス用のサーバーを、ローカル・ネットワークまたは社内ネットワークとは別のネットワーク上に置く。
- サーバーの内部プロセスにリモート・ユーザーがアクセスできるユーティリティを使用不可にする。サーバーを実行しているシステムでは、特に、**telnet**、**TN3270**、**rlogin**、および **finger** の各クライアントを使用可能とすることを検討する。
- パケット・フィルター操作およびファイアウォールを使用する。

パケット・フィルター操作によって、データがどこから来てどこへ行くかを定義することができます。一定の発信元と送信先の組み合わせを拒否するよう、システムを構成することができます。

ファイアウォールは、社内ネットワークを、インターネットなどの公衆ネットワークから分離します。ファイアウォールはコンピューター・グループまたは単一のコンピューターで、両方向のゲートウェイの役目を果たし、そこを通過するトラフィックを規制したり追跡したりできます。IBM Firewall は、ファイアウォール・ソフトウェアの一例です。

- CGI スクリプトを制御する。Web サーバー上で CGI スクリプトを使用すると、セキュリティ上のリスクが生じる可能性があります。その理由は、CGI スクリプトがユーザー ID やパスワードなどの機密データが入った環境変数を表示する可能性があるからです。CGI プログラムをサーバーで実行にする前に、その働きをよく理解し、サーバー上の CGI スクリプトにアクセスするユーザーを管理するようにしてください。

注: 「構成ウィザード」を使用してプロキシ・サーバーを構成し、SSL を使用可能にする場合は、ポート 443 を介して受け取るプロキシ要求に対するマッピング・ルールを作成する必要があります。詳細については、44 ページの『マッピング・ルールの定義』を参照してください。

例:

```
Proxy /* http://content server :443
```

または

```
Proxy /* https://content server :443
```



---

## 第 25 章 サーバー保護セットアップ

この章では、保護セットアップを使用して、サーバー上のデータおよびファイルを保護する方法について説明します。保護セットアップは、サーバーが受信する要求、特に、特定のディレクトリー、ファイル、または要求が対応するファイルのタイプに基づいて起動されます。保護セットアップでは、サブディレクティブが保護の対象となるディレクトリーまたはファイルの特性に基づいてアクセスを認可または拒否する方法を制御します。

---

### 「構成および管理」フォームを使用した保護の設定

保護セットアップおよびその適用方法を定義するには、「構成および管理」フォームで、「サーバー構成」→「文書保護」を選択します。このフォームを使用して、以下のステップを実行します。

1. この保護規則に順序の設定を設定します。

保護規則は、構成フォームの表にリストされた順序で適用されます。通常、規則は特定のものから全般的なものの順にリストされます。

ドロップダウン・メニューおよびボタンを使用して、保護規則の配置を指定します。

2. 要求テンプレートを定義します。

保護機能は、クライアントがプロキシ・サーバーに送信する要求の内容と比較される要求テンプレートに基づいて活動化されます。

要求 は、サーバーのホスト名に続く完全 URL の一部です。例えば、サーバーの名前が `fine.feathers.com` で、ブラウザー・ユーザーが URL `http://fine.feathers.com/waterfowl/schedule.html` と入力すると、サーバーが受信する要求は `/waterfowl/schedule.html` となります。要求テンプレートは、保護の対象となるディレクトリーまたはファイル名、あるいはその両方を指定します。例えば、今説明した要求テンプレート (`/waterfowl/schedule.html`) に基づいて保護機能を活動化する要求には `/waterfowl/*` および `/*schedule.html` が含まれます。

「URL 要求テンプレート」フィールドに要求テンプレートを入力します。

3. 保護セットアップを定義します。

保護セットアップは、要求テンプレートと一致する要求に対して行うことを Caching Proxy に指示します。名前付きの保護セットアップを使用することも、「文書保護」フォームで新規セットアップを定義することもできます。

名前付きセットアップを使用するには、「名前付き保護」ラジオ・ボタンをクリックして、提供されているフィールドに使用する名前付きセットアップの名前を入力します。新規セットアップを定義するには、「インライン」ラジオ・ボタンをクリックし、与えられた指示に従います (ステップ 6 を参照)。

4. 要求元のアドレスを選択します。(オプション)

要求に対してさまざまなサーバー・アドレスからさまざまな規則を適用することができます。例えば、会社割り当てられた IP アドレスから要求を受信する場合に、ログ・ファイルに対する要求に異なる保護セットアップを適用することが必要となる場合があります。

**注:** 要求元アドレスを画面表示するには、DNS ルックアップを使用可能にする必要があります。詳しくは、225 ページの『DNS-Lookup - クライアントのホスト名を検索するかどうかを指定する』を参照してください。

要求元のアドレスを規則に組み込む場合、そのアドレスを「**サーバー IP アドレスまたはホスト名**」フィールドに入力します。

5. 「**実行依頼**」をクリックします。

名前付き保護セットアップを使用した場合、これ以上の入力はありません。インライン保護セットアップを選択した場合、あるいは存在しない名前付きセットアップを指定した場合には、システムは追加のフォームをオープンします。

6. 保護の詳細を設定します。

既存の名前付き保護セットアップを指定しなかった場合には、追加のフォームがオープンされ、要求テンプレートと一致する文書またはディレクトリーに対してどのユーザーをアクセスできるようにするか、またどのアクションをそのユーザーに許可するかを指定できます。

- **パスワード認証設定** — ユーザーの認証のために、ユーザーにパスワード・ファイル、グループ・ファイル、または両方を指定します。また、要求元の名前とパスワードのプロンプトを出す場合にサーバーを識別するのに使用される名前を指定します。

**注:** ブラウザーのなかには、ユーザー ID とパスワードをキャッシュに入れて、それらをサーバー ID に関連付けているものもあります。常に同じサーバー ID と同じパスワード・ファイルを使用すれば、ユーザーにとって便利です。

- **許可** — 保護されたファイルに対する読み取り、書き込み、または削除をどのユーザーまたはグループに許可するかを指定します。

7. 「**実行依頼**」をクリックします。

8. サーバーを再始動します。

---

## 構成ファイル・ディレクティブを使用した保護の設定

Caching Proxy の構成ファイルを直接編集して保護を設定するためには、まず次の問題を理解しておく必要があります。

- Protect、defProt、および Protection の各ディレクティブの間の相違
  - Protect ディレクティブは、要求テンプレートを保護セットアップへリンクすることによって、保護機能を設定します。詳しくは、273 ページの『Protect - テンプレートと一致する要求の保護セットアップを活動化する』を参照してください。

- defProt ディレクティブは、特定の要求テンプレートにデフォルトの保護セットアップを設定します。詳しくは、217 ページの『DefProt - テンプレートと一致する要求にデフォルトの保護セットアップを指定する』を参照してください。
- Protection ディレクティブは、名前付き保護セットアップを定義するために使用します。詳しくは、278 ページの『Protection - 名前付き保護セットアップを構成ファイル内に定義する』を参照してください。
- 保護と要求経路指定がどのように対応しているか

Map、Exec、Pass、Proxy などの要求経路指定ディレクティブは、サーバーがどの要求を受け入れ、実際のファイル位置へどのように要求を経路指定するかを制御するために使用されます。要求経路指定ディレクティブは、保護ディレクティブと同じタイプの要求テンプレートを使用します。要求ごとに最初に突き合わせするテンプレートに関連した指示が実行されるので、保護が正しく機能するようにするためには、構成ファイル内で経路指定ディレクティブの前に保護ディレクティブをリストする必要があります。詳しくは、273 ページの『Protect - テンプレートと一致する要求の保護セットアップを活動化する』を参照してください。

- インライン保護セットアップと名前付き保護セットアップの相違

Protect ディレクティブを使用してインライン保護セットアップを指定することも、既存の名前付きセットアップを参照することもできます。2 つのタイプのステートメントの構文は多少異なります。詳しくは、273 ページの『Protect - テンプレートと一致する要求の保護セットアップを活動化する』を参照してください。

- 保護セットアップを記述する方法

保護セットアップは、保護サブディレクティブを使用する一連のステートメントです。保護セットアップの作成に関する構文および解説情報は、185 ページの『付録 B. 構成ファイル・ディレクティブ』に記載されています。次の該当するセクションを参照してください。

- 273 ページの『Protect - テンプレートと一致する要求の保護セットアップを活動化する』
- 278 ページの『Protection - 名前付き保護セットアップを構成ファイル内に定義する』
- 279 ページの『Protection subdirectives - 一連のリソースの保護方法を指定する』

---

## デフォルトの保護設定

デフォルトのプロキシー構成ファイルには、/admin-bin/ ディレクトリー内のファイルにアクセスするために管理者 ID とパスワードを必要とする保護セットアップが含まれています。この設定値は、「構成および管理」フォームへのアクセスを制限します。



---

## 第 26 章 Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) は、情報を自動的に暗号化してからそれをインターネット上で送信し、使用前に相手側でそれを暗号化解除するシステムです。これにより、インターネット上の伝送中に、クレジット・カード番号などの機密情報が保護されます。

Caching Proxy は SSL を使用して代理サーバーの安全を保護し、以下のセクションで説明するようにセキュア・リモート管理を可能にします。また、SSL を使用して、バックエンド・サーバー (例えば、コンテンツ・サーバーやアプリケーション・サーバー) への接続を保護したり、プロキシ・サーバーとそのクライアントの間の通信を保護したりすることもできます。

フォワード・プロキシの場合、Caching Proxy は SSL トンネリングをサポートします。これにより、SSL がバイパスされ、すでに暗号化されたデータが変更なしに送り出されます。

---

### SSL ハンドシェーク

SSL 保護は、セキュア・コネクション要求が 1 つのマシンから別のマシンに送信されるとき、例えば、ブラウザーが要求を代理プロキシ・サーバーに送信するときに、開始されます。要求構文として `http://` の代わりに `https://` を使用すると、サーバーがセキュア接続要求を `listen` するポート 443 (通常の要求用のポート 80 の代わり) 上で要求を送信するようにブラウザーに指示されます。ブラウザーとサーバーの間でセキュア・セッションを確立するには、2 つのマシンが SSL ハンドシェークと呼ばれる交換を実行してその暗号仕様に関して同意し、情報の暗号化および暗号化解除に使用される鍵を選択します。鍵は自動的に生成され、セッションが満了するとその有効期限が切れます。一般的なシナリオ (SSL バージョン 3 を想定) は、次のとおりです。

#### 1. Client hello

クライアントは、クライアントの暗号化能力を説明する Client Hello メッセージを送信して、Caching Proxy との SSL セッションを開始する。

#### 2. Server hello

サーバーはクライアントに証明書を送信し、データ暗号化に使用する暗号方式を選択する。

#### 3. Client finish

クライアントは、暗号化されたデータの対称暗号鍵の作成に使用される鍵の情報を送信する。この鍵の材料は、プリマスターリング・シークレットと呼ばれ、サーバーの公開鍵 (サーバーの証明書から取得します。130 ページの『鍵および認証の管理』を参照) で暗号化されます。サーバーもクライアントも、このプリマスターリング・シークレットから、読み取りおよび書き込みの対称暗号鍵を得ることができます。

#### 4. Server finish

サーバーは、ハンドシェーク・プロトコル全体の最終確認とメッセージ確認コード (MAC) を送信する。

#### 5. Client validation

クライアントは、サーバーの最終メッセージを検証するためのメッセージを送信する。

#### 6. Secure data flow

クライアントがサーバーの最終メッセージを検証すると、暗号化されたデータのフローが始まる。

Caching Proxy をセキュア接続のエンドポイントとして使用することによって、コンテンツ・サーバーまたはアプリケーション・サーバーの負荷を軽減することができます。Caching Proxy は、セキュア接続を維持する場合に、すべて CPU 集中操作となる暗号化および暗号化解除と、鍵の作成を行います。また、Caching Proxy によって SSL セッションのタイムアウトを構成して、それぞれの鍵の使用を最大限にすることができます。

### SSL の制限

WebSphere Application Server の Caching Proxy 内の SSL には、以下のような制限が適用されます。

- Caching Proxy そのものを認証局として使用することはできません (130 ページの『鍵および認証の管理』を参照してください)。
- 一部のブラウザーは、Caching Proxy で使用されているすべての暗号化テクノロジーをサポートしているとは限らない場合があります。

## SSL パフォーマンスの調整

HTTPS トラフィックの量が増すと、Caching Proxy サーバーが原因で、CPU 使用が高くなる場合があります。環境変数 (GSK\_V3\_SIDCACHE\_SIZE) およびプロキシー・ディレクティブ (SSLV3Timeout) に対する変更を調整すると、プロキシー・サーバーが負荷を処理しやすくなり、また CPU 使用の削減にも役立ちます。

SSL セッション ID は、再使用可能な SSL セッションを識別する ID です。ブラウザーとサーバーの両方で使用される、暗号化または暗号化解除の鍵を含みます。また、SSL セッション ID は、新規接続での 不要な SSL ハンドシェーク (サーバーの CPU 時間を大きく消費します) を回避するためにも使用されます。Caching Proxy サーバー用の GSKit ライブラリーは、SSL セッション ID をサポートし、SSL セッション ID キャッシュを含んでいます。デフォルトでは、SSL セッション ID キャッシュに 512 個の項目が含まれています。項目の制限に達すると、一番古いセッション項目が除去され、新規の項目がキャッシュに追加されます。

SSL セッション ID キャッシュのデフォルトのサイズを変更するには、GSK\_V3\_SIDCACHE\_SIZE 環境変数を使用します。この変数の有効な値は、1 から 4096 です。このサイズを大きくするほど、キャッシュ SSL セッションを探し出すために必要な検索時間が増加します。ただし、検索時間が増加することは、SSL 接続を確立するために必要となるオーバーヘッドと比較すると、それほど重要なことではありません。キャッシュ・サイズを大きくすると、プロキシー・サーバーは、



より多くの並行 SSL セッションを処理できるようになります。また、プロキシ・サーバーに高い HTTPS 負荷がかかっているときの CPU 使用も削減されます。

Caching Proxy には、調整可能な SSLV3Timeout ディレクティブもあります。(306 ページの『SSLV3Timeout - SSLV3 セッションが有効期限切れになるまでの待ち時間を指定する』を参照してください。) このディレクティブのデフォルト値は、1000 秒です。このディレクティブでは、セッション・キャッシュでの SSL セッションの存続時間を定義します。着信 SSL 接続で既存の SSL セッションが使用されず、しかもセッションの存続時間がこの値を超えると、そのセッションはセッション・キャッシュから除去されます。SSLV3Timeout 値は、保護されたクライアント・セッションの標準的な長さに設定することをお勧めします。タイムアウトをあまり短く設定すると、プロキシのパフォーマンスが低下することがあります。これは、単一の保護セッションを完了するために、複数の SSL ハンドシェイク・セッションが必要となるからです。ただし、長すぎる値を設定すると、保護セッションのセキュリティが損なわれる場合があります。

## SSL トンネリング

これは、フォワード・プロキシ構成にのみ適用されます。

Caching Proxy は、フォワード・プロキシ・サーバーとして構成されると、SSL トンネリングを使用して、クライアントとコンテンツ・サーバーの間のセキュア接続をサポートします。SSL トンネリングでは、暗号化されたデータが変換されずにプロキシ・サーバーをパススルーします。プロキシ・サーバーはデータの暗号化を解除しないため、プロキシ・サーバーが要求をまたは文書ヘッダーを読み取る必要のある機能は SSL トンネリングではサポートされていません。またトンネルに入れられる要求はキャッシュに入れられません。

図 2 では、SSL トンネリングを使用することにより接続を確立する方法を示しています。

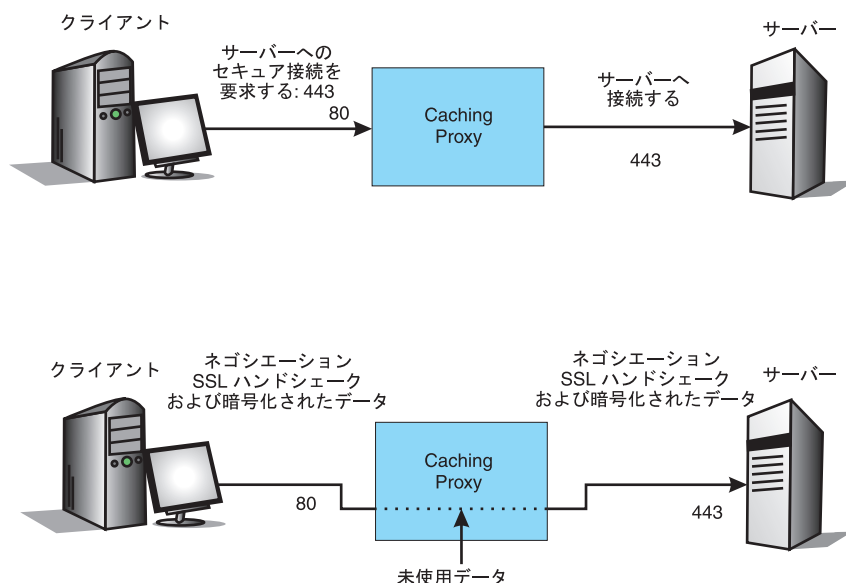


図 2. SSL トンネリング

SSL トンネリング・プロセスは、次のとおりです。

1. クライアントは、トンネリング要求を行います: `CONNECT server-host-name:port HTTP/1.1` (または `HTTP/1.0`)。ポート番号はオプションで、通常は 443 です。クライアントのブラウザーにフォワード・プロキシが構成されている場合、まず最初にブラウザーは、すべての `HTTPS` 要求において、プロキシ・サーバーへ自動的に `CONNECT` 要求を送信します。
2. プロキシはそのポート 80 で接続を受け入れ、要求を受信し、クライアントによって要求されたポートで宛先サーバーに接続します。
3. プロキシは、接続が確立されたことを伝えてクライアントに応答します。
4. プロキシは、`SSL` ハンドシェイク・メッセージを両方向で中継します。両方向とは、クライアントから宛先サーバーへの方向と、宛先サーバーからクライアントへの方向のことです。
5. セキュア・ハンドシェイクが完了すると、プロキシは、クライアントまたは宛先サーバーで暗号化解除するために、暗号化されたデータを送受信します。
6. クライアントまたは宛先サーバーがいずれかのポートでクローズを要求すると、プロキシ・サーバーは両方の接続 (ポート 443 と 80) をクローズして、その通常の活動を再開します。

## SSL トンネリングの構成

フォワード・プロキシ設定では、`SSL` トンネリングのみが使用可能です。`SSL` トンネリングを使用可能にするには、「構成および管理」フォームで、「**プロキシ構成**」->「**プロキシ設定**」を選択します。次に、「**SSL トンネリング**」チェック・ボックスを選択します。

`SSL` トンネリング接続では、`CONNECT` メソッド (デフォルトでは使用不可) も使用可能にする必要があります。これを使用可能にするには、「構成および管理」フォームで、「**サーバー構成**」->「**要求処理**」を選択し、「**HTTP メソッド**」フォームを使用します。

拡張 `SSL` トンネリング・セキュリティのために、`Enable CONNECT` ディレクティブには 3 つのオプション (`OutgoingPorts`、`OutgoingIPs`、`IncomingIPs`) が用意されています。少なくとも `OutgoingPorts` には、値を指定する必要があります。このオプションの値を指定しないと、`CONNECT` メソッドは有効になりません。

- `OutgoingPorts` (リモート・サーバーのポートによる `SSL` トンネリングのアクセスを制限します)。形式は次のとおりです。

```
Enable CONNECT OutgoingPorts [all | [port1|port1-port2|port1-*],...]
```

`SSL` トンネリングのために、クライアントがリモート・サーバーのポート 443 にのみ接続できるようにするには、次のディレクティブを設定します。(通常、ポート 443 は、リモート・サーバー上の `HTTPS` 要求に使われます。)

```
Enable CONNECT OutgoingPorts 443
SSLTunneling on
```

`SSL` トンネリングのために、クライアントがリモート・サーバーのすべてのポートに接続できるようにするには、次のディレクティブを設定します。

```
Enable CONNECT OutgoingPorts all
SSLTunneling on
```



SSL トンネリングのために、クライアントがリモート・サーバーのポート 80、8080-8088、および 9000 以上のポートに接続できるようにするには、次のディレクティブを設定します。

```
Enable CONNECT OutgoingPorts 80,8080-8088,9000-*
SSLTunneling on
```

ポートおよびポート範囲は、リスト内にスペースを入れずに、コンマで区切って指定します。

重要: フォワード・プロキシー構成の場合、通常の SSL トンネリングを有効にするためには、`OutgoingPorts` オプションで、少なくとも 443 または `all` を指定してください。

- `OutgoingIPs` (リモート・サーバーの IP アドレスによる SSL トンネリングのアクセスを制限します)。形式は次のとおりです。

```
Enable CONNECT OutgoingIPs [![IP_pattern,...]
```

たとえば、IP/ホスト名 `*.ibm.com` に一致し、`192.168.*.*` には一致しないリモート・サーバーのすべてのポートに、クライアントが接続できるようにするには、次のディレクティブを設定します。

```
Enable CONNECT OutgoingPorts all OutgoingIPs *.ibm.com,!192.168.*.*
SSLTunneling on
```

注: `IP_patterns` は、リスト内にスペースを入れずに、コンマで区切って指定します。

- `IncomingIPs` (クライアントの IP アドレスによる SSL トンネリングのアクセスを制限します)。形式は次のとおりです。

```
Enable CONNECT IncomingIPs [![IP_Pattern,...]
```

たとえば、SSL トンネリングのために、IP アドレスが `192.168.*.*` であるクライアントが、リモート・サーバーのすべてのポートへ接続を行えるようにするには、次のディレクティブを設定します。

```
Enable CONNECT OutgoingPorts all IncomingIPs 192.168.*.*
SSLTunneling on
```

注:

1. `192.168.*.*` が内部 LAN IP マスクであると想定した場合、上記のオプションでは、内部ユーザーのみが `CONNECT` メソッドおよび SSL トンネリング機能を使用できるようにしています。
2. `IP_patterns` は、リスト内にスペースを入れずに、コンマで区切って指定します。

プロキシー構成ファイルを編集して、SSL トンネリングおよび `CONNECT` ディレクティブを使用可能にするには、次のディレクティブの 185 ページの『付録 B. 構成ファイル・ディレクティブ』において、それぞれの解説セクションを参照してください。

- 226 ページの『`Enable` - HTTP メソッドを使用可能にする』
- 305 ページの『`SSLTunneling` - SSL トンネリングを使用可能にする』

---

## セキュア・リモート管理の構成

Caching Proxy のリモート管理は、Secure Sockets Layer (SSL) が提供するセキュリティー機能とパスワード認証を使用することにより行うことができます。この方法をとると、許可されない人がプロキシ・サーバーにアクセスする可能性が大幅に減少します。

サーバーのリモート管理を実行しているときに SSL を適用するには、`http://` 要求の代わりに `https://` 要求を使用して、「構成および管理」フォームをオープンします。例えば、次のとおりです。

`https://your.server.name/yourFrontPage.html`

---

## 鍵および認証の管理

前述のように SSL を構成する前に、鍵データベースをセットアップし、証明書を取得または作成する必要があります。証明書は、サーバー ID を認証するために使用されます。IBM 鍵管理ユーティリティー (iKeyman と呼ばれる場合もある) を使用して、証明書ファイルをセットアップしてください。このユーティリティーは Application Server に付属している GSKit ソフトウェアの一部です。GSKit には、証明書ファイルを開くための、Java ベースのグラフィカル・インターフェースも含まれています。

以下は、SSL キーおよび証明書をセットアップするための基本的な手順です。

1. GSKit がインストールされていることを確認する。ほとんどのプラットフォームでは、これは Caching Proxy コンポーネントによって自動的にインストールされます。パッケージの名前は、`gsk7ikm` (i386 用 Linux システムの場合、`gsk7ikm_gcc295`) になります。GSKit は、通常、`ibm/gsk7/` ディレクトリーにインストールされます (AIX システムの場合、`ibm/gskit/`)。Windows プラットフォームでは、「スタート」メニューからアクセスすることもできます。

注: Windows の場合、InstallShield の使用時に GSKit がインストールされない場合は、インストール・メディア・ディレクトリーへのパスがブランク・スペースを含んでいないことを確認してください。

2. 鍵管理を使用して、セキュア・ネットワーク・コミュニケーション用の鍵を作成し、認証局が発行する証明書を受信する。認証局から証明書を受け取るのを待っている間に、自己署名の証明書を作成するよう決定することができます。
3. 鍵データベースを作成し、鍵データベース・パスワードを指定する。

注: Caching Proxy がアンインストールされると、必ず `key` および `keystash` ファイルがアンインストールされます。認証局から新規証明書を要求しなくてもよいようにするには、これら 2 つのファイルのバックアップ・コピーを別のディレクトリーに保管してから、プロキシ・ソフトウェアをアンインストールしてください。

Linux 用以外のすべてのオペレーティング・システムで、証明書がの有効期限が切れた場合、Caching Proxy は適切に開始せず、鍵データベースの有効期限が切れたことを示すエラー・メッセージが表示されます。Linux の場合は、プロキシが現れて開始しますが、プロセスは即時に消失し、何のエラー・メッセージも生成しません。

Red Hat Enterprise Linux 3.0 システムでこの問題を回避するには、GCC パッケージが以下に示しているレベルか、またはそれ以上のレベルであることを確認してください。

- libstdc++-3.2.3-52
- libgcc-3.2.3-52

---

## 認証局

公開鍵は、サーバーのトラステッド・ルート認証局 (CA) と指定された CA による、デジタル署名済み証明書に関連したものでなければなりません。署名済み証明書は、認証局 (CA) プロバイダーに証明書要求を依頼することによって、購入することができます。Caching Proxy は、次の外部 CA をサポートしています。

- VeriSign
- Thawte

デフォルトでは、トラステッド CA として、以下のものが指定されています。

- Verisign Class 1 Individual Subscriber CA - Persona Not Validated
- Verisign Class 2 Individual Subscriber CA - Persona Not Validated
- Verisign Class 3 Individual Subscriber CA - Persona Not Validated
- VeriSign Class 3 International Server CA
- VeriSign Class 2 OnSite Individual CA
- VeriSign Class 1 Public Primary CA
- VeriSign Class 2 Public Primary CA
- VeriSign Class 3 Public Primary CA
- VeriSign Class 1 Public Primary CA - G2
- VeriSign Class 2 Public Primary CA - G2
- RSA Secure Server CA (VeriSign から)
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Premium Server CA
- Thawte Server CA

---

## IBM 鍵管理ユーティリティの使用

この項では、IBM 鍵管理ユーティリティ (iKeyman) の使用に関するクイック・リファレンスを提供します。鍵管理を使用して、SSL 鍵データベース・ファイル、公開鍵と秘密鍵のペア、および証明書要求を作成します。CA の署名済み証明書を受け取ったら、鍵管理を使用して、その証明書を、オリジナルの証明書要求を作成した鍵データベースに入れてください。

IBM 鍵管理および GSKit の詳細な資料が、GSKit ソフトウェアと同梱されています。

鍵管理を実行するためのシステムのセットアップ

IKeyman GUI を開始する前に、以下を実行してください。

1. IBM 32 ビット Java 2 Technology、バージョン 1.4.2、またはそれに相当するものをインストールします。
2. JAVA\_HOME を Java ディレクトリー・ロケーションに設定します。例えば、次のとおりです。
  - Windows の場合: `set JAVA_HOME=C:\Program Files\IBM\Java142`
  - Linux および UNIX の場合: `export JAVA_HOME=/usr/opt/IBMJava2-142`
3. `ibmjssse.jar` および `gskikm.jar` (ある場合)、および `ibmjcaprovider.jar` ファイルを `JAVA_HOME/jre/lib/ext` ディレクトリーから除去します。

注:

- a. Solaris の場合は、`JAVA_HOME/jre/lib/ext` ディレクトリーではなく `JAVA_HOME/lib/ext/` ディレクトリーで行います。
  - b. 別の製品 (たとえば、WebSphere Application Server) が依存している JDK の JAR ファイルは、移動や削除を行わないでください。ファイルの移動や削除を行うと、依存している製品が切断されたり、適正に動作しなくなることがあります。JDK が使用中であるかどうか不確実な場合は、IBM 鍵管理ユーティリティーのために別個の JDK をインストールしてください。
4. 次の JAR ファイルはすべて、現在 `GSKit_Installation_path/classes/jre/lib/ext/` にあります。
    - 指定された JAR ファイルを `JAVA_HOME/jre/lib/` にコピーします。  
`ibmjcefw.jar`  
`ibmpkcs11.jar`
    - 指定された JAR ファイルを `JAVA_HOME/jre/lib/ext` にコピーします。  
`ibmjceprovider.jar`  
`ibmpkcs.jar`
    - 指定された JAR ファイルを `JAVA_HOME/jre/lib/security` にコピーします。  
`local_policy.jar`  
`US_export_policy.jar`
  5. IBM JCE、IBM CMS、IBMJCEFIPS などのサービス・プロバイダーを登録します。

`JAVA_HOME/jre/lib/security/java.security` ファイルを更新し、Sun プロバイダーの後に、IBM CMS および IBM JCE プロバイダーの両方に追加します。例えば、次のとおりです。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.spi.IBMCMSProvider
security.provider.3=com.ibm.crypto.provider.IBMJCE
```

`java.security` ファイルのサンプルは、

`GSKit_Installation_path/classes/gsk_java.security` にあります。

- FIPS の操作を使用可能にするには、`JAVA_HOME/jre/lib/security/java.security` ファイルを更新し、Sun プロバイダーの後に `IBMJCEFIPS` も追加します。`IBMJCEFIPS` プロバイダーが、`IBMJCE` より高い優先順位で登録されたことを確認します。例えば、次のとおりです。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.spi.IBMCMSProvider
security.provider.3=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.4=com.ibm.crypto.provider.IBMJCE
```

6. (オプション) JSSE ユーザーが JSSE を使用して暗号ハードウェアにアクセスする場合は、ibmpkcs11.jar を JAVA\_HOME/jre/lib ディレクトリーにインストールし、GSKit\_Installation\_path/classes/native/native-support.zip の指示に従って、暗号ハードウェア共用ライブラリーをセットアップします。

注: ibmpkcs11.jar は、2002 年 8 月 5 日以降にリリースされた JSSE パッケージから入手することもできます。IBMPKCS11 サービス・プロバイダーを登録するための JAVA\_HOME/jre/lib/security/java.security ファイルの更新のサンプルは、以下のとおりです。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.crypto.pkcs11.provider.IBMPKCS11
```

## 鍵管理の開始

鍵管理グラフィカル・ユーザー・インターフェースは、次の方法で開始します。

- Linux および UNIX プラットフォームでは、コマンド・プロンプトから gsk7ikm と入力します。
- Windows プラットフォームでは、「スタート」->「プログラム」->「IBM WebSphere」->「Edge Components」->「Caching Proxy」->「鍵管理ユーティリティーの開始」をクリックします。

このセッション時に新規の鍵データベース・ファイルを作成する場合、作成されたファイルは、鍵管理を開始したディレクトリーに保管されます。

## 新規の鍵データベース、パスワード、および stash ファイルの作成

鍵データベースは、1 つまたは複数の鍵のペアと証明書を保管するために、サーバーが使用するファイルです。すべての鍵のペアと証明書に 1 つの鍵データベースを使用することも、複数のデータベースを作成することもできます。鍵管理ユーティリティーを使用して、新規の鍵データベースを作成し、そのパスワードと stash ファイルを指定します。

鍵データベースおよび stash ファイルを作成する手順は、次のとおりです。

1. 鍵管理ユーティリティーを開始します。
2. メインメニューから、「鍵データベース・ファイル」->「新規」を選択します。
3. 「新規」ダイアログ・ボックスで、ファイル・タイプ「CMS 鍵データベース」が選択されていることを確認します。鍵データベース名およびファイル場所を入力し、デフォルトの key.kdb を受け入れます。「了解」をクリックします。
4. 「パスワード・プロンプト」ダイアログ・ボックスに、このデータベースに対するパスワードを入力し、確認します。「了解」をクリックします。
5. パスワード・ファイルを隠すためのチェック・ボックスを選択します。プロンプトが出たところで、パスワードを入力し、検証のため確認パスワードを入力します。次のメッセージが表示されます。

DB-Type: CMS key database file keyfile\_database\_name

注: パスワード・ファイルを隠しておかないと、サーバーは始動しますが、ポート 443 で listen しません。

新規の鍵データベースを作成するときに指定するパスワードは、秘密鍵を保護します。文書に署名し、公開鍵で暗号化されたメッセージを暗号化解除することができるのは、秘密鍵だけです。

パスワードを指定する際には、以下のガイドラインに従ってください。

- パスワードには、米国英語の文字セットを使用しなければならない。
- パスワードは最低 6 文字で、そのうち少なくとも 2 文字は、連続しない数字でなければならない。パスワードには、容易に手に入る情報 (ユーザーの名前または肉親の名前、イニシャル、あるいは誕生日など) を決して使用しないでください。
- パスワードを隠しておく。

鍵データベースのパスワードをしばしば変更するのは、よい方法です。ただし、パスワードに有効期限を指定した場合は、パスワードの変更時期を記録してください。変更する前にパスワードが有効期限切れになると、エラー・ログにメッセージが書き込まれます。この場合、サーバーは始動しますが、安全なネットワーク接続ができなくなります。

鍵データベース・パスワードを変更するには、次のステップに従ってください。

1. メインメニューから、「**鍵データベース・ファイル**」->「**オープン**」をクリックします。
2. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力するか、デフォルトの **key.kdb** を受け入れます。次に「**了解**」をクリックします。
3. 「**パスワード・プロンプト**」ダイアログ・ボックスに、設定したパスワードを入力し、「**了解**」をクリックします。
4. メインメニューから、「**鍵データベース・ファイル**」->「**パスワード変更**」をクリックします。
5. 「**パスワード変更**」ダイアログ・ボックスに新規パスワードを入力し、確認します。「**了解**」をクリックします。

プロキシ・サーバーと LDAP サーバーの間の SSL 接続の場合は、鍵データベース・パスワードを **pac\_keyring.pwd** ファイルに入れます。(pac\_keyring.pwd ファイルは IKeyMan の生成する stash ファイルではないことに注目してください。)

### 新規の鍵のペアと証明書要求を作成する

鍵データベースは、鍵のペアと証明書要求を保管します。公開鍵と秘密鍵のペアおよび証明書要求を作成するには、次のようにします。

1. 鍵データベースを作成していない場合は、133 ページの『新規の鍵データベース、パスワード、および stash ファイルの作成』の説明を参照します。
2. 鍵管理ユーティリティで、メインメニューから、「**鍵データベース**」->「**ファイル**」->「**オープン**」をクリックします。
3. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力し (デフォルト設定を使用する場合は、**key.kdb** をクリックする)、「**了解**」をクリックします。



4. 「パスワード・プロンプト」ダイアログ・ボックスに、パスワードを入力し、「了解」をクリックします。
5. メインメニューから、「作成」->「新規証明書要求」をクリックします。
6. 「新規鍵および証明書要求」ダイアログ・ボックスで、次の項目を指定します。
  - **鍵ラベル:** データベースで鍵と証明書を識別するのに使用する、名前 (ラベル) を入力します。(例えば、my self-signed certificate または www.companyA.com)
  - **鍵サイズ:** 鍵のサイズ、例えば 1024。(128 ビット暗号化を活用するためには、鍵サイズ 1024 を推奨します。)
  - **組織名:** 鍵に関連付ける組織の名前。(例えば、Company A)
  - **組織単位** (オプション)
  - **所在地** (オプション)
  - **州/郡** (オプション)
  - **郵便番号** (オプション)
  - **国:** 国別コード。少なくとも 2 文字 (US など) を指定しなければなりません。
  - **証明書要求ファイル名:** 要求ファイルの名前。オプションとして、デフォルト名を使用できます。
7. 「了解」をクリックします。次のような確認メッセージが表示されます。

A new certificate request has been successfully created  
in the file *keyfile\_database\_name*.
8. 「了解」をクリックします。「パーソナル証明書要求」ヘッダーの下に、入力したラベル名が表示されます。
9. 「情報」ダイアログ・ボックスで、「了解」をクリックします。ファイルを認証局に送信するようにとのメッセージが表示されます。
10. 自己署名の証明書を作成していない場合には (詳しくは、『自己署名の証明書を作成する』を参照)、証明書要求を CA に送信します。
  - 鍵管理を実行したままにします。
  - Web ブラウザーを起動して、証明書を獲得したい CA の URL を入力します。
  - 証明書を送信するための、CA の指示に従ってください。

証明書要求の完了には、2 週間から 3 週間かかります。CA が証明書要求を処理するのを待つ間に、ご自身が CA として動作し、iKeyman を使用して、自己署名のサーバー証明書を作成し、クライアントと Caching Proxy サーバーの間の SSL セッションを使用可能にすることができます。

### 自己署名の証明書を作成する

証明書が発行されるのを待つ間に、鍵管理ユーティリティーを使用して自己署名のサーバー証明書を作成し、クライアントとプロキシー・サーバー間の SSL セッションを使用可能にすることができます。また、この自己署名の証明書はテスト目的にも使用できます。

次の手順に従って、自己署名の証明書を作成します。



1. 鍵データベースを作成していない場合は、133 ページの『新規の鍵データベース、パスワード、および stash ファイルの作成』の説明を参照します。
2. 鍵管理ユーティリティで、メインメニューから、「**鍵データベース**」->「**ファイル**」->「**オープン**」をクリックします。
3. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「**了解**」をクリックします。
4. 「**パスワード・プロンプト**」ダイアログ・ボックスに、パスワードを入力し、「**了解**」をクリックします。
5. 「**鍵データベース**」コンテンツ・フレームで「**パーソナル証明書**」を選択し、「**新規の自己署名証明書の作成**」をクリックします。
6. 「**新規の自己署名証明書の作成**」ウィンドウで、次の項目を指定します。
  - **鍵ラベル:** データベースで鍵と証明書を識別するのに使用する、名前 (ラベル)。(例えば、my self-signed certificate)
  - **鍵サイズ:** 鍵のサイズ (例えば 512)
  - **共通名:** サーバーの完全ホスト名。(例えば、www.myserver.com)
  - **組織名:** 鍵に関連付ける組織の名前。(例えば、Company A)
  - **組織単位** (オプション)
  - **所在地** (オプション)
  - **州/郡** (オプション)
  - **郵便番号** (オプション)
  - **国:** 国別コード。少なくとも 2 文字 (US など) を指定しなければなりません。
  - **有効期間:** 証明書が有効である期間。
7. 「**了解**」をクリックします。
8. 鍵ファイルおよび stash ファイルを構成設定に追加して、サーバーに鍵データベースを登録します (133 ページの『新規の鍵データベース、パスワード、および stash ファイルの作成』を参照してください)。

### 鍵をエクスポートする

次の手順を使用して、鍵を別の鍵データベースにエクスポートします。

1. 鍵管理ユーティリティを開始します。
2. メインメニューから、「**鍵データベース・ファイル**」->「**オープン**」をクリックします。
3. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「**了解**」をクリックします。
4. 「**パスワード・プロンプト**」ダイアログ・ボックスに、パスワードを入力し、「**了解**」をクリックします。
5. 「**鍵データベース**」コンテンツ・フレームで、「**パーソナル証明書**」を選択し、ラベル上の「**エクスポート/インポート**」ボタンをクリックします。
6. 「**鍵のエクスポート/インポート**」ウィンドウでは、次のようにします。

- 「**鍵のエクスポート**」を選択します。
  - ターゲット・データベースのタイプ (例えば、**PKCS12**) を選択します。
  - ファイル名を入力するか、「**参照**」をクリックしてファイル名を選択します。
  - 正確な場所を入力します。
7. 「**了解**」をクリックします。
  8. 「**パスワード・プロンプト**」ダイアログ・ボックスで、正しいパスワードを入力し、確認のためもう一度そのパスワードを入力した後、「**了解**」をクリックして、選択した鍵を別の鍵データベースにエクスポートします。

### 鍵をインポートする

別の鍵データベースから鍵をインポートするには、次のようにします。

1. 鍵管理ユーティリティーを開始します。
2. メインメニューから、「**鍵データベース・ファイル**」->「**オープン**」を選択します。
3. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「**了解**」をクリックします。
4. 「**パスワード・プロンプト**」ダイアログ・ボックスに、正しいパスワードを入力し、「**了解**」をクリックします。
5. 「**鍵データベース**」コンテンツ・フレームで、「**パーソナル証明書**」を選択し、ラベル上の「**エクスポート/インポート**」ボタンをクリックします。
6. 「**鍵のエクスポート/インポート**」ウィンドウでは、次のようにします。
  - 「**鍵のインポート**」を選択します。
  - 鍵データベースのタイプ (例えば、**PKCS12**) を選択します。
  - ファイル名を入力するか、「**参照**」をクリックしてファイル名を選択します。
  - 正確な場所を選択します。
7. 「**了解**」をクリックします。
8. 「**パスワード・プロンプト**」ダイアログ・ボックスに、正しいパスワードを入力し、「**了解**」をクリックします。
9. 「**鍵ラベルから選択**」リストで正確なラベル名を選択し、「**了解**」をクリックします。

### 認証局のリスト表示

鍵データベース中のトラステッド認証局 (CA) のリストを表示するには、次のようにします。

1. 鍵管理ユーティリティーを開始します。
2. メインメニューから、「**鍵データベース・ファイル**」->「**オープン**」をクリックします。
3. 「**オープン**」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「**了解**」をクリックします。

4. 「パスワード・プロンプト」ダイアログ・ボックスに、正しいパスワードを入力し、「了解」をクリックします。
5. 「鍵データベース」コンテンツ・フレームで「署名者の証明書」を選択します。
6. 「署名者の証明書」、「パーソナル証明書」、または「証明書要求」をクリックして、「鍵情報」ウィンドウに CA のリストを表示させます。

## CA の証明書を受け取る

この手順は、デフォルトによってトラステッド CA として指定された認証局 (CA) から電子メールで送信されてきた証明書を受け取るために使用します (131 ページの『認証局』のリストを参照)。CA の署名済み証明書を発行する CA が、鍵データベースのトラステッド CA でない場合は、まず CA の証明書を保管し、その CA をトラステッド CA に指定しなければなりません。そうすれば、CA の署名済み証明書を、データベースに受け入れることができます。トラステッド CA 以外の CA から、CA の署名済み証明書を受け取ることはできません。(『CA の証明書を保管する』を参照)

CA の署名入り証明書を鍵データベースに受け入れるには、次のようにします。

1. 鍵管理ユーティリティを開始します。
2. メインメニューから、「鍵データベース・ファイル」->「オープン」を選択します。
3. 「オープン」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「了解」をクリックします。
4. 「パスワード・プロンプト」ダイアログ・ボックスに、パスワードを入力し、「了解」をクリックします。
5. 「DB タイプ」リストのファイル名が正しいことを確認します。
6. 「鍵データベース」ウィンドウで、「パーソナル証明書」を選択し、「受信」をクリックします。
7. 「ファイルから証明書を受信」ダイアログ・ボックスで、base 64 でエンコードされた有効なファイルの名前を「認証ファイル名」テキスト・フィールドに入力し、「了解」をクリックします。
8. 鍵管理ユーティリティをクローズするには、メインメニューから、「鍵データベース・ファイル」->「終了」をクリックします。

## CA の証明書を保管する

セキュア接続を確立するために、トラステッド CA によって署名された証明書のみが受け入れられます。トラステッド認証局のリストに CA を追加するには、そのトラステッド証明書を取得し、保管する必要があります。新規の CA から発行された証明書を保管するには、それをデータベースに受け入れる前に次の手順に従います。

1. 鍵管理ユーティリティを開始します。
2. メインメニューから、「鍵データベース・ファイル」->「オープン」をクリックします。

3. 「オープン」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「了解」をクリックします。
4. 「パスワード・プロンプト」ダイアログ・ボックスに、パスワードを入力し、「了解」をクリックします。
5. 「鍵データベース」コンテンツ・フレームで、「署名者の証明書」を選択し、「追加」をクリックします。
6. 「ファイルから CA の証明書を追加」ダイアログ・ボックスで、base 64 でエンコードされた ASCII データ証明書のファイル名を選択するか、「参照」オプションを使用し、「了解」をクリックします。
7. 「ラベル」ダイアログ・ボックスで、ラベル名を入力し、「了解」をクリックします。
8. チェック・ボックスを使用して、その証明書をトラステッドとして指定します(デフォルト)。

注: 証明書が作成された後、「表示/編集」ボタンを使用してチェック・ボックスを表示します。チェック・ボックスはパネルにリストされていますが、証明書を追加している間は表示されません。

#### 鍵データベースのデフォルト鍵を表示する

次の手順でデフォルトの鍵項目を表示します。

1. 鍵管理ユーティリティを開始します。
2. メインメニューから、「鍵データベース・ファイル」->「オープン」をクリックします。
3. 「オープン」ダイアログ・ボックスに鍵データベースの名前を入力します。(または、デフォルトの **key.kdb** を受け入れます。) 次に「了解」をクリックします。
4. 「パスワード・プロンプト」ダイアログ・ボックスに、パスワードを入力し、「了解」をクリックします。
5. 「鍵データベース」コンテンツ・フレームで「パーソナル証明書を」選択し、CA 証明書のラベル名を選択します。
6. 「鍵情報」ウィンドウで、「表示/編集」をクリックして、証明書のデフォルトの鍵情報を表示させます。

---

## サポートしている暗号仕様

SSL のバージョン 2 および 3 で使用されている、暗号化アルゴリズムとハッシュを次の表に示します。

鍵のペアの生成: RSA 512 - 1024 秘密鍵のサイズ

#### SSL バージョン 2

米国内バージョン	エクスポート・バージョン
RC4 US	RC4 エクスポート
RC2 US	RC2 エクスポート

DES 56 ビット	適用不可
Triple DES US	適用不可
RC4 エクスポート	適用不可
RC2 エクスポート	適用不可

### SSL バージョン 3

米国内バージョン	エクスポート・バージョン
Triple DES SHA US	DES SHA エクスポート
DES SHA エクスポート	RC2 MD5 エクスポート
RC2 MD5 エクスポート	RC4 MD5 エクスポート
RC4 SHA US	NULL SHA
RC4 MD5 US	NULL MD5
RC4 MD5 エクスポート	NULL NULL
RC4 SHA 56 ビット	適用不可
DES CBC SHA	適用不可
NULL SHA	適用不可
NULL MD5	適用不可
NULL NULL	適用不可

これらの SSL 仕様は、プロキシ構成ファイルを直接に編集しても構成できます。詳しくは、以下のディレクティブの 185 ページの『付録 B. 構成ファイル・ディレクティブ』の該当するセクションを参照してください。

- 311 ページの『V2CipherSpecs - SSL バージョン 2 についてサポートされる暗号仕様をリストする』
- 312 ページの『V3CipherSpecs - SSL バージョン 3 についてサポートされる暗号仕様をリストする』
- 234 ページの『FIPSEnable - SSLV3 および TLS 用の Federal Information Processing Standard (FIPS) 承認済み暗号を使用可能にする』

### Caching Proxy の場合の 128 ビット暗号化

Caching Proxy の 128 ビット暗号化バージョンのみが出荷されます。56 ビット・バージョンは使用できなくなりました。直前のバージョンを更新する場合には、現在インストール済みの 128 ビットまたは 56 ビット・バージョンに Caching Proxy を直接インストールすることができます。これまで 56 ビット (エクスポート) ブラウザーを使用していた場合には、プロキシで 128 ビット暗号化を活用できるように、128 ビット・ブラウザーにアップグレードする必要があります。

Caching Proxy の 56 ビット・バージョンから 128 ビット・バージョンへのアップグレード後に、証明書の暗号化に使用される鍵のサイズが 1024 に設定されている場合には、構成の変更は不要です。しかし、鍵のサイズが 512 に設定されている場合には、プロキシの 128 ビット暗号化を活用できるように、鍵のサイズが 1024 の新規証明書を作成する必要があります。IBM 鍵管理ユーティリティ (iKeyman) を使用して、新しい鍵を作成します。

1. 鍵管理ユーティリティを開始します。
  - Linux および UNIX プラットフォームでは、コマンド・プロンプトから `gsk7ikm` と入力します。
  - Windows システムでは、「スタート」->「プログラム」->「IBM WebSphere」->「Edge Components」->「鍵管理ユーティリティの開始」をクリックします。
2. メインメニューから、「鍵データベース・ファイル」->「オープン」をクリックします。
3. 「オープン」ダイアログ・ボックスに鍵データベースの名前を入力し (デフォルト設定を使用する場合は、`key.kdb` をクリックする)、「了解」をクリックします。
4. 「パスワード・プロンプト」ダイアログ・ボックスがオープンされた場合には、パスワードを入力して、「了解」をクリックします。
5. メインメニューから、「作成」->「新規証明書要求」をクリックします。
6. 「新規鍵および証明書要求」ウィンドウで、次の項目を指定します。
  - 鍵ラベル: データベースで鍵と証明書を識別するのに使用する、名前を入力します。
  - 鍵のサイズ: 「1024」を選択します。
  - 組織名: この鍵と関連した組織の名前を入力します。
  - 国: 国別コードを入力します。少なくとも 2 文字 (US など) を指定しなければなりません。
  - 証明書要求ファイル名: 要求ファイルの名前を入力します。あるいは、オプションとしてデフォルト名を使用します。
7. 「了解」をクリックします。

IBM 鍵管理ユーティリティの詳細な説明については、130 ページの『鍵および認証の管理』を参照してください。

製品のこのバージョンでは、SUSE Linux での暗号化はサポートされないことに注意してください。





---

## 第 27 章 暗号ハードウェアのサポートの使用可能化

これは、リバース・プロキシ構成にのみ適用されます。

次の手順に従って、SSL ハンドシェーク・ルーチンを暗号ハードウェア・カードにオフロードできるようにします。

1. メーカーの指示に従って、暗号ハードウェア・カードをインストールします。
2. Caching Proxy で SSL を使用できるようにします。詳細については、125 ページの『第 26 章 Secure Sockets Layer (SSL)』を参照してください。
3. 手作業で、ibmproxy.conf 構成ファイルの SSLCryptoCard ディレクティブを編集します。「構成および管理」フォームには、このディレクティブの項目は表示されません。詳細については、303 ページの『SSLCryptoCard - インストール済み暗号カードを指定する』で SSLCryptoCard ディレクティブの解説を参照してください。

AIX で、IBM 4960 PCI 暗号アクセラレーター・カードをサポートするためには、270 ページの『PKCS11DefaultCert、PKCS11DriverPath、PKCS11TokenPassword - IBM 4960 PCI 暗号アクセラレーター・カードをサポートする (AIX のみ)』を参照してください。



---

## 第 28 章 Tivoli Access Manager プラグインの使用

Caching Proxy プラグインは、Tivoli Access Manager (以前は Tivoli Policy Director) とともに提供され、Caching Proxy が Access Manager を認証および許可に使用できるようにします。このプラグインによって、Web アクセス制御に Access Manager を使用する企業は、プロキシ・サーバー 用に個別の許可スキーマをセットアップする重複した作業を必要とせずに、Edge テクノロジーを追加できます。

Tivoli Access Manager についての追加情報は、製品 Web サイト <http://www.ibm.com/software/tivoli/products/> をご覧ください。ソフトウェアおよびハードウェア要件と Access Manager プラグインのインストールについては、Tivoli Access Manager とともに提供される文書を参照してください。

注: Tivoli Access Manager プラグインは、Red Hat Linux ではサポートされていません。Linux プラットフォームに関する現在のサポート情報については、Tivoli にご連絡ください。

---

### 構成

Caching Proxy 用のセットアップ・スクリプトは、Access Manager プラグインとともに提供されます。

#### 構成スクリプトを使用する前にとるステップ

スクリプトを実行する前に、以下を実行してください。

- 必要なすべてのソフトウェアをインストールします。
- プロキシ・サーバーがポート 80 を使用するように設定されていることを確認します (これがデフォルト値です)。
- LDAP および Access Manager コンポーネントを構成し、Access Manager プラグインの構成中にそれらが稼働することを確認します。
- Access Manager 管理者 ID および LDAP 管理者名が使用可能であることを確認します。プロキシ・サーバーをセットアップするにはこれらの値が必要です。

#### 構成スクリプトの使用

セットアップ・スクリプトは **wslconfig.sh** という名前で、`/opt/pdweb-lite/bin/` ディレクトリーで提供されています。プロンプトに応じて、Access Manager 管理者 ID および LDAP 管理者名を入力してください。

構成スクリプトでは、以下のステップが自動的に実行されます。

- Caching Proxy ユーザー ID を `root` に設定し、グループ ID を `other` に設定します。
- `noLog` ディレクティブを `*` に設定します。これにより、Caching Proxy のアクセス・ログに項目は書き込まれません。
- 以下の情報を使用して `ServerInit` ディレクティブを作成します。

```
ServerInit /opt/pdweb-lite/lib/wesauth.so:WTESeal_Init  
/opt/pdweb-lite/etc/ibmwesas.conf
```

- 以下の情報を使用して PreExit ディレクティブを作成します。

```
PreExit /opt/pdweb-lite/lib/wesauth.so:WTESeal_PreExit
```

- 以下の情報を使用して Authorization ディレクティブを作成します。

```
Authorization * /opt/pdweb-lite/lib/wesauth.so:WTESeal_Authorize
```

- 以下の情報を使用して ServerTerm ディレクティブを作成します。

```
ServerTerm /opt/pdweb-lite/lib/wesauth.so:WTESeal_Term
```

以下のように、すべての要求を Access Manager 認証プロセスに転送する、Protect ステートメントおよび Protection セットアップを作成します。

```
Protection PROXY-PROT {  
    ServerId WebSEAL-Lite  
    Mask All@(*)  
    AuthType Basic  
}  
Protect * PROXY-PROT
```

---

## Caching Proxy および Access Manager プラグインの始動

プロキシ・サーバー および Access Manager プラグインを構成した後に、**ibmproxy start** ではなく **wslstartwte** コマンドを使用して プロキシ・サーバー を始動します。**wslstartwte** コマンドは、Access Manager プラグインが初期化のために必要とする環境変数を自動的にロードします。プロキシ・サーバー の始動時に **wslstartwte** を使用しない場合は、Access Manager プラグインに関するエラー・メッセージが表示されます。プラグインが使用されているときには、対応する停止コマンド **ibmproxy stop** がまだ有効です。

---

## 第 29 章 PAC-LDAP 許可モジュールの使用

---

### 概要

PAC-LDAP 許可モジュールによって、Caching Proxy は許可または認証ルーチンの実行時に Lightweight Directory Access Protocol (LDAP) サーバーにアクセスできます。このモジュールは 2 つのコンポーネント・セットから構成されています。Caching Proxy API と Policy Authentication Control (PAC) デーモンに LDAP 機能を追加する共用ライブラリーのペアです。ibmproxy.conf ファイルの中の ServerInit ディレクティブは、Caching Proxy の始動時に 1 つ以上の PAC デーモンを初期化するように共用ライブラリーに指示します。共用ライブラリーは、paccp.conf ファイルを読み取って、PAC デーモンの数値および特性を判別します。初期化中にこのデーモンは、pac.conf ファイルで構成ディレクティブを、また pacpolicy.conf でポリシー情報を読み取ります。次に、ibmproxy.conf ファイル内の Authentication ディレクティブが、認証が必要なときに共用ライブラリーを呼び出すようにプロキシ・サーバーに指示するか、あるいは Authorization ディレクティブが標準 HTTP 要求の処理中に Caching Proxy のワークフローを取り出します。

### 認証

認証のプロセスは、提供されたクリデンシャルのセット、すなわちユーザー名およびパスワードが有効かどうかを判断します。このプロセスには、ユーザーがレジストリー内に存在すること、および提供されたパスワードがレジストリーに保管されたパスワードと一致することの検証が含まれます。これらのアクションは、認証のステップの中で PAC-LDAP モジュールを使用して実行されます。

PAC-LDAP 許可モジュールは、認証に使用できるようになっている場合は、ユーザー ID、パスワード、およびグループを検索するデフォルトのリポジトリになります。HTTP 要求が Caching Proxy のワークフローに渡されると、それぞれの Protect ディレクティブが要求された URL をその要求テンプレートと比較します。一致が見つかり、Protect ディレクティブは、サーバー ID、使用する認証のタイプ、要求元クライアントに適用されるマスキング規則、およびパスワード・ファイルとグループ・ファイルの場所が入っている保護スキーマを呼び出します。パスワード・ファイルが定義されていない場合には、ユーザー ID とパスワードは PAC-LDAP 許可モジュールを介して検索されます。タイプ 0、1、2、および 3 のポリシーは認証スキーマを定義します。認証されるとその要求はサービスされ、認証されないと Caching Proxy はクライアントに 401 エラーを戻します。

### 許可

認証のプロセスは、ユーザーが保護リソースにアクセスするために必要な許可を持っているかどうかを判断します。PAC-LDAP モジュールが使用される場合は、HTTP 要求に対して pacpolicy.conf ファイル内の許可規則が適用されます。

PAC-LDAP 許可モジュールが許可について使用可能になっていると、pacpolicy.conf ファイル内の許可規則が HTTP 要求に適用されます。HTTP 要求が Caching Proxy のワークフローに渡されると、それぞれの Protect ディレクティブが要求された

URL をその要求テンプレートと比較します。一致が見つかったら、Protect ディレクティブは保護スキーマを呼び出します。この場合には、保護スキーマは PAC-LDAP 許可モジュールによって取り出された許可ルーチンです。Authorization ディレクティブは、要求された URL をその要求テンプレートと比較して、一致するものが見つかったら PAC-LDAP 許可モジュールが呼び出されます。タイプ 4 のポリシーは、pacpolicy.conf ファイル内で定義されて、各種の URL 要求に必要な認証を詳細に定義します。

## Lightweight Directory Access Protocol (LDAP)

LDAP は、最少のシステム・リソースを用いて対話的に X.500 ディレクトリーにアクセスします。IANA は LDAP に TCP ポート 389 と UDP ポート 389 を割り当てています。詳細については、LDAP を定義している RFC 1777 を参照してください。

サポートされる LDAP クライアントの例としては、IBM Tivoli LDAP クライアントおよび IBM SecureWay LDAP クライアントがあります。

## インストール

PAC-LDAP 許可モジュールのすべてのコンポーネントは、WebSphere Application Server、バージョン 6.1 の Caching Proxy システムのインストール時に自動的にインストールされます。Linux および UNIX システムでは、Caching Proxy ライブラリー (./lib/) ディレクトリー、PAC-LDAP 許可モジュール ライブラリー (./lib/plugins/pac/) ディレクトリー、バイナリー (./bin/) ディレクトリー、および構成 (./etc/) ディレクトリーが /opt/ibm/edge/cp/ ディレクトリー内に作成されます。次に、/usr/lib/、/usr/sbin/、および /etc ディレクトリーからそれらの製品特有のものへのシンボリック・リンクが作成されます。

ディレクトリー構造

Linux および UNIX ディレクトリー	Windows ディレクトリー	コンテンツ
/opt/ibm/edge/cp/	%Program Files%\IBM\edge\cp%	Caching Proxy 基本ディレクトリー (cp_root)
cp_root/sbin/	%Program Files%\IBM\edge\cp%\Bin%	Caching Proxy バイナリーおよびスクリプト
/usr/sbin/		cp_root/sbin/ へのシンボリック・リンク
cp_root/etc/	%Program Files%\IBM\edge\cp%\etc%	Caching Proxy 構成ファイル
/etc/		cp_root/etc/ へのシンボリック・リンク
cp_root/lib/	%Program Files%\IBM\edge\cp%\lib%\plugins%	Caching Proxy ライブラリー

Linux および UNIX ディレクトリー	Windows ディレクトリー	コンテンツ
<code>cp_root/lib/ plugins/pac/</code>	<code>¥Program Files¥IBM¥edge¥cp¥lib¥plugins¥pac¥</code>	PAC-LDAP 許可モジュール・ライブラリー
<code>/usr/lib/</code>		<code>cp_root/lib/</code> および <code>cp_root/lib/ plugins/pac/</code> へのシンボリック・リンク
<code>cp_root/server_root/pac/data/</code>	<code>¥Program Files¥IBM¥edge¥cp¥server_root¥pac¥data¥</code>	PAC-LDAP 許可モジュール・データ・ストレージ
<code>cp_root/server_root/ pac/creds/</code>	<code>¥Program Files¥IBM¥edge¥cp¥server_root¥pac¥creds¥</code>	PAC-LDAP 許可モジュール・クリデンシャル

#### LDAP プラグイン・ファイル

Linux および UNIX ファイル名	Windows ファイル名	説明
<code>libpacwte.so</code>	<code>pacwte.dll</code>	共用ライブラリー
<code>libpacman.so</code>	<code>pacman.dll</code>	共用ライブラリー
<code>pacd_restart.sh</code>	<code>pacd_restart.bat</code>	PAC デーモン再始動スクリプト
<code>paccp.conf, pac.conf, pacpolicy.conf</code>	<code>paccp.conf, pac.conf, pacpolicy.conf</code>	構成およびポリシー・ファイル

## セキュア PACD-LDAP サーバー接続のための追加要件および制限事項

### LDAP クライアント・パッケージでは GSKit が必要

PACD デーモンと LDAP サーバーとの Secure Sockets Layer (SSL) 接続を可能にするには、LDAP クライアント・パッケージで必要とされる GSKit パッケージをインストールしてください。GSKit 7 は Caching Proxy マシンでは必要であり、マシン上でデフォルトで提供されていますが、マシン上の LDAP クライアントが必要とするバージョンとは異なるバージョンの可能性があります。同一のマシン上で、様々なプロセスで異なるバージョンの GSKit を使用することは可能です。

GSKit 鍵ファイルを `$pacd_creds_dir/pac_keyring.kdb` に置き、パスワードを `$pacd_creds_dir/pac_keyring.pwd` に置きます。

注: LDAP サーバー上の GSKit 要件の情報については、Web サイト <http://www.ibm.com/software/tivoli/products/directory-server/> にある IBM Tivoli Directory Server (ITDS) 文書を参照してください。



## Linux システムには LD\_PRELOAD 環境変数の設定が必要

Linux システムでは、PACD デーモンと LDAP サーバーとの SSL 接続を可能にするために、以下に説明されているように LD\_PRELOAD 環境変数を構成する必要があります。変数に次の値を設定してください。

```
LD_PRELOAD=/usr/lib/libstdc++-libc6.1-1.so.2
```

このセクションで前述の GSKit 要件は、Linux システムにも適用されます。

## Linux システムで IBM Tivoli Directory Server (ITDS) 6.0 LDAP クライアントを使用する際、PACD プロセスの開始に失敗する

Red Hat Enterprise Linux 4.0 システムでは、認証に ITDS 6.0 LDAP プラグインを使用するよう Caching Proxy が構成されている場合、PACD プロセスが開始されません。その結果、次のエラー・メッセージが出されます。

```
"error while loading shared libraries:
/usr/lib/libldapiconv.so: R_PPC_REL24 relocation at 0x0fb58ad0
for symbol 'strpbrk' out of range"
```

現在、ITDS 6.0 で RHEL 4.0 システムがサポートされないという制限があります。

## AIX システムで、IBM Tivoli Directory Server (ITDS) LDAP クライアントを使用する際、PAC-LDAP モジュールをロードできない

ITDS LDAP クライアントの使用時、未解決のリンクがあると、AIX システムで PACD プロセスが開始されません。PACD プロセスを開始する際、次のようなエラーが発生する可能性があります。

```
exec(): 0509-036 Cannot load program /usr/sbin/pacd
because of the following errors:
    0509-022 Cannot load module /usr/lib/libpacman.a.
    0509-150 Dependent module libldap.a could not be loaded.
    0509-022 Cannot load module libldap.a.
```

ITDS バージョン 5 の LDAP クライアントで、この問題に対処するには、次のシンボリックを作成します。

```
ln -s /usr/lib/libibmldap.a /usr/lib/libldap.a
```

ITDS バージョン 6 の LDAP クライアントで、この問題に対処するには、次のシンボリックを作成します。

```
ln -s /opt/IBM/ldap/V6.0/lib/libibmldap.a /usr/lib/libldap.a
```

---

## PAC-LDAP 許可モジュールを使用可能にする ibmproxy.conf ファイルの編集

PAC-LDAP 許可モジュールを初期化するには、3 つのディレクティブ ServerInit、Authorization または Authentication、および ServerTerm を、ibmproxy.conf ファイルの API ディレクティブ・セクションに追加する必要があります。これらのディレクティブを作成するには、ibmproxy.conf ファイルを手作業で編集するか、あるいはプ

ロキシー・サーバーがすでに実行中の場合にはインターネット・ブラウザで「構成および管理」フォームに接続して「API 要求処理」フォームを開きます（「サーバー構成」→「要求処理」→「API 要求処理」をクリックします）。このセクションの例では分かりやすくするために改行を入れてありますが、プロキシー構成ファイルではそれぞれのディレクティブは単一行にある必要があります。

ibmproxy.conf ファイルの API セクションには、(コメントの形で) プロトタイプ・ディレクティブが提供されていることに注意してください。この API ディレクティブは目的別の順序で配列されています。API ディレクティブを追加して新しい機能やプラグイン・モジュールを使用できるようにするには、各ディレクティブを構成ファイルのプロトタイプ・セクションに示されているように配列してください。あるいは、必要に応じて API ディレクティブをアンコメントして編集し、それぞれ必要な機能やプラグインに対するサポートを組み込んでください。

ServerInit ディレクティブには 3 つの引き数があります。(1) 共用ライブラリーの完全修飾パス、(2) 関数呼び出し、および (3) paccp.conf ファイルの完全修飾パスです。最初の引き数と 2 番目の引き数はコロン (:) で区切ります。2 番目の引き数と 3 番目の引き数はスペースで区切ります。最初の引き数と 3 番目の引き数はシステム特有で、プラグイン・コンポーネントのインストール場所によって決まります。2 番目の引き数は共用ライブラリーにハードコーディングされるので表示されたとおりに入力する必要があります。「API 要求処理」フォームを使用して ServerInit ディレクティブを作成する場合には、2 番目と 3 番目の両方の引き数を「関数名」フィールドに入力する必要があります。3 番目の引き数は「IP テンプレート」欄に表示されます。

Authorization ディレクティブには 3 つの引き数があります。(1) 要求テンプレート、(2) 共用ライブラリーの完全修飾パス、および (3) 関数名です。HTTP 要求は、アプリケーション機能が呼び出されたかどうかを判断するために、要求テンプレートと比較されます。要求テンプレートには、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、/front\_page.html、http://www.ics.raleigh.ibm.com、/pub\*、/\*、および \* はすべて有効です。関数名は、プログラムの中でアプリケーション機能に与えられた名前です。これはハードコーディングされるので、表示されたとおりに入力する必要があります。最初の 2 つの引き数はスペースで区切ります。最後の 2 つの引き数はコロン (:) で区切ります。

Authentication ディレクティブには 2 つの引き数があります。(1) 共用ライブラリーの完全修飾パスと (2) 関数名です。これらの引き数はコロン (:) で区切られます。最初の引き数はシステム特有で、共用ライブラリーがインストールされている場所によって決まります。Caching Proxy をリバース・プロクシーとして使用しているとき、最初の引き数の URL テンプレートは文書のルート (/) から開始する必要があります。2 番目の引き数は共用ライブラリーにハードコーディングされるので表示されたとおりに入力する必要があります。

ServerTerm ディレクティブには 2 つの引き数があります。(1) 共用ライブラリーの完全修飾パスと (2) 関数名です。これらの引き数はコロン (:) で区切られます。最初の引き数はシステム特有で、共用ライブラリーがインストールされている場所によって決まります。2 番目の引き数は共用ライブラリーにハードコーディングされるので表示されたとおりに入力する必要があります。このディレクティブは、プロ

キシー・サーバーの終了時に PAC デーモンを終了します。このデーモンの所有者がプロキシー・サーバーの所有者と違う場合には、プロキシー・サーバーはそのデーモンを停止させることができません。この場合には、管理者がデーモンを手作業で停止する必要があります。

```
ServerInit path_of_shared_library:pacwte_auth_init path_of_conf_policy_file
```

Linux および UNIX の例:

```
ServerInit /usr/lib/libpacwte.so:pacwte_auth_init /etc/pac.conf
```

Windows の例:

```
ServerInit C:%Progra~1%IBM%edge%cp%lib%plugins%  
pac%pacwte.dll:pacwte_auth_init C:%Progra~1%IBM%edge%cp  
Authorization request-template path_of_shared_library:pacwte_auth_policy
```

Linux および UNIX の例:

```
Authorization http://* /usr/lib/libpacwte.so:pacwte_auth_policy
```

Windows の例:

```
Authorization http://* C:%Program Files%IBM%edge%cp%lib%plugins%  
pac%pacwte.dll:pacwte_auth_policy  
Authentication BASIC path_of_shared_library:pacwte_auth_policy
```

Linux および UNIX の例:

```
Authentication BASIC /usr/lib/plugins/pac/libpacwte.so:pacwte_auth_policy
```

Windows の例:

```
Authentication BASIC C:%Program Files%IBM%edge%cp%lib%plugins%  
pac%pacwte.dll:pacwte_auth_policy  
ServerTerm path_of_shared_library:pacwte_shutdown
```

Linux および UNIX の例:

```
ServerTerm /usr/lib/libpacwte.so:pacwte_shutdown
```

Windows の例:

```
ServerTerm BASIC C:%Program Files%IBM%edge%cp%lib%plugins%  
pac%bin%pacwte.dll:pacwte_shutdown
```

---

## PAC-LDAP 許可モジュール構成ファイルの編集

PAC-LDAP 許可モジュール構成およびポリシー・ファイルは、テキスト・エディターを使用して手作業で編集する必要があります。ディレクティブ名とその最初の引き数はコロン (:) で区切ります。複数の引き数はコンマ (,) で区切ります。構成およびポリシー・ファイルには、編集時に注釈を組み込みます。以下に、主要なポリシー・ディレクティブを示します。

### paccp.conf

paccp.conf ファイルは、Caching Proxy の初期化中に共用ライブラリーによって読み取られ、開始するそれぞれの PAC デーモンの定義 ([PAC\_MAN\_SERVER] スタンザ)が入っています。それぞれの PAC デーモンには、独自の [PAC\_MAN\_SERVER] スタンザがなければなりません。

```
[PAC_MAN_SERVER]
hostname:                # name of PAC daemon
port:                    # port pacd is listening on

[PACWTE_PLUGIN]
hostname_check:[true|false] # enables DNS lookup. Must have
                             # DNS lookup turned on for ibmproxy to work.
```

## pac.conf

pac.conf ファイルは、PAC デーモンが接続しようとする LDAP サーバーを指定します。

```
[PAC_MAN_SERVER]
hostname:                # name of PAC daemon
port:                    # port pacd is listening on
conn_type:ssl            # comment out if you do not use SSL
authentication_sequence: [primary|secondary|none]
authorization_sequence: [primary|secondary|none]

[LDAP_SERVER]
hostname:                # LDAP Server hostname
port:389                 # Port LDAP is listening on
ssl_port:636             # SSL port used by the LDAP server
admin_dn:                # User with permission to access the LDAP server
                             # specify admin_dn=NULL to enable anonymous binding
search_base:             # Portion of LDAP tree to search for policy info
                             # If not required, specify search_base=NULL
search_key:              # ID field to search

[CACHE]
cred_cache_enabled [TRUE|FALSE] # turn credentials cache on
cred_cache_min_size:100      # minimum number of credentials to cache in pacd
cred_cache_max_size:64000    # maximum number of credentials to cache in pacd
cred_cache_expiration:86400  # when a credential expires
policy_cache_enabled:[TRUE|FALSE] # turns policy cache on/off
policy_cache_min_size:100    # min. number of policy related items to cache
policy_cache_max_size:64000  # max. number of policy related items to cache
policy_cache_expiration:86400 # when a policy related item expires
```

## pacpolicy.conf

すべての LDAP ポリシーは、構成およびポリシー・ファイル内の次のテンプレートを使用します。それぞれのポリシーは、大括弧で囲んだ大文字のキーワード POLICY で始まっていなければなりません。

```
[POLICY]
default_policy:[grant|deny] # describes the default policy for users
                             # that are not described in the POLICY section
pac_client_hostname:        # the instances of Caching Proxy that are allowed
                             # to use a policy list
id:                          # the id for the LDAP entry or ip/hostname
                             # (wildcard supported, such as *.ibm.com)
grant:[true|false]          # true means to grant access, false means
                             # to deny access
type:[0|1|2|3|4]            # 0 LDAP entry that is a group,
                             # 1 LDAP entry that is not a group,
                             # 2 IP address
                             # 3 hostname
                             # 4 URL
propagate:[true|false]      # true means that the access rights (grant
                             # or deny) will be propagated to all
                             # descendants or members
stop_entry:[entry|NULL]     # Propagation of the access right stops
                             # at this entry. If the id is a group,
                             # stop_entry must be set to NULL.
```

```

# stop_entry may be applied to an IP
# address or hostname. Each stop_entry
# must be on its own line
exception_entry:[entry|NULL] # Assignment of the access right skips
# these entries, but continues through their
# subtrees. This may be a list of entries.
# exception_entry may be applied to a group,
# IP address, or hostname. Each
# exception_entry must be on its own line.

Exception_type:
Exception:

```

ワイルドカード (\*) がサポートされるのは、id および stop\_entry ディレクティブの IP アドレスの最後の桁か、ホスト名の最初の桁だけです。ワイルドカードは、exception\_entry ではサポートされません。ワイルドカードは、いずれのフィールドの LDAP 項目でもサポートされません。

複数のポリシーがサポートされ、ポリシーが競合する場合には、偽の値が常に優先します。言い換えれば、ポリシー内の単一の否定でアクセスが阻止されます。構成およびポリシー・ファイルにポリシーがリストされている順序は無関係で、優先順位を設定するものではありません。

一連のポリシーの例では、構成ファイル・ディレクトリーの pacpolicy.conf ファイルを参照してください。

注: ネストされたグループは、親グループからポリシーを継承しません。グループで適用される唯一のポリシーは、そのグループが明示的なメンバーであるポリシーだけです。

---

## pac\_ldap.cred の作成

pac\_ldap.cred という名前のプレーン・テキストを /cp\_root/server\_root/pac/creds 内に作成します。このファイルには、pac.conf ファイル内の admin\_dn ディレクティブのユーザー名に対応するパスワードが入ります。

注: 匿名バインディングを可能にするには、pac.conf 内の admin\_dn ディレクティブを admin\_dn:NULL に変更し、pac\_ldap.cred ファイルにダミーのストリングを入れてください。

PAC デーモンは、初めてそのファイルを読み取ったときにパスワードを暗号化します。

ファイル pac\_ldap.cred を Linux および UNIX プラットフォームで作成するには、次のコマンドを実行します。

```

cd cp_root/server_root/pac/creds
echo "password" > pac_ldap.cred
chown nobody pac_ldap.cred
chgrp nobody pac_ldap.cred
(on SUSE Linux, use chgrp nogroup pac_ldap.cred.)

```

このファイルを Windows プラットフォームで作成するには、テキスト・ファイルにパスワードを入力し、そのファイルを server\_root¥pac¥creds¥ ディレクトリーに格納します。

## pacd の始動および停止

LDAP 許可デーモンは、pacd プロセスとして実行します。規定のスクリプトを使用して、Caching Proxy に割り込まずに LDAP 許可を再始動することができます。次の手順で pacd スクリプトを実行します。

- Linux および UNIX プラットフォームの場合:

```
/usr/sbin/pacd_restart.sh pacd_user_id
```

- Windows プラットフォームの場合:

```
C:\Program Files\IBM\edge\cp\Bin\pacd_restart.bat CP_install_root
```

注: AIX システムの場合は **stopsrc -ibmproxy** コマンドを、HP-UX、Linux および Solaris システムの場合は **ibmproxy -stop** コマンドを使用して、Caching Proxy サーバーをシャットダウンした後も、pacd プロセスの稼働を継続することが可能です。以下のような **kill** コマンドを使用することによって、pacd プロセスを安全に終了させることができます。

```
kill -15 pacd_process_ID
```

**HP-UX の場合:** PAC-LDAP プラグインおよび pacd は、実行時にすべての従属共有ライブラリーをロードしません。それらを使用する前に、システム変数が次のように設定されていることを確認してください。

```
SHLIB_PATH=/usr/lib:/usr/IBMldap/lib  
PATH=/usr/IBMldap/bin:$PATH  
PATH=/usr/IBMldap/bin
```

/usr/IBMldap/ は、HP-UX の LDAP クライアント用のデフォルトのインストール・パスです。LDAP クライアントが別のロケーションにインストールされている場合は、それに応じて PATH および SHLIB\_PATH を調整してください。これらの変数を設定しないと、次のエラーが発生する可能性があります。

- PAC-LDAP プラグインを使用可能にすると、エラー・ログに次のメッセージが表示されます。

```
"Serverinit Error: server did not load functions  
from DLL module /opt/ibm/edge/cp/lib/plugins/pac/libpacwte.sl"
```

- /usr/sbin/pacd を開始しようとする、次のリンク・エラーが表示されます。

```
"/usr/lib/dld.sl: Can't find path for shared library: libibmldap.sl  
/usr/lib/dld.sl: No such file or directory  
Abort"
```

**Linux の場合:** SUSE Linux Enterprise Server 9 の場合、ldd pacd は、libldap.so が見つからない、とレポートことがあります。この問題に対処するには、以下のシンボリックを作成してください。

```
ln -s /usr/lib/libldap.so.19 /usr/lib/libldap.so
```

**AIX の場合:** IBM Tivoli Directory Server 5.2 で pacd を開始する場合は、PAC-LDAP モジュールがロードできずに、次のエラーが出される可能性があります。

```
exec(): 0509-036 Cannot load program /usr/sbin/pacd because of the following errors:  
0509-022 Cannot load module /usr/lib/libpacman.a.  
0509-150 Dependent module libldap.a could not be loaded.  
0509-022 Cannot load module libldap.a.
```

この問題に対処するには、以下のシンボリックを作成してください。

```
ln -s /usr/lib/libibmldap.a /usr/lib/libldap.a
```

**注:** LDAP 認証を使用するために Caching Proxy を構成した後に、以下のエラーが表示されます。

```
Could not extract a value for: Uid, return code:3
```

このエラーは、LDAP 認証が正しく機能している場合でも表示されますので、無視することができます。



---

## 第 6 部 Caching Proxy のモニター

ここでは、ログおよびサーバー・アクティビティー・モニターを使用した Caching Proxy のモニターについて説明します。

ここには、次の章が含まれます。

159 ページの『第 30 章 ログの構成』

167 ページの『第 31 章 サーバー・アクティビティー・モニターの使用』



---

## 第 30 章 ログの構成

ログをカスタマイズするために、「構成および管理」フォームを使用するか、またはプロキシ構成ファイルのディレクティブを編集することができます。以下のオプションを設定できます。

- ログ・ファイルを格納するパスおよびファイル名
- アクセス・ログ・ファイルへ情報を組み込んだり、除外したりするフィルター
- ログをアーカイブまたは削除する保守オプション

---

### ログについて

Caching Proxy では、3 種類のアクセス・ログに加えて 1 つのイベント・ログと 1 つのエラー・ログを作成できます。

- アクセス・ログ:
  - **アクセス・ログ** — Caching Proxy 自体へのローカル管理要求をトラッキングします。
  - **キャッシュ・アクセス・ログ** — キャッシュ内にあるオブジェクトの要求をトラッキングします。
  - **プロキシ・アクセス・ログ** — 起点サーバーへの代理の要求をトラッキングします。
- イベント・ログ — キャッシュ情報メッセージをトラッキングします。
- エラー・ログ — Caching Proxy に関連するエラー・メッセージをトラッキングします。

Caching Proxy は、毎日、午前 0 時に新規ログ・ファイルを作成します。プロキシが午前 0 時に実行されていない場合、その日の最初に開始されるときに、新規ログが作成されます。それぞれのログ・ファイルにディレクトリーとファイル名接頭部を指定することができます。また、作成されるそれぞれのログ・ファイルには、`.Mmmddyyyy` 形式の日付接尾部 (例えば、`.Apr142000`) が含まれます。

ログは大量のスペースを使用することがあるので、エラーを回避するために、ログ・ファイルをオペレーティング・システムおよびキャッシュとは別のストレージ・デバイスに格納することを検討してください。また、163 ページの『ログの保守とアーカイブ』で示すようなログ保守ルーチンを構成してください。

---

### ログ・ファイル名と基本オプション

プロキシ・サーバー・ログの基本構成を指定するには、「構成および管理」フォームで、「サーバー構成」->「ログ記録」->「ログ・ファイル」を選択します。使用するログ・ファイルごとに、パス名およびファイル名を指定します。各ログの現在のファイル名が対応するテキスト・ボックスに表示されます。パスを指定していない場合には、デフォルトのパスが表示されます。

プロキシ・ログに記録された情報は、システム・ログに自動的に書き込まれるのではなく、独自のログの他に追加で、またはそのログの代わりにシステム・ログに書き込むように Caching Proxy を構成することができます。「**ログ・ファイル**」フォームで、「**情報を Syslog へ記録**」チェック・ボックスを選択します。このオプションを選択する前に、システム・ログを作成しておく必要があることに注意してください。

プロキシ・サーバー・ログ情報をシステム・ログのみに書き込むように指定するには、プロキシ構成ファイルを編集する必要があります。252 ページの『LogToSyslog - アクセス情報をシステム・ログに送信するかどうかを指定する (Linux と UNIX 専用)』を参照してください。

### 関連する構成ファイル・ディレクティブ

プロキシ構成ファイルを使用することによりログをセットアップするには、185 ページの『付録 B. 構成ファイル・ディレクティブ』の以下のディレクティブについての解説セクションを参照してください。

- 187 ページの『AccessLog - アクセス・ログ・ファイルのパスを指定する』
- 201 ページの『CacheAccessLog - キャッシュ・アクセス・ログ・ファイルのパスを指定する』
- 227 ページの『ErrorLog - サーバー・エラーがログに記録される場所のファイルを指定する』
- 230 ページの『EventLog - イベント・ログ・ファイルのパスを指定する』
- 252 ページの『LogToSyslog - アクセス情報をシステム・ログに送信するかどうかを指定する (Linux と UNIX 専用)』
- 256 ページの『MaxLogFileSize - 各ログ・ファイルの最大サイズを指定する』
- 283 ページの『ProxyAccessLog - プロキシ・アクセス・ログ・ファイルのパスを指定する』

---

## アクセス・ログ・フィルター

アクセス・ログは、ホスト・マシン、プロキシ、およびキャッシュの活動を記録します。プロキシがアクセス要求を受信するたびに、以下の情報を含む項目が適切なアクセス・ログに作成されます。

- 何が要求されたか
- いつ要求されたか
- 誰が要求したか
- 要求のメソッド
- サーバーがその要求に応じて送信したファイルのタイプ
- 要求が正常に行なわれたかどうかを示す戻りコード
- 送信されたデータのサイズ

アクセス・エラーは、サーバーのエラー・ログに記録されます。

## ログを制御する理由

ログを制御する理由は、いくつかあります。

- ログ・サイズを小さくするため

使用頻度の高いサーバーのログ・ファイルは、サーバーのディスク・スペースがいっぱいになるほど大きくなることがあります。デフォルトでは、すべてのアクセス要求がログに記録されるようになっていますが、そのことは、HTML ページだけでなくそのページに含まれるそれぞれのイメージについてもログ項目が作成されることを意味します。有意義なアクセス要求だけを含めることによって、ログの項目数を大幅に減らすことができます。例えば、アクセス・ログを構成して、HTML ページに対するアクセス要求を示すログ項目だけを含めて、GIF イメージのアクセス要求の項目は含めないようにすることができます。

- 対象となる情報を収集するため

例えば、社外からサーバーにアクセスしている人を知りたい場合は、社内の IP アドレスから出るアクセス要求をフィルターに掛けて除外します。特定の Web サイトの利用者数を知りたい場合は、その URL に対するアクセス要求のみを示すアクセス・ログを作成することができます。

アクセス・ログから除外された情報は、アクセス・レポートには記録されず、後で使用することはできません。したがって、どの程度のアクセス情報をトラッキングする必要があるかどうかわからない場合は、サーバーのモニターに関して経験を得るまで、除外フィルターの適用を控えめにします。

## アクセス・ログ・フィルターの構成

アクセス・ログ項目は、次のいずれかの属性に基づいてフィルターに掛けることができます。

- URL (ファイルまたはディレクトリー)
- IP アドレスまたはホスト名
- ユーザー・エージェント
- メソッド
- MIME タイプ
- 戻りコード

フィルターを指定するには、「構成および管理」フォームで、「サーバー構成」→「ログ記録」→「アクセス・ログ除外」を選択します。除外する必要があるもののみ指定します。すべてのカテゴリーを使用する必要はありません。

- 「アクセス・ログ内の次のディレクトリーまたはファイルへの要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外する URL パスをリストします。
- 「次のユーザー・エージェントからの要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外するプロキシ・エージェントをリストします。
- 「次のホスト名または IP アドレスからの要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外するホスト名または IP アドレスをリストします。

- 「次のメソッドを使用した要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外するすべてのメソッドのボックスにチェックマークを付けます。
- 「次の MIME タイプのファイルへの要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外するすべての MIME タイプのボックスにチェックマークを付けます。

注: このディレクティブは、プロキシー・アクセス・ログにのみ影響を与えます。それらの MIME タイプによって、これらのキャッシュされたオブジェクトのログ・リスト作成をフィルターに掛けることはできません。これを行うためには、AccessLogExcludeURL を使用してください。

- 「次の戻りコードを伴う要求はログに記録しない」というヘッダーの付いたセクションで、ログ項目を除外する要求戻りコードのボックスにチェックマークを付けます。

「実行依頼」をクリックします。

### 関連する構成ファイル・ディレクティブ

プロキシー構成ファイルを使用することによりアクセス・ログ・フィルターを設定するには、185 ページの『付録 B. 構成ファイル・ディレクティブ』の以下のディレクティブについての解説セクションを参照してください。

- 188 ページの『AccessLogExcludeMethod - 指定されたメソッドに必要なファイルまたはディレクトリーのログ項目を抑制する』
- 189 ページの『AccessLogExcludeMimeType - 特定の MIME タイプのプロキシー・アクセス・ログ項目を抑制する』
- 190 ページの『AccessLogExcludeReturnCode - 指定の戻りコードのログ項目を抑制する』
- 190 ページの『AccessLogExcludeURL - 特定のファイルまたはディレクトリーのログ項目を抑制する』
- 191 ページの『AccessLogExcludeUserAgent - 特定のブラウザからのログ項目を抑制する』
- 262 ページの『NoLog - テンプレートと一致する特定のホストまたはドメインのログ項目を抑制する』

---

## デフォルトのログ設定値

### デフォルト・パス

Caching Proxy デフォルト構成では、すべてのログが使用可能にされています。すべてのログは、インストール・ディレクトリーの logs/ サブディレクトリーに格納されます。デフォルト・パスは、次のとおりです。

- ローカル (管理) アクセス・ログ:
  - Linux および UNIX の場合: /opt/ibm/edge/cp/server\_root/logs/local
  - Windows の場合: drive:¥Program Files¥IBM¥edge¥cp¥logs¥local
- キャッシュ・アクセス・ログ:
  - Linux および UNIX の場合: /opt/ibm/edge/cp/server\_root/logs/cache

- Windows の場合: `drive:¥Program Files¥IBM¥edge¥cp¥logs¥cache`
- プロキシ・アクセス・ログ:
  - Linux および UNIX の場合: `/opt/ibm/edge/cp/server_root/logs/proxy`
  - Windows の場合: `drive:¥Program Files¥IBM¥edge¥cp¥logs¥proxy`
- エラー・ログ:
  - Linux および UNIX の場合: `/opt/ibm/edge/cp/server_root/logs/error`
  - Windows の場合: `drive:¥Program Files¥IBM¥edge¥cp¥logs¥error`
- イベント・ログ:
  - Linux および UNIX の場合: `/opt/ibm/edge/cp/server_root/logs/event`
  - Windows の場合: `drive:¥Program Files¥IBM¥edge¥cp¥logs¥event`

各ログ・ファイル名は、ベース名と `.Mmmddyyyy` という形式の日付接尾部を組み合わせた名前になります (例えば、`proxy.Feb292000`)。

#### • デフォルト形式

デフォルトでは、ログは共通ファイル形式で格納されます。結合ログ形式を使用することもでき、次の行をプロキシ構成ファイル (`ibmproxy.conf`) に追加することによって設定できます。

```
LogFileFormat combined
```

結合ログ形式は共通形式と似ていますが、参照者、ユーザー・エージェント、および Cookie 情報を表示する追加のフィールドがあります。地方時形式がデフォルトの時間形式です。

#### • デフォルト・コンテンツ

デフォルトでは、アクセス要求はすべて適切なアクセス・ログに記録されて、システム・ログにはアクセス情報は記録されません。エラー・ログ情報はエラー・ログにのみ書き込まれ、イベント・ログ情報はイベント・ログにのみ書き込まれます。

#### • デフォルト保守

デフォルト構成では、ログはアーカイブも削除もされません。

---

## ログの保守とアーカイブ

Caching Proxy は、プラグインを使用してログを管理します。詳細については、185 ページの『付録 B. 構成ファイル・ディレクティブ』の構成ファイル・ディレクティブについて 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』の解説ページを参照してください。

日々のログをアーカイブしたり、ログを除去する方法を指定することができます。基本的なオプションとして、次のものがあります。

- ログを圧縮して、指定した経過日数より古いログを除去する。
- 一定の経過日数またはそのログ・カテゴリが一定の総量に達してからログを削除する
- 毎晩午前 0 時に独自のプログラムを実行して、ログを保守しアーカイブする。



デフォルトでは、本日と前日のログが保守エージェントによって削除されることはありません。本日のログおよび前日のキャッシュ・アクセス・ログのすべてが、保守エージェントによって圧縮されることはありません。

ログ保守を構成するには、「構成および管理」フォームで、「**サーバー構成**」→「**ログ記録**」→「**ログ・アーカイブ**」を選択します。このフォームで、ドロップダウン・ボックスを使用して保守メソッドを指定します。

- ・「**パージ**」を選択した場合は、削除するログを決定するのに使用する経過日数、ファイル・サイズ、または両方を設定します。経過日数とサイズでファイルをパージする場合、最大サイズよりも大きなファイルがパージされる前に、最大経過日数よりも古いファイルがパージされます。サイズでファイルをパージする場合は、古いログから順に削除されます。
- ・「**圧縮**」を選択した場合は、圧縮するログの経過日数と、ログ・ファイル (すべてのパラメーターを含む) の圧縮に使用するコマンドを設定します。また、ログの最大経過日数も設定します。ログを圧縮した後、保守エージェントは、圧縮されたログの中から最大経過日数より古いログを削除します。

#### 関連する構成ファイル・ディレクティブ

プロキシ構成ファイルを使用してログのアーカイブを構成するには、185 ページの『付録 B. 構成ファイル・ディレクティブ』の以下のディレクティブについての解説ページを参照してください。

- ・ 212 ページの『CompressAge - ログをいつ圧縮するかを指定する』
- ・ 214 ページの『CompressDeleteAge - ログをいつ削除するかを指定する』
- ・ 213 ページの『CompressCommand - 圧縮コマンドおよびパラメーターを指定する』
- ・ 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- ・ 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- ・ 288 ページの『PurgeAge - ログの経過時間限度を指定する』
- ・ 288 ページの『PurgeSize - ログ・アーカイブのサイズの限度を指定する』。

---

## ログ・ファイル・シナリオ

次の例では、要件に合わせてログ記録をカスタマイズする方法を示しています。Caching Proxy を購入してインストールした直後を想定します。以下の要件について、サーバーをセットアップして、アクセス情報およびエラー情報をログに記録します。

- ・ ログはローカル・タイム・スタンプを持ち、共通ログ・ファイル形式にします。
- ・ アクセス・ログは、30 日を超えるか、またはログが 25 MB の合計サイズに達したときにパージしなければなりません。
- ・ アクセス・ログには、以下の要求タイプは記録する必要がありません。
  - GIF イメージに対する要求
  - パターン 130.128.\*.\* と一致する IP アドレスを持つホストからの要求
  - リダイレクト要求 (これらの要求は 300 から 399 の間の戻りコードを返す)

これらの基準に従ってログを保持するように Caching Proxy を構成するには、「構成および管理」フォームで「サーバー構成」->「ロギング」と選択します。

1. オプションで、「ログ・ファイル・フォーム」を選択して、アクセス・ログ・ファイルのパスを指定します。(デフォルト・パスが提供されています)
2. ファイルの保存方法を指定するには、「ログ・アーカイブ」フォームを使用します。
  - アーカイブ・メソッドとして「パージ」を指定します。
  - 「パージを使用する時」の下で、次のようにフィールドに入力します。
    - 30 日以上経過したログを削除する
    - 25 MB を超えるログを削除する
3. 次のようなログ項目をフィルターに掛けるには、「アクセス・ログ除外」フォームを使用します。
  - 「次のホスト名または IP アドレスからの要求はログに記録しない」リストで、「除外されるホスト」フィールドに 130.128.\*.\* を追加します。
  - 「次の MIME タイプのファイルの要求をログに記録しない」の下で、「image/gif」チェック・ボックスを選択します。
  - 「次の戻りコードの要求をログに記録しない」の下で、「(3xx) Redirection」チェック・ボックスを選択します。

これらの指示に従って、プロキシー構成ファイルに次の行が作成されます。

```
LogArchive purge
PurgeAge 30
PurgeSize 25
AccessLogExcludeURL *.gif
NoLog 130.128.*.*
AccessLogExcludeReturnCode 300
```



## 第 31 章 サーバー・アクティビティ・モニターの使用

Caching Proxy のサーバー活動モニターは、サーバーおよびネットワークのパフォーマンス統計、サーバーおよびネットワークの状況、およびアクセス・ログ項目を表示します。このモニターはリモート側で使うことができ、プロキシ・サーバーを実行している同じマシン上になくてもかまいません。サーバー活動はデフォルトで使用可能にされており、構成は必要ありません。

サーバー活動モニターを開くには、以下の 2 とおりあります。

- 接続されているブラウザで、サーバーの完全名を使用して次の URL を入力します。

`http://your.server.name/Usage/Initial`

- 「構成および管理」フォームで、「サーバー活動モニター」を選択します。

構成クライアント内のその他のフォームとは異なり、このカテゴリのフォームでは、サーバーの構成を設定するのではなく、サーバーの使用状況に関するデータが表示されます。これらのフォームでは、単一のコンソール・ウィンドウに表示できる情報よりも相当に多くの情報が提供されます。

以降のセクションでは、「サーバー活動モニター」が提供する情報のタイプ、およびその情報を使用してパフォーマンスを調整する方法を示しています。

以下の「サーバー活動モニター」ページを使用できます。

- 活動統計
- ネットワーク統計
- アクセス統計
- プロキシ・アクセス統計
- キャッシュ統計
- キャッシュ・リフレッシュ要約

これらのページのそれぞれには「最新表示」ボタンがあり、これを使用すると、情報を更新できます。

### 活動統計

表 4 は、「活動統計」ページの例です。

表 4. 活動統計

活動統計	
接続	1 アクティブ、431 最大
応答時間	利用不能
スループット	0 接続/秒
今日処理した要求	0
処理した要求の総量	114
要求エラー	3

これらのサーバー活動統計を使用して、アクセス要求の数、応答時間、スループット、今日処理した要求数、処理済みの合計要求数、エラーなどによって、サーバーのトラフィックをモニターすることができます。以下の構成変更によって、「活動」ページの統計が影響を受けます。

- **アクティブ・スレッドの数** — これは、サーバー要求に使用するスレッドの数を指定します。使用可能なスレッドの数は、搭載メモリーの大きさに応じて増減できます。アクティブ・スレッドの数を変更するには、「構成および管理」フォームで、「サーバー構成」->「サーバー管理」->「パフォーマンス」を選択するか、または構成ファイルで `MaxActiveThreads` ディレクティブを編集してください。(256 ページの『`MaxActiveThreads` - アクティブ・スレッドの最大数を指定する』を参照してください。)
- **持続接続** — クライアントとの持続接続がプロキシにより許可されることは、ネットワークのスループットに影響することがあります。この設定を変更するには、「構成および管理」フォームで、「プロキシ構成」->「プロキシ・パフォーマンス」を選択して持続接続を使用可能または使用不可にし、さらに「サーバー構成」->「サーバー管理」を選択して接続の維持方法を定義してください。構成ファイルを使用することによりこれらの設定を変更するには、以下のディレクティブを参照してください。
  - 257 ページの『`MaxPersistRequest` - 持続接続で受信する要求の最大数を指定する』
  - 269 ページの『`PersistTimeout` - クライアントが別の要求を送信するのを待機する時間を指定する』
  - 285 ページの『`ProxyPersistence` - 持続接続機能を許可する』

## ネットワーク統計

表 5 は、「ネットワーク統計」ページの例です。

表 5. ネットワーク統計

ネットワーク統計	
送信データ:	1KB/秒
着信データ:	1KB/秒
保管された帯域幅:	3KB (0KB/秒)
今日保管された帯域幅:	0KB (0KB/秒)

「ネットワーク統計」フォームは、プロキシが実行中のネットワークに関する情報 (送受信バイト数によるデータ速度を含む) を提供します。

## アクセス統計

「アクセス統計」ページには、アクセス・ログの中の最新の 20 項目が表示されます。このページには、プロキシ・アクセス・ログ (黒のタイプ) およびキャッシュ・アクセス・ログ (青のタイプ) の中の最新の項目が表示されます。ログの対象となるものをカスタマイズすることによって、表示内容をカスタマイズできます。アクセス・ログ統計についての詳細は、160 ページの『アクセス・ログ・フィルター』を参照してください。

## プロキシ・アクセス統計

「プロキシ・アクセス統計」フォームでは、どの URL が要求されたか、URL がキャッシュから提供されたことなど、プロキシ活動に関する情報が提供されます。URL のあとには、クライアントに提供される戻りコードとバイト単位のファイル・サイズが示されます。以下の設定によって、プロキシ・アクセス統計を改善できます。

- 要求された文書がキャッシュに入っている確率を高めるには、自動キャッシュ・リフレッシュを使用します。詳しくは、99 ページの『第 20 章 自動リフレッシュおよびプリロードのためのキャッシュ・エージェントの構成』を参照してください。
- キャッシュ・ファイルの最小保持時間を長くします。詳しくは、94 ページの『キャッシュの新しさの構成』を参照してください。
- ローカル・ドメインから提供されるファイルはキャッシュに入れないようにします。この設定によって、キャッシュから要求が満たされる機会が減少する傾向がありますが、ファイルがローカル・イントラネット上のサーバーから提供される速度がキャッシュから提供される速度と同程度であれば、パフォーマンスが低下することはありません（より速い場合もあります）。詳しくは、87 ページの『第 18 章 キャッシュ対象の制御』を参照してください。

### キャッシュ統計

キャッシュが使用可能な場合、「**キャッシュ統計**」ページには最新のキャッシュ・アクセス情報が表示されます。以下のようなキャッシュと索引に関する情報が提供されます。

- キャッシュが現在作動可能かどうか、またはサーバー始動から再索引付けされているかどうか。
- ガーベッジ・コレクションが実行中かどうか。
- キャッシュのヒット率

多くのキャッシュ構成オプションでキャッシュ統計結果が変更されます。（75 ページの『第 4 部 プロキシ・サーバー・キャッシングの構成』を参照）

### キャッシュ・リフレッシュ要約

キャッシュ・エージェントがキャッシュにファイルをプリロードするように構成されている場合には、「**キャッシュ・リフレッシュ要約**」ページにキャッシュ・エージェントの最後の実行に関する情報が表示されます。情報を表示するには、キャッシュ・エージェントが少なくとも一度は実行されている必要があります。キャッシュ・エージェントのリフレッシュ方法を変更するには、以下の事項について検討してください。

- イントラネット上のトラフィックのほとんどがローカル Web サイトへのものでない場合は、ローカル・ドメイン用のキャッシュを使用不可にすることを検討してください。詳しくは、87 ページの『第 18 章 キャッシュ対象の制御』を参照してください。
- キャッシュ・アクセス・ログに表示されないページを多くのクライアントが要求する場合、ロードされる URL を手動で設定してください。詳しくは、104 ページの『関連したプロキシ構成ファイル・ディレクティブ』を参照してください。

- プリロードする頻繁にアクセスされる URL の数を調整してください。詳しくは、104 ページの『関連したプロキシー構成ファイル・ディレクティブ』を参照してください。
- キャッシュ・エージェントが実行を許可される最長時間を指定してください。詳しくは、104 ページの『関連したプロキシー構成ファイル・ディレクティブ』を参照してください。



---

## 付録 A. Caching Proxy コマンドの使用

この付録には、プロキシー・サーバー・コマンドのリファレンスを記載してあります。

---

## cgiparse コマンド

### 目的

**cgiparse** コマンドは、CGI スクリプトの `QUERY_STRING` 環境変数を解析するために使用します。`QUERY_STRING` 環境変数が設定されていない場合は、コマンドは、標準入力から `CONTENT_LENGTH` で指定された文字数だけ文字を読み取ります。戻される出力はすべて、標準出力に書き込まれます。

### 形式

```
cgiparse -Flag [Modifier]
```

### パラメーター

フラグ、フラグと等価な 1 文字 (`-k -f -v -r -i -s -p -c -q -P`)、およびフラグの機能を以下に示します。

#### **-keywords | -k**

`QUERY_STRING` をキーワードとして構文解析します。キーワードはデコードされ、標準出力に 1 行に 1 つずつ書き込まれます。

#### **-form | -f**

`QUERY_STRING` をフォーム要求として解析します。シェルによる評価が行われる際に、接頭部 が `FORM_` で、それにフィールド名が続くシェル変数を設定するストリングを戻します。フィールド値は、変数の内容です。

#### **-value field-name | -v field-name**

`QUERY_STRING` をフォーム要求として解析します。*field-name* の値のみを戻します。

#### **-read | -r**

標準入力から `CONTENT_LENGTH` 文字を読み取り、それらを標準出力に書き出します。

#### **-init | -i**

`QUERY_STRING` が設定されていなければ、標準入力の値を読み取り、`QUERY_STRING` をこの値に設定する `SET` ステートメントを出力します。これは、`GET` および `POST` の両方のメソッドに使用することができます。次に、典型的な使用例を示します。

```
eval 'cgiparse -init'
```

このコマンドが呼び出されると、`GET` または `POST` のどちらが使用されたかに関係なく、`QUERY_STRING` 環境変数が設定されます。

**cgiparse** は、`GET` メソッドが使用される場合には同じスクリプトで複数回呼び出すことができますが、`POST` メソッドが使用される場合には、一度だけ呼び出します。`POST` メソッドの場合、標準入力を読み取られた後、2 回目に **cgiparse** を呼び出しても標準入力は空であり、無期限に待機することになります。

#### **-sep separator | -s separator**

複数の値を区切るために使用されるストリングを指定します。**-value** フラグを

使用している場合、デフォルトの区切り記号は改行です。**-form** フラグを使用している場合、デフォルトの区切り記号はコンマ (,) です。

**-prefix** *prefix* | **-p** *prefix*

このフラグは **-POST** および **-form** とともに使用され、環境変数名を作成する際に使用する接頭部を指定します。デフォルトは **FORM\_** です。

**-count** | **-c**

**-keywords**、**-form**、および **-value** とともに使用されて、これらのフラグに関連する項目のカウントを出力します。

**-keywords** | **-k**

キーワードの数を戻します。

**-form** | **-f**

固有のフィールドの数を戻します (複数の値は 1 とカウントされます)。

**-value** *field-name* | **-v** *field-name*

*field-name* の値の数を出力します (*field-name* という名前のフィールドがない場合は 0 が出力されます)。

**-number**

**-keywords**、**-form**、および **-value** とともに使用されて、これらのフラグに関連する指定されたオカレンスを戻します。

**-keywords**

*n* 番目のキーワードを戻します。(例えば、**-2 -keywords** と指定すると、2 番目のキーワードが出力されます。)

**-form**

*n* 番目のフィールドの値すべてを出力します。(例えば、**-2 -form** と指定すると、2 番目のフィールドの値すべてが出力されます。)

**-value** *field-name*

フィールド *field-name* の複数の値のうち *n* 番目の値を出力します。(例えば、**-2 -value -whatsit** と指定すると、**whatsit** フィールドの 2 番目の値が出力されます。)

**-quiet** | **-q**

すべてのエラー・メッセージが表示されないようにします。(ただし、ゼロ以外の終了状況はエラーを示しています。)

**-POST** | **-P**

標準入力 (または、ファイル名が意図されている場合は、**stdin** ファイル) からの情報は直接デコードされ、解析されてシェル変数となります。

**QUERY\_STRING** は使用されません。**-POST** は、**-init** と **-form** を連続して使用するのと同じです。

## 例

以下の例では、実際には **QUERY\_STRING** がすでにサーバーによって設定されているという事実を無視しています。これらの例において、**\$** は Bourne シェル (B シェル) のプロンプトです。

- キーワード検索

```
$ QUERY_STRING="is+2%2B2+really+four%3F"
$ export QUERY_STRING
$ cgifparse -keywords
is
2+2
really
four?
$
```

- すべてのフォーム・フィールドの構文解析

```
$ export QUERY_STRING="name1=Value1&name2=Value2%3f+That%27s+right%21";
$ cgifparse -form
FORM_name1='Value1'; FORM_name2='Value2? That'¥'s right!'
$ eval `cgifparse -form`
$ set | grep FORM
FORM_name1="Value1"
FORM_name2="Value2? That's right!"
$
```

- 1 つのフィールド値だけの抽出

```
$ QUERY_STRING="name1=value1&name2=Second+value%3F+That'¥'s%27s
$ cgifparse -value name1
value1
$ cgifparse -value name2
Second value? That's right!
$
```

## 結果

- 0 成功
- 1 コマンド行に違反がある。
- 2 環境変数の設定に誤りがある。
- 3 要求された情報を入手できなかった (該当フィールドがない、フォーム・フィールド値が要求されたときに `QUERY_STRING` にキーワードが入っていた、など)。

注: これらのエラー・コードの 1 つを受信するときには、追加の通知メッセージも受信することがあります。メッセージは発行されたコマンドにより異なります。

---

## cgiutils コマンド

### 目的

nph (no-parse header) プログラムで **cgiutils** コマンドを使用して、完全な HTTP 1.0 応答を生成します。

注: ユーザー固有の nph (no-parse header) をプログラムを実行してユーザー固有の戻り値を戻したい場合、そのプログラムの名前は **nph-** で始まらなければなりません。このようにしておくと、サーバー・ヘッダーは、ユーザー固有の戻り値を標準サーバー戻り値でオーバーライドしません。

### 形式

```
cgiutils -Flag [Modifier]
```

*Modifier* にブランクが含まれる場合は、それを引用符 (") で囲ってください。

### パラメーター

#### **-version**

バージョン情報を戻します。

#### **-nodate**

**Date:** ヘッダーを戻しません。

#### **-noel**

ヘッダーの後にブランク行を出力しません。これは、最初のヘッダー行の後に他の MIME ヘッダーを出力したい場合に役立ちます。

#### **-status nnn**

HTTP ヘッダーだけでなく、状況コード *nnn* 付きの完全な HTTP 応答を生成します。**Expires:** ヘッダーだけが必要な場合は、このフラグを使用しないでください。

#### **-reason explanation**

HTTP 応答の理由行を指定します。このフラグは、**-status nnn** フラグを指定した場合にのみ使用できます。

#### **-ct [type/subtype]**

MIME Content-Type ヘッダーを指定します。次の例では、text/html という MIME コンテンツ・タイプを指定します。

```
cgiutils -ct text/html
```

*type/subtype* を省略すると、MIME コンテンツ・タイプは、デフォルトの text/plain に設定されます。次の例は、MIME コンテンツ・タイプを text/plain に戻します。

```
cgiutils -ct
```

#### **-ce encoding**

MIME Content-Encoding ヘッダーを指定します。例えば、次のとおりです。

```
cgiutils -ce x-compress
```

**-cl** *language-code*

MIME Content-Language ヘッダーを指定します。例えば、次のとおりです。

```
cgiutils -cl en_UK
```

**-length** *nnn*

MIME Content-Length ヘッダーを指定します。

**-expires** *Time-Spec*

MIME **Expires:** ヘッダーを指定します。このフラグは、有効期間 (文書の有効期限) を days (日)、hours (時間)、minutes (分)、および seconds (秒) の任意の組み合わせで指定します。これらの値は、文書が有効と見なされる時間を指定します。例えば、次のとおりです。

```
cgiutils -expires 2 days 12 hours
```

この **cgiutils** コマンドは、指定された時間を現在の GMT (グリニッジ標準時) に加算して有効期限日付を求めます。有効期限日付は、HTTP 形式で **Expires:** ヘッダーに書き込まれます。

**-expires now**

**Date:** ヘッダーに一致する **Expires:** ヘッダーを生成します。

**-uri** *URI*

戻される文書の URI (Universal Resource Identifier) を指定します。URI は URL と同じものと考えて構いません。

**-extra** *xxx: yyy*

**cgiutils** コマンドでしか指定できない特別なヘッダーを指定します。

## 例

- 次の例では、**Expires:** ヘッダーの有効期限を計算します。

```
cgiutils -expires "1 year 3 months 2 weeks 4 days 12 hours 30 mins"
```

- 次の例では、状況コードと理由を指定し、**Expires:** ヘッダーを **Date:** ヘッダーに等しく設定します。

```
cgiutils -status 200 -reason "Virtual doc follows" -expires now
```

このコマンドは、次のようなヘッダーを生成します。

```
HTTP/1.0 200 Virtual doc follows
MIME-Version: 1.0
Server: IBM-ICS
Date: Tue, 05 Jan 1996 03:43:46 GMT
Expires: Tue, 05 Jan 1996 03:43:46 GM
```

**Server:** ヘッダーは CGI 環境で使用可能なため、**cgiutils** コマンドは自動的にこのヘッダーを生成します。また、**-nodate** フラグを指定しない限り、**Date:** フィールドも自動的に生成されます。

MIME ヘッダー・セクションの終わりを示すために、出力の後にさらにブランク行が 1 行出力されます。ユーザー固有のヘッダーをさらに出力したい場合は、**-noel** (NO-Empty-Line) フラグを次の例で示すように使用してください。

- ヘッダー行の後にブランク行を出力したくない場合は、**-noel** フラグを使用してください。

```
cgiutils -noel -expires "2 days" -nodate
```

HTTP/1.0 200 Virtual doc follows  
MIME-Version: 1.0  
Server: IBM-ICS  
Expires: Tue, 07 Jan 1996 03:43:46 GMT



---

## htadm コマンド

### 目的

**htadm** コマンドの使用目的は、サーバー・パスワード・ファイルを制御することです。サーバーはパスワード・ファイルを使用して、ファイルへのアクセスを制御します。パスワード・ファイルへユーザー名を追加したり、パスワード・ファイルからユーザーを削除したり、ユーザーのパスワードを検査したり、空のパスワード・ファイルを作成したりすることができます。まずユーザーのパスワードを削除してから、新しいパスワードを作成することによって、ユーザーのパスワードを変更することもできます。

**注:** **htadm** を使用してユーザーの追加、パスワードの変更、またはパスワードの検査を行うときは、パスワードをコマンド行に入力しなければなりません。このコマンドではすぐにコマンド行からパスワードが破棄されるので、マシン上のプロセス・リストを調べても (例えば **ps** コマンドを使用して)、ユーザーのパスワードが判明することはほとんどありません。

### 形式

`htadm -Flag [Modifier]`

### パラメーター

**-adduser** *password-file* *user-name* [*password* [*real-name*]]

ユーザーおよびパスワードをパスワード・ファイルに入れます。*password-file* のみを指定してコマンドを入力すると、他のパラメーターの入力を求めるプロンプトが出されます。

*password-file*

ユーザー名を入れるパスワード・ファイルのパスとファイル名。

*user-name*

入れたいユーザーの名前。

ユーザー名に使用できる文字は英字および数字のみです。特殊文字を使用してはなりません。

指定した名前のユーザーがすでにパスワード・ファイル内にある場合は、このコマンドは失敗します。

*password*

ユーザー名に定義したいパスワード。

パスワードは最大 32 文字の長さです。パスワードに使用できる文字は英字および数字のみです。特殊文字を使用してはなりません。

**注:**

- 一部のブラウザーは、8 文字を超える長さのパスワードを読み取ることも送信することもできません。この制限があるので、8 文字を超える長さのパスワードを定義した場合、サーバーは、パスワード全体を有効と認識するか、またはパスワードの最初の 8 文字のみを有効と認識します。

2. 管理者ユーザー名およびパスワードには、たとえオペレーティング・システムが大/小文字の区別をしない場合でも、大/小文字の区別があります。「構成および管理」フォームにアクセスするときは、`htadm` コマンドを使用して、必ず正確なユーザー名およびパスワードを入力するようにしてください。

*real-name*

追加したユーザー名の識別のために使用したいコメントまたは名前。入力した情報がそのままパスワード・ファイルに書き込まれます。

**-deluser** *password-file* [*user-name*]

パスワード・ファイルからユーザーを削除します。*password-file* のみを指定してコマンドを入力すると、*user-name* パラメーターの入力を求めるプロンプトが出されます。

*password-file*

ユーザー名が削除されるパスワード・ファイルのパスとファイル名。

*user-name*

削除したいユーザーの名前。指定したユーザー名がパスワード・ファイルにない場合は、このコマンドは失敗します。

**-passwd** *password-file* [*user-name* [*password*]]

パスワード・ファイル内にすでに定義されているユーザー名のパスワードを変更します。*password-file* のみを指定してコマンドを入力すると、他のパラメーターの入力を求めるプロンプトが出されます。

*password-file*

パスワード変更の対象であるユーザー名が入っているパスワード・ファイルのパスとファイル名。

*user-name*

パスワードを変更したいユーザー名。指定したユーザー名がパスワード・ファイルにない場合は、このコマンドは失敗します。

*password*

ユーザー名に対して定義したい新規パスワード。

パスワードは最大 32 文字の長さです。パスワードに使用できる文字は英字および数字のみです。特殊文字を使用してはなりません。

**注:**

1. 一部のブラウザーは、8 文字を超える長さのパスワードを読み取ることも送信することもできません。この制限があるので、8 文字を超える長さのパスワードを定義した場合、サーバーは、パスワード全体を有効と認識するか、またはパスワードの最初の 8 文字のみを有効と認識します。
2. 管理者ユーザー名およびパスワードには、たとえオペレーティング・システムが大/小文字の区別をしない場合でも、大/小文字の区別があります。「構成および管理」フォームにアクセスするときは、`htadm` コマンドを使用して、必ず正確なユーザー名およびパスワードを入力するようにしてください。

**-check** *password-file* [*user-name* [*password*]]

パスワード・ファイル内にすでに定義されているユーザー名のパスワードを検査

し、それが正しいかどうかを知らせます。*password-file* のみを指定してコマンドを入力すると、他のパラメーターの入力を求めるプロンプトが出されます。

*password-file*

パスワード検査の対象であるユーザー名が入っているパスワード・ファイルのパスとファイル名。

*user-name*

パスワードを検査したいユーザー名。指定したユーザー名がパスワード・ファイルにない場合は、このコマンドは失敗します。

*password*

検査したいパスワード。入力したパスワードが、ユーザー名に対して定義されているパスワードである場合は、このコマンドは **Correct** を標準出力に書き出し、戻りコード 0 で終了します。入力したパスワードが、ユーザー名に対して定義されているパスワードでない場合は、このコマンドは **Incorrect** を標準出力に書き出します。

**-create** *password-file*

空のパスワード・ファイルを作成します。

*password-file*

作成したいパスワード・ファイルのパスとファイル名。

## 例

- パスワード・ファイルにユーザーを追加するには、次のように入力します。

- Linux および UNIX の場合:

```
htadm -adduser /opt/ibm/edge/cp/server_root/protect/heroes.pwd  
clark superman "Clark Kent"
```

- Windows システムの場合:

```
htadm -adduser "C:\Program Files\IBM\edge\cp\server_root\protect\  
heroes.pwd" clark superman "Clark Kent"
```

**注:** **htadm** コマンドは 1 行で入力する必要があります。上記では、読みやすくするために複数行にしてあります。実際にコマンドを入力する際には、**clark** と **superman** の間に 1 つ以上のスペースを挿入して、すべて 1 行に入力してください。

- パスワード・ファイルからユーザーを削除するには、次のように入力します。

- Linux および UNIX の場合:

```
htadm -deluser /opt/ibm/edge/cp/server_root/protect/  
heroes.pwd clark superman "Clark Kent"
```

- Windows システムの場合:

```
htadm -deluser "C:\Program Files\IBM\edge\cp\server_root\protect\  
heroes.pwd" clark superman "Clark Kent"
```

---

## htcformat コマンド

### 目的

**htcformat** コマンドは、プロキシ・キャッシュを保持するロー・デバイスまたはファイルの準備を行うときに使用します。プロキシ・キャッシュで使用するようデバイスを指定する前に、このフォーマット・コマンドを使用する必要があります。

装置パスでロー・デバイスを指定することが必要です。ロー・デバイスへのアクセス方法の詳細については、使用ファイル・システムの資料を参照してください。75 ページの『第 4 部 プロキシ・サーバー・キャッシングの構成』の例を利用できます。

注: Linux 2.2 カーネルは、ロー・デバイスへのキャッシュをサポートしていません。Linux プラットフォームでは、ファイルとメモリーだけをキャッシュ・ストレージに使用できます。

Caching Proxy キャッシュの最小サイズは 16392 KB であり、2049 ブロックです。

### 形式

```
htcformat device [-blocksize <block size>] [-blocks number of blocks]
htcformat -file filepath [-blocksize block size] -blocks number of blocks
```

### パラメーター

#### **-blocksize**

これは、キャッシュ・デバイスのメディア内のブロックのサイズを設定します。ブロック・サイズは、バイト 単位です。デフォルトは 8192 であり、あらゆる場合にこのデフォルトを使用してください。

#### **-blocks**

デバイス上またはファイル内に作成するブロックの数。ファイルをフォーマットするときは、ファイル・サイズを指定するためにこの引き数は必須です。また、この引き数を使用して、キャッシュ・ストレージ用に使用される特定のデバイスまたは区画の容量を制限することもできます。ブロック引き数を指定しないと、区画に入るだけの数のブロックが作成されます。

#### **-file**

ストレージ・デバイスではなく、ファイルをフォーマットします。

### 使用法

さらに、キャッシュ・ファイルまたはキャッシュ・デバイスは、キャッシュ・システムによって索引化とガーベッジ・コレクションのためにコンテナ内に分離されます。コンテナのサイズは、一定数のブロックに設定されます。つまり、コンテナのサイズを構成することはできません。ガーベッジ・コレクションを実行するために、最低限 2 つのコンテナが必要です。最小キャッシュ・サイズは 16392 KB です。

**htcformat** コマンドでは、コンテナが 2 つ未満のキャッシュ・デバイスを生成するようなフォーマット要求は拒否されます。

## 例

次の例では、c0t0d0s0 というディスク区画を Solaris 上でフォーマットします。

```
htcformat /dev/rdisk/c0t0d0s0
```

次の例では、lv02 というディスク区画を AIX 上でフォーマットします。

```
htcformat /dev/r1v02
```

次の例では、d: というディスク区画を Windows 上でフォーマットします。

```
htcformat ¥¥.¥d:
```

次の例では、filecache という名前のファイルを約 1 GB の大きさにフォーマットします。

```
htcformat -file /opt/ibm/edge/cp/filecache -blocks 131072
```

---

## ibmproxy コマンド

### 目的

**ibmproxy** コマンドは、サーバーを始動するために使用します。

これらのフラグは、(**-r** を除く) すべて、サーバー構成ファイル内のディレクティブを使用して設定することができます。

通常、そのディレクトリーについてよく知らないユーザーが読む指示または注意が書かれている **README** という名前のファイルを作成します。**ibmproxy** コマンドはデフォルトでは、ハイパーテキスト・バージョンのディレクトリーに **README** ファイルを埋め込みます。**README** ファイルの指示は、**DirReadme** 構成ディレクティブで設定することもできます。

### 形式

```
ibmproxy [-Flag [-Flag [-Flag..]]]
```

### パラメーター

#### **-nobg**

サーバーをバックグラウンド・プロセスとしてではなく、フォアグラウンド・プロセスとして実行します。デフォルトでは、バックグラウンド・プロセスとして実行します。

#### **-nosnmp**

SNMP サポートを off にします。

#### **-p** *port-number*

このポート番号を listen します。デフォルトのポート番号は 80 です。このフラグは、構成ファイルで指定された **Port** ディレクティブをオーバーライドします。デフォルト値または構成ファイルで指定された値を使用するときは、このフラグを省略してください。

#### **-r** *configuration-file*

構成ファイルとして使用するファイルを指定します。デフォルト構成ファイル以外の構成ファイルを使用してサーバーを始動したい場合は、このフラグを使用しなければなりません。これによって、複数の構成ファイルが使用できるようになります。

#### **-restart**

現在実行中のサーバーを再始動します。**ibmproxy** コマンドは、実行中のサーバーのプロセス番号を **PidFile** から取得し、そのプロセス番号を **HangUP** シグナル (HUP) に送信します。その後、構成ファイルを再ロードし、ログ・ファイルを再オープンします。ファイルの破壊を回避するために、同じ **PidFile**、ログ・ファイル、およびプロキシー・キャッシュを使用して、サーバーの 2 つのインスタンスを同時に実行しないでください。

**http** デーモンは、サーバーが現在 **PidFile** にアクセスするために使用している構成ファイルを読み取る必要があるため、再始動の際には、同じ構成ファイル

を指定する必要があります。サーバーの始動時に **-r** フラグと特定の構成ファイルを使用した場合、このフラグおよび同じファイルを **-restart** で指定する必要があります。

#### **-snmp**

SNMP サポートを on にします。

#### **-unload**

Linux では、これが、関連するファイアウォール規則を削除します。

また、シグナル処理オプションは Linux および UNIX プラットフォームのみに存在します。Linux および UNIX プラットフォームでは、次のオプションが使用可能です。

#### **SIGTERM**

完了時には **ibmproxy** は停止し、終了します。SIGKILL または CANCEL を使用すると、ただちに終了させることができます。

#### **SIGHUP**

実行中の場合、**ibmproxy** は構成ファイルを再始動して再ロードし、処理を続行します。

## 例

- デフォルトの `/etc/ibmproxy.conf` の代わりに `/usr/etc/ibmproxy.conf` 構成ファイルを使用してサーバーをポート 8080 で始動するには、次のように入力します。

```
ibmproxy -p 8080 -r /usr/etc/ibmproxy.conf
```

- AIX では、システム・リソース制御プログラムを使用して、デフォルトの構成ファイルでサーバーを始動するには、以下のように入力します。

```
startsrc -s ibmproxy
```

デフォルト構成ファイルがない場合、**ibmproxy** は、`/Public` ディレクトリー・ツリーをエクスポートします。このツリーには、他のディレクトリー・ツリーへのソフト・リンクを入れることができます。

---

## 付録 B. 構成ファイル・ディレクティブ

この 付録 では、ibmproxy.conf 構成ファイルに含まれているディレクティブを説明します。

- **Linux** および **UNIX** システムの場合。これらのディレクティブは、/etc/ ディレクトリーの ibmproxy.conf 構成ファイルにあります。
- **Windows** システムの場合。これらのディレクティブは、通常、C:\Program Files\IBM\edge\cp\ にあります。

ibmproxy.conf ファイルを編集してサーバーを構成するときに、この情報を参照してください。「構成および管理」フォームを使用する場合は、この章を参照する必要はありません。

ディレクティブは、アルファベット順にリストされています。

---

### 再始動時に変更されないディレクティブ

ディレクティブの中には、再始動時に更新されないものがあります。サーバーの実行中に以下のディレクティブを変更した場合には、手動でサーバーを停止してから、それを再始動する必要があります。( 15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。)

表 6. 再始動でリフレッシュされないディレクティブ

ディレクティブ・グループ	ディレクティブ
CGI	DisinheritEnv、InheritEnv
Caching	Caching
ロギング	AccessLog、CacheAccessLog、ErrorLog、ProxyAccessLog、ServerRoot
ネットワーク・アクセス	BindSpecific、Hostname、ListenBacklog、Port
パフォーマンス	MaxActiveThreads
RTSP	すべての RTSP ディレクティブ
SSL	すべての SSL ディレクティブ
Linux および UNIX プロセス制御	GroupId、UserId
その他	TransparentProxy

---

### ディレクティブの概要

この付録では、各ディレクティブについて以下の情報を示します。

- ディレクティブの名前および要旨を示すヘッダー
- 使用法の説明
- 以下のように、一般的な構文に従ったディレクティブの形式:

*DirectiveName value*



- 該当する場合は、構成ファイル内のディレクティブの可能な設定の例

**注:** Windows 固有のパスがあるディレクティブの例には、`server_root` が含まれている場合があります。これは、インストール時に選択されたサーバーのルート・ディレクトリーです。

- デフォルトの値またはディレクティブの値

これらの値は、デフォルトの構成ファイルに最初からコーディングされている値です。変更を加える場合は、デフォルト値から、変えたい設定値の部分だけを変更してください。当初からデフォルト値のコーディングされないディレクティブは、ファイルの中で前にコメント・マーカー (#) を付けて表示されています。そのディレクティブに値を指定したい場合には、コメント・マーカーを除去し、値を構成ファイル内の行に追加してください。

## 許容値

以下のリストには、構成ファイルで受け入れられる値が含まれています。

- 一部のディレクティブに対する参照情報では、*value* 部分に要求、パス名、またはホスト名のテンプレートが含まれます。特に断りがない限り、テンプレート内でアスタリスク文字 (\*) を使用することができます。テンプレートを突き合わせる場合、アスタリスクは任意の文字ストリングまたは単一の文字で置き換えることができます。
- 肯定のストリングの入力を可能にする構成ディレクティブは、以下の値を受け入れます。
  - Yes
  - On
  - OK
  - Enable
- 否定のストリングの入力を可能にする構成ディレクティブは、以下の値を受け入れます。
  - No
  - Off
  - None
  - Disable
- 時間の指定を可能にする構成ディレクティブは、次の任意の組み合わせを受け入れます。
  - *hh* — 時間
  - *hh:mm* — 時間および分
  - *hh:mm:ss* — 時間、分、および秒
  - *n years* — 365 日からなる年の数
  - *n months* — 30 日からなる月の数
  - *n weeks* — 7 日からなる週の数
  - *n days* — 24 時間からなる日の数
  - *n hours* — 60 分からなる時間の数
  - *n minutes* — 60 秒からなる分の数 *minutes*
  - *n seconds* — 秒の数

—  $n$  fortnights — 14 日からなる間隔の数

すべての項目は、秒に変換されてから集計されます。

- 構成ファイル内で指定するファイル名に、空白文字は使用できません。空白文字は、区切り文字と見なされます。

---

## 構成ファイル・レコードの構文

構成ファイルを編集する際には、次の要件を忘れないでください。

- 各ディレクティブは新規の行で始める必要があります。
- 値は、1 つ以上のブランクによって区切られます。スペース文字とタブ文字の区別は行われません。
- コメントの先頭は、# 記号で示します。この # 記号からその行の終わりまでの文字はすべて無視されます。
- 番号記号またはブランクをディレクティブに指定する必要がある場合には、それらの前にエスケープ文字として円記号 (¥) を使用してください。エスケープ文字は、次の文字をコマンドとしてではなく、文字として解釈するよう指示します。例えば、ある行に ¥# がある場合に、サーバーは、この文字をコメントの始まりではなく # 記号として解釈し、文字の読み取りを継続します。ある行に ¥ が見つかった場合、サーバーはこの文字を区切り文字でなくブランクと解釈し、引き続き文字を読み取って値を設定します。

---

## Caching Proxy ディレクティブ

Caching Proxy ディレクティブは、以下のとおりです。

### AcceptAnything - すべてのファイルを提供する

このディレクティブは、ファイルの MIME タイプがクライアントから送信された ACCEPT: ヘッダーと一致しない場合であってもそのクライアントにファイルを提供するときに使用します。このディレクティブが OFF に設定されている場合は、クライアントが受け入れ可能なタイプと異なる MIME タイプのファイルは表示されません。代わりに、エラー・ページが表示されます。

#### 形式

AcceptAnything {on | off}

#### 例

AcceptAnything off

#### デフォルト

AcceptAnything on

### AccessLog - アクセス・ログ・ファイルのパスを指定する

このディレクティブを使用して、サーバーがアクセス統計をログに記録する場所のディレクトリーとファイルを指定します。デフォルトでは、クライアントがサーバーに、ローカル・サーバーに格納されているデータを要求するたびに、サーバーがその項目をこのログに書き込みます。通常、これらの項目には、Caching Proxy マシ

ンが起点サーバーとして使用されるときに、構成クライアントからの要求またはアクセスが含まれるだけです。このログには、プロキシまたはキャッシュ・アクセス情報は含まれません。

NoLog ディレクティブは、その要求をログに記録しないクライアントを指定するときに使用します。NoLog ディレクティブについては、262 ページの『NoLog - テンプレートと一致する特定のホストまたはドメインのログ項目を抑制する』を参照してください。

サーバーは、午前 0 時に新規ログ・ファイルを開始します (サーバーが稼働している場合)。午前 0 時にサーバーが稼働していない場合は、その日における最初のサーバー始動時に新規ログ・ファイルの記録を開始します。ファイル作成時に、サーバーは、指定されたファイル名を使用し、日付接尾部を付加します。日付接尾部は、*Mmmddyyyy* という形式です。*Mmm* は月の最初の 3 文字、*dd* は日、*yyyy* は年です。

**注:** サーバーのユーザー ID、グループ ID、あるいはログ・ディレクトリー・パスに対するデフォルトを変更する場合は、新規ディレクトリーを作成し、その許可および所有権を更新します。サーバーが情報をユーザー定義のログ・ディレクトリーに書き込むことができるようにするには、そのディレクトリーの許可を 755 として設定し、ユーザー定義のサーバー・ユーザー ID を所有者として設定します。例えば、サーバーのユーザー ID をデフォルトから *jdoe* に変換し、デフォルト・ログ・ディレクトリーを *server\_root/account* に変更すると、*server\_root/account* ディレクトリーの許可は、755 になり、*jdoe* によって所有されます。

古いログ・ファイルは、ハード・ディスク上で大量のスペースを使用する可能性があるため、これらのファイルは除去するようにしてください。

## 形式

AccessLog */directory\_path/logfile\_name*

## 例

AccessLog */logs/accesslog*

## デフォルト

- **Linux** および **UNIX** システム: AccessLog  
*/opt/ibm/edge/cp/server\_root/logs/local*
- **Windows** システム: AccessLog *drive:%Program Files%IBM%edge%cp%logs%local*

## AccessLogExcludeMethod - 指定されたメソッドに必要なファイルまたはディレクトリーのログ項目を抑制する

このディレクティブは、ファイルまたはディレクトリーにアクセスするために特定のメソッドによって行われた要求のロギングは防止するときに使用します。例えば、ファイルまたはディレクトリーに対する DELETE 要求はログに記録したくない場合があります。

このディレクティブは、構成ファイル内に複数回指定することができます。また、1 つまたは複数のスペースで区切れば、1 つのディレクティブに複数のメソッドを指定することもできます。

## 形式

`AccessLogExcludeMethod method [...]`

## 例

```
AccessLogExcludeMethod GET
AccessLogExcludeMethod PUT
AccessLogExcludeMethod POST
AccessLogExcludeMethod DELETE
AccessLogExcludeMethod GET PUT
```

## デフォルト

なし。サーバーは、すべての種類のメソッドに必要なファイルとディレクトリーをアクセス・ログに記録します。

## AccessLogExcludeMimeType - 特定の MIME タイプのプロキシ・アクセス・ログ項目を抑制する

このディレクティブは、指定の MIME タイプのディレクトリーまたはファイルに対するアクセスの要求をプロキシ・アクセス・ログに記録したくないことを指定するときに使用します。(MIME タイプの例としては、`text/html`、`image/gif`、および `image/jpeg` があります。) 例えば、GIF イメージへのアクセス要求を記録しないようにすることができます。

このディレクティブは、構成ファイル内に複数回指定することができます。また、1 つまたは複数のスペースで区切れば、1 つのディレクティブに複数の MIME タイプを指定することもできます。

**注:** このディレクティブは、プロキシ・アクセス・ログにのみ影響を与えます。それらの MIME タイプによって、これらのキャッシュされたオブジェクトのログ・リスト作成をフィルターに掛けることはできません。これを行うためには、`AccessLogExcludeURL` を使用してください。

## 形式

`AccessLogExcludeMimeType MIME_type [...]`

## 例

```
AccessLogExcludeMimeType image/gif
AccessLogExcludeMimeType text/html
AccessLogExcludeMimeType image/gif text/html
```

## デフォルト

なし。アクセス・ログには、すべての MIME タイプのファイルとディレクトリーに対する要求 (サーバー用) が含まれています。

## AccessLogExcludeReturnCode - 指定の戻りコードのログ項目を抑制する

このディレクティブは、指定の範囲のエラー・コード番号に入るアクセス要求はログに記録したくないことを指定する場合に使用します。これらのエラー・コード番号は、プロキシー・サーバー状況コードです。個々のコードを指定することはできません。300 を指定すると、リダイレクト戻りコード (301、302、303、および 304) を持つアクセス要求を除外したいということが示されます。

このディレクティブは、構成ファイル内に複数回指定することができます。また、1 つまたは複数のスペースで区切れば、1 つのディレクティブに複数の戻りコードを指定することもできます。

### 形式

`AccessLogExcludeReturnCode range`

### 例

`AccessLogExcludeReturnCode 300`

### デフォルト

なし。アクセス・ログには、コードとは関係なく、サーバーへ送信するすべての要求が含まれています。

## AccessLogExcludeURL - 特定のファイルまたはディレクトリーのログ項目を抑制する

このディレクティブは、指定の URL テンプレートに一致する特定のファイルまたはディレクトリーに対するアクセスの要求をログに記録したくないことを指定するときに使用します。例えば、GIF ファイルへのアクセス要求はログに記録したくない場合や、サーバー上の特定のファイルまたはディレクトリーに対するアクセス要求はログに記録したくない場合があります。

このディレクティブは、構成ファイル内に複数回指定することができます。また、1 つまたは複数のスペースで区切れば、1 つのディレクティブに複数の項目を指定することもできます。

### 形式

`AccessLogExcludeURL file_or_type [...]`

### 例

```
AccessLogExcludeURL *.gif
AccessLogExcludeURL /Freebies/*
AccessLogExcludeURL *.gif /Freebies/*
```

### デフォルト

なし。サーバーは、すべてのファイルおよびディレクトリーに対するアクセスの要求をログに記録します。

## AccessLogExcludeUserAgent - 特定のブラウザーからのログ項目を抑制する

このディレクティブは、特定のユーザー・エージェント（例えば、Internet Explorer 5.0）が行ったアクセス要求をログに記録しないことを指定する場合に使用します。

このディレクティブは、構成ファイル内に複数回指定することができます。また、1 つまたは複数のスペースで区切れば、1 つのディレクティブに複数の項目を指定することもできます。

### 形式

```
AccessLogExcludeUserAgent user_agent [...]
```

### 例

```
AccessLogExcludeUserAgent *Mozilla/2.0
AccessLogExcludeUserAgent *MSIE 5*
```

### デフォルト

ibmproxy.conf ファイルは、AccessLogExcludeUserAgent ディレクティブに対する次の定義をデフォルトで含みます。

```
AccessLogExcludeUserAgent IBM_Network_Dispatcher_HTTP_Advisor
AccessLogExcludeUserAgent IBM_Network_Dispatcher_WTE_Advisor
```

上記のユーザー・エージェントは、通常、Caching Proxy サーバーの前面に配置された、特定の Load Balancer アドバイザー用に定義されています。これらのユーザー・エージェントは、ログへの書き込み回数を最少化してパフォーマンスを向上させるため、ログに記録されません。デフォルトでは、サーバーは、他のすべてのユーザー・エージェントによるアクセス要求をログに記録します。

## AddBlankIcon - ディレクトリー・リストの見出しの位置合わせに使用するアイコンの URL を指定する

このディレクティブは、サーバーが FTP 要求のためのプロキシとして働く場合に、戻されたディレクトリー・リストの見出しの位置合わせに使用するアイコンを指定するときに使用します。アイコンは関連ファイルのそばに表示され、ファイルを区別するのに役立ちます。

このアイコンは、ブランク・アイコンとするか、あるいはディレクトリー・リストの見出しに表示されるように指定する別のアイコンとすることができます。正しい位置合わせのためには、使用するアイコンのサイズは、ディレクトリー・リスト上で使用する他のアイコンと同じでなければなりません。

### 形式

```
AddBlankIcon icon_URL alternative_text
icon_URL
```

アイコンの URL の最後の部分を指定します。サーバーはこの値を /icons/ ディレクトリーに付加して、完全な URL 要求を形成します。ローカル・ファイルに対する要求の場合は、サーバーは、マッピング・ディレクティブによって要求を

変換します。アイコンが検索されるためには、マッピング・ディレクティブで要求が渡されるようにしなければなりません。

サーバーをプロキシ・サーバーとして使用する場合は、完全要求は、サーバーを指す完全修飾 URL でなければなりません。

#### *alternative\_text*

要求側ブラウザがグラフィックスを表示しない場合に、アイコンの代りに使用される代替テキストを指定します。

### 例

```
AddBlankIcon logo.gif logo
```

### デフォルト

- **Linux** および **UNIX:** AddBlankIcon blank.m.pm.gif
- **Windows:** AddBlankIcon blank.gif

デフォルトでは、アイコンがブランクになっているので、代替テキストは指定されていません。

## AddDirIcon - ディレクトリー・リスト上のディレクトリーを示すアイコン URL を指定する

このディレクティブを使用して、ディレクトリー・リスト上のディレクトリーを表すアイコンを指定します。

### 形式

```
AddDirIcon icon_URL alternative_text
```

#### *icon\_URL*

アイコンの URL の最後の部分を指定します。サーバーはこの値を /icons/ ディレクトリーに付加して、完全な URL 要求を形成します。ローカル・ファイルに対する要求の場合は、サーバーは、マッピング・ディレクティブによって要求を変換します。アイコンが検索されるためには、マッピング・ディレクティブで要求が渡されるようにしなければなりません。

サーバーをプロキシ・サーバーとして使用する場合は、完全要求は、サーバーを指す完全修飾 URL でなければなりません。URL をローカル・ファイルにマップし、マッピング・ディレクティブによって、URL が渡されるようにしなければなりません。

#### *alternative\_text*

要求側ブラウザがグラフィックスを表示しない場合に、アイコンの代りに使用される代替テキストを指定します。

### 例

```
AddDirIcon direct.gif DIR
```

### デフォルト

- **Linux** および **UNIX:** AddDirIcon dir.m.pm.gif DIR
- **Windows:** AddDirIcon dir.gif DIR



## AddEncoding - 特定の接尾部を持つファイルの MIME コンテンツ・エンコードを指定する

このディレクティブは、特定の接尾部を持つファイルを MIME エンコード・タイプにバインドする場合に使用します。このディレクティブはあまり使用されません。

### 形式

AddEncoding *.extension encoding*

*.extension*

ファイル接尾部のパターンを指定します。

*encoding*

対応する接尾部パターンと一致するファイルにバインドされる MIME エンコード・タイプを指定します。

### 例

AddEncoding .qp quoted\_printable

### デフォルト

AddEncoding .Z x-compress

## AddIcon - アイコンを MIME コンテンツ・タイプまたはエンコード・タイプにバインドする

このディレクティブを使用して、特定の MIME コンテンツ・タイプまたはエンコード・タイプを持つファイルを表すアイコンを指定します。サーバーは、このアイコンを FTP ディレクトリー・リストを含むディレクトリー・リストで使用します。

### 形式

AddIcon *icon\_URL alternative\_text MIME\_type\_template*

*icon\_URL*

アイコンの URL の最後の部分を指定します。サーバーはこの値を /icons/ ディレクトリーに付加して、完全な URL 要求を形成します。ローカル・ファイルに対する要求の場合は、サーバーは、マッピング・ディレクティブによって要求を変換します。アイコンが検索されるためには、マッピング・ディレクティブで要求が渡されるようにしなければなりません。

サーバーをプロキシ・サーバーとして使用する場合は、完全要求は、サーバーを指す完全修飾 URL でなければなりません。URL をローカル・ファイルにマップし、マッピング・ディレクティブによって、URL が渡されるようにしなければなりません。

*alternative\_text*

要求側ブラウザがグラフィックスを表示しない場合に、アイコンの代りに使用される代替テキストを指定します。

*type\_template*

MIME コンテンツ・タイプまたはエンコード・タイプのいずれかのテンプレート



トを指定します。コンテンツ・タイプのテンプレートには、常にスラッシュ (/) が含まれます。エンコード・タイプのテンプレートには、スラッシュは含まれません。

## 例

```
AddIcon    video_file.m.pm.gif    MOV    video/*
```

## デフォルト

ibmproxy.conf 構成ファイルの AddIcon ディレクティブには、多数のデフォルトが設定されています。

## AddParentIcon - ディレクトリー・リスト上で親ディレクトリーを表すアイコンの URL を指定する

このディレクティブを使用して、ディレクトリー・リスト上の親ディレクトリーを表すアイコンを指定します。

## 形式

```
AddParentIcon    icon_URL    alternative_text
```

*icon-URL*

アイコンの URL の最後の部分を指定します。サーバーはこの値を /icons/ ディレクトリーに付加して、完全な URL 要求を形成します。ローカル・ファイルに対する要求の場合は、サーバーは、マッピング・ディレクティブによって要求を変換します。アイコンが検索されるためには、マッピング・ディレクティブで要求が渡されるようにしなければなりません。

サーバーをプロキシ・サーバーとして使用する場合は、完全要求は、サーバーを指す完全修飾 URL でなければなりません。URL をローカル・ファイルにマップし、マッピング・ディレクティブによって、URL が渡されるようにしなければなりません。

*alternative\_text*

要求側ブラウザがグラフィックスを表示しない場合に、アイコンの代りに使用される代替テキストを指定します。

## 例

```
AddParentIcon    parent.gif    UP
```

## デフォルト

```
AddParentIcon    dir-up.gif    UP
```

## AddType - 特定の接尾部を持つファイルのデータ・タイプを指定する

このディレクティブを使用して、特定の接尾部を持つファイルを MIME タイプおよびサブタイプにバインドします。このディレクティブは、構成ファイル内に複数回指定することができます。サーバーは、最も一般的に使用される接尾部のためのデフォルトを提供します。

## 形式

AddType *.extension type/subtype encoding [quality[ character\_set]]*

### *.extension*

ファイル接尾部のパターン。ワイルドカード文字 (\*) は、以下の 2 つの特別な接尾部パターンにおいてのみ使用することができます。

- \*.\*** ドット文字 (.) を含み、他のルールによって突き合わされないすべてのファイル名を突き合わせます。
- \*** ドット文字 (.) を含まず、他のルールによって突き合わされないすべてのファイル名を突き合わせます。

### *type/subtype*

対応する接尾部パターンと一致するファイルにバインドされる MIME タイプ/サブタイプ。

### *encoding*

データがどの形式に変換されたかを示す MIME コンテンツ・エンコード形式。FTP プロキシ・サーバーも、ファイルをバイナリー方式で検索すべきかどうかを判断するためにエンコードを使用します。多くの場合、使用されるエンコードは、7bit、8bit、または binary であり、次のように判別されます。

- 7bit** データはすべて、8859-1 ASCII データの短い行 (1000 文字未満) で表されます。通常、ソース・コードまたはプレーン・テキスト・ファイルがこのカテゴリーに含まれます。ただし、線画文字またはアクセント付き文字を含むファイルは例外です。
- 8bit** データは短い行で表されますが、高ビット・セット付きの文字 (例えば、線画文字またはアクセント付き文字) を含めることができます。ヨーロッパのサイトからの PostScript ファイルおよびテキスト・ファイルは、通常、このカテゴリーに含まれます。
- binary** このエンコードは、すべてのデータ・タイプに使用できます。データには、非 ASCII 文字だけでなく長い行 (1001 文字以上) が含まれます。application/\* タイプの 2 進データ・ファイルと同様、image/\*、audio/\*、および video/\* タイプのほとんどすべてのファイルは、このカテゴリーに含まれます。

その他のエンコード値はすべて binary と同様に扱われ、コンテンツ・エンコード MIME ヘッダーとして MIME ヘッダーに入れて渡されます。7bit および 8bit は、MIME ヘッダーに入れて送信されることはありません。

### *quality*

コンテンツ・タイプの相対値 (0.0 から 1.0 までの範囲) を示すオプションの標識を指定します。品質値は、1 つのファイルの複数の表示が要求に一致する場合に使用されます。サーバーは、最も高い品質値に関連したファイルを選択します。例えば、internet.ps というファイルが要求されていて、サーバーには以下のような AddType ディレクティブがあった場合は、サーバーは品質数値がより高いという理由で application/postscript 行を使用します。

```
AddType .ps application/postscript 8bit 1.0
AddType *.* application/binary binary 0.3
```

### *character\_set*

テキスト・ファイルに関連付けられる文字セットを示すオプションの標識。文字

セットを割り当てる先のファイルについては、ファイルを表示するときに使用する文字セットをサーバーがクライアント・ブラウザに通知します。

*character\_set* フィールドに値を設定した場合には、*quality* フィールドにも値を含めなければなりません。

## 例

```
AddType .bin application/octet-stream binary 0.8
```

## デフォルト

構成ファイル (*ibmproxy.conf*) には、*AddType* ディレクティブの多数のデフォルト設定が含まれています。

## AddUnknownIcon - ディレクトリー・リスト上の不明ファイル・タイプのアイコン URL を指定する

このディレクティブを使用して、ディレクトリー・リスト上のファイル・タイプ不明のファイルを表すアイコンを指定します。

## 形式

```
AddUnknownIcon icon_URL alternative_text
```

*icon\_URL*

アイコンの URL の最後の部分を指定します。サーバーは、この値を */icons/* に付加して、完全な URL 要求を形成します。ローカル・ファイルに対する要求の場合は、サーバーは、マッピング・ディレクティブによって要求を変換します。アイコンが検索されるためには、マッピング・ディレクティブで要求が渡されるようにしなければなりません。

サーバーをプロキシ・サーバーとして使用する場合は、完全要求は、サーバーを指す完全修飾 URL でなければなりません。URL をローカル・ファイルにマップし、マッピング・ディレクティブによって、URL が渡されるようにしなければなりません。

*alternative\_text*

要求側ブラウザがグラフィックスを表示しない場合に、アイコンの代りに使用される代替テキストを指定します。

## 例

```
AddUnknownIcon saywhat.gif unknown
```

## デフォルト

- **Linux** および **UNIX:** *AddUnknownIcon* *unknown.gif* ???
- **Windows:** *AddUnknownIcon* *unknown.gif* ???

## AdminPort - 管理ページまたはフォームを要求するためのポートを指定する

このディレクティブを使用して、管理者がサーバーの状況ページまたは構成フォームにアクセスする場合に使用するポートを指定します。このポートに対する要求は、*Port* ディレクティブで定義されている標準ポート上の、他のすべての着信要求

とともにキューには入りません。ただし、AdminPort 上の要求は、通常のアクセス制御および Pass、Exec、Protect などの要求マッピング規則を介して実行します。

注: 管理ポートは、Port ディレクティブで定義されている標準ポートと同一ではありません。

## 形式

AdminPort *port\_number*

## 例

AdminPort 2001

## デフォルト

AdminPort 8008

## AggressiveCaching - キャッシュ不可能ファイルのキャッシュを指定する

このディレクティブは、起点サーバーによって戻され、キャッシュ不可能のマークのあるファイルをキャッシュに入れる必要があるかどうかを指定するときに使用します。このディレクティブに従ってキャッシュに入れられたキャッシュ不可能なファイルには、再妥当性検査が必要である旨のマークが付けられます。このファイルが要求されるたびに、プロキシ・サーバーは、応答がキャッシュから提供される前にその応答の妥当性を再検査するために、If-Modified-Since (その後変更されたかどうか) の要求を起点サーバーに送信します。現在のところ、"cache-control: no-cache" ヘッダーを含む起点サーバーからの応答は、このディレクティブの影響を受けるキャッシュ不可能なファイルだけです。このディレクティブは、複数回指定することができます。

## 形式

AggressiveCaching *url\_pattern*

## 例

AggressiveCaching http://www.hosta.com/\*  
AggressiveCaching http://www.hostb.com/\*

逆方向の互換性のために、このディレクティブの前の構文 (AggressiveCaching {on | off}) は、以下のように取り扱われることになりました。

AggressiveCaching on は AggressiveCaching \* として取り扱われます。

AggressiveCaching off は無視されます。

注: AggressiveCaching off と AggressiveCaching *url\_pattern* の両方を指定した場合は、AggressiveCaching off は無視されて、警告メッセージが表示されます。

## デフォルト

なし

## AlwaysWelcome - 要求されたディレクトリーからウェルカム・ファイルを検索するかどうかを指定する

ディレクトリー名を含むが、ファイル名は含まない要求の場合、AlwaysWelcome ディレクティブは、サーバーがディレクトリーで戻すべきウェルカム・ファイルを探すかどうかを指定します。デフォルトにより、AlwaysWelcome の値は on に設定されます。つまり、サーバーは必ず要求されたディレクトリーを見て、Welcome ディレクティブに指定された名前に一致するファイルを探すということです。一致するファイルが見つかったら、そのファイルが要求側に戻されます。サーバーが、ディレクトリー内のファイルと Welcome ディレクティブに指定されたファイル名との間で一致するものを複数見つけた場合は、Welcome ディレクティブの順序によって、どのファイルが戻されるかが決まります。サーバーは、構成ファイルの一番上に最も近い Welcome ディレクティブを使用します。

### 形式

AlwaysWelcome on | off

### デフォルト

AlwaysWelcome on

### 関連ディレクティブ

- 313 ページの『Welcome - ウェルカム・ファイルの名前を指定する』

## appendCRLFtoPost - POST 要求に CRLF を付加する

このディレクティブは、Caching Proxy が POST 要求の本文の最後に復帰文字と改行文字を付加する必要がある URL を指定するときに使用します。このディレクティブは、複数回指定することができます。

注: このディレクティブは、POST 要求の処理で既知の問題がある URL の場合にのみ指定してください。

### 形式

appendCRLFtoPost *url\_pattern*

### 例

appendCRLFtoPost http://www.hosta.com/

### デフォルト

なし

## ArrayName - リモート・キャッシュ配列を指定する

このディレクティブは、複数のサーバーに共用されるリモート・キャッシュ配列を指定するときに使用します。

注: 配列をセットアップするときには、その配列のすべてのメンバーで同じ Hostname ディレクティブを構成してください。

## 形式

ArrayName *array\_name*

## デフォルト

なし

## Authentication - 認証ステップをカスタマイズする

このディレクティブを使用して、サーバー要求処理の認証ステップ実行中にサーバーで呼び出したいカスタマイズ済みアプリケーション関数を指定します。このコードは、認証方式に従って実行されます。BASIC 認証だけがサポートされています。

注: 認証は認可プロセスの一部で、認可が要求された場合にだけ行われます。

## 形式

Authentication *type* */path/file:function\_name*

*type*

アプリケーション関数が呼び出されるかどうかをさらに判別する認証方式を指定します。アスタリスク (\*) と BASIC の両方の値を使用することができます。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でアプリケーション関数に付けた名前を指定します。

## 例

Authentication BASIC /ics/api/bin/icsextpgm.so:basic\_authentication

## デフォルト

なし

## Authorization - 許可ステップをカスタマイズする

このディレクティブは、サーバー要求プロセスの許可ステップの実行中にサーバーで呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、要求されたオブジェクトをクライアントに提供できるようにします。

## 形式

Authorization *request\_template* */path/file:function\_name*

*request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ文字 (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、

/front\_page.html、http://www.ics.raleigh.ibm.com、/pub\*、/\*、および \* はすべて有効です。Caching Proxy をリバース・プロクシーとして使用しているとき、要求テンプレートは文書のルート (/) から開始する必要があります。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でアプリケーション関数に付けた名前を指定します。

## 例

Authorization /index.html /api/bin/icsextpgm.so:auth\_url

## デフォルト

なし

## AutoCacheRefresh - キャッシュ・リフレッシュを使用するかどうかを指定する

このディレクティブは、キャッシュ・リフレッシュを On または Off に設定するとき 사용합니다。リフレッシュが On にされると、キャッシュのコンテンツが自動的にリフレッシュされます。リフレッシュが Off の場合には、キャッシュ・エージェントは呼び出されず、その設定はすべて無視されます。キャッシュ・エージェントを別の方法で (例えば、Linux または UNIX システムで **cron** ジョブを使用することによって) 開始している場合には、このディレクティブは Off に設定してください。

## 形式

AutoCacheRefresh {on | off}

## デフォルト

AutoCacheRefresh On

## BindSpecific - サーバーが 1 つまたはすべての IP アドレスのどちらにバインドするかを指定する

このディレクティブは、サーバーが単一のネットワーク・アドレスを listen するかどうかを指定するときに、マルチホーム・システム上で使用します。値を On に設定すると、サーバーは、すべてのローカル IP アドレスにバインドせずに、Hostname ディレクティブに指定された IP アドレスにバインドします。

このディレクティブが指定されていないと、サーバーはデフォルトの Hostname とバインドします。

このディレクティブを変更した場合は、手動でサーバーを停止してから再始動しなければなりません。再始動しただけでは、サーバーは変更を行いません。(15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。)

## 形式

BindSpecific {on | off} [OutgoingSrcIp ip\_addr | host\_name]

[OutgoingSrcIp ip\_addr | host\_name]

OutgoingSrcIp オプションを使用すると、Caching Proxy が発信接続を作成するときに特定の送信元 IP アドレスを使用することができます。DMZ での Caching Proxy 設定と、特殊なファイアウォール規則に応じる上で役立ちます。



## デフォルト

BindSpecific Off

## BlockSize - キャッシュ内のブロックのサイズを指定する

このディレクティブは、キャッシュ・デバイスのメディア内のブロックのサイズ (バイト単位) を指定します。デフォルトでは、その値は 8192 です。これがサポートされる唯一のサイズであるため、値は変更しないようにしてください。詳しくは、181 ページの『htcformat コマンド』の解説セクションを参照してください。

### 形式

BlockSize *size*

### デフォルト

デフォルトでは、構成ファイルに BlockSize の設定はありません。(デフォルト値は 8192 です。)

## CacheAccessLog - キャッシュ・アクセス・ログ・ファイルのパスを指定する

このディレクティブは、プロキシ・キャッシュへのアクセスのログをサーバーに保管させる場所のパスとファイル名を指定するために使用します。このディレクティブは、サーバーがプロキシとして実行されている場合にのみ有効です。詳しくは、211 ページの『CacheRefreshTime - キャッシュ・エージェントをいつ開始するかを指定する』を参照してください。

プロキシ・キャッシュへの要求のログ記録を使用可能にするには、Caching ディレクティブを ON に設定し、CacheMemory および CacheAccessLog ディレクティブの値を設定する必要があります。オプションで、CacheDev ディレクティブを使用して 1 つまたは複数のキャッシュ・デバイスを定義できます。

CacheAccessLog の値は、絶対パスか ServerRoot への相対パスのいずれかとすることができます。(それぞれについて例を 1 つ示してあります。)

### 形式

CacheAccessLog *path/file*

### 例

```
CacheAccessLog /absolute/path/logfile
CacheAccessLog /logs/logfile
```

### デフォルト

- **Linux** および **UNIX** システム: CacheAccessLog  
/opt/ibm/edge/cp/server\_root/logs/cache
- **Windows** システム: CacheAccessLog *drive:¥Program*  
Files¥IBM¥edge¥cp¥logs¥cache



## CacheAlgorithm - キャッシュ・アルゴリズムを指定する

このディレクティブは、ガーベッジ・コレクション中にサーバーが使用するキャッシュ・アルゴリズムを指定するときに使用します。

### 形式

CacheAlgorithm {bandwidth | responsetime | blend}

#### bandwidth

ネットワーク帯域幅の節約を最大化する試み。

#### responsetime

ユーザーの応答時間を最小化する試み。

#### blend

bandwidth および responsetime のバランスのとれた組み合わせの使用。

### デフォルト

CacheAlgorithm bandwidth

## CacheByIncomingUrl - キャッシュ・ファイル名を生成する場合の基準を指定する

このディレクティブは、生成されるキャッシュ・ファイル名が要求の着信 URL を基にするかどうかを指定するときに使用します。

このディレクティブを On に設定すると、キャッシュ・ファイル名は着信 URL を基にして生成されます。このディレクティブを Off に設定すると、着信 URL は、最初にすべての適用可能な名前変換プラグイン、MAP 規則、および PROXY 規則経由で渡され、生成されるキャッシュ・ファイル名はその結果の URL に基づきます。

注: キャッシュ・フィルターを定義する際は、URL ベースのキャッシュ・フィルターのリバース・プロキシのシナリオでは、/ (スラッシュ) の文書ルートから始まるフォーマットを使用してください。例えば、/test/index.html です。このフォーマットにはプロトコルを含めないでください。例えば、http:// としないでください。

### 形式

CacheByIncomingUrl {on | off}

### デフォルト

CacheByIncomingURL off

## CacheClean - キャッシュされたファイルの保持期間を指定する

このディレクティブは、キャッシュされたファイルをサーバーが保持する期間を指定するために使用します。ガーベッジ・コレクションの実行の際に、サーバーは、この期間を過ぎたキャッシュされたファイルを、そのファイルの有効期限とは無関係に削除します。指定された時間より長くファイルをキャッシュするよう要求されると、サーバーは、ファイルを供給する前に、そのファイルが有効であることを確認するためにファイルの再検証を行います。

## 形式

CacheClean *time\_specification*

## 例

CacheClean 2 weeks

## デフォルト

CacheClean 1 month

## CacheDefaultExpiry - デフォルトのファイル有効期限時間を指定する

このディレクティブは、Expires または Last-Modified ヘッダーのいずれもサーバーにより提供されていないファイルのデフォルトの有効期限時間を設定するときに使用します。URL テンプレートを指定し、そのテンプレートと一致する URL を持つファイルの有効期限時間を指定します。このディレクティブは、構成ファイル内で複数回使用することができます。テンプレートごとに別々のディレクティブを組み込んでください。URL テンプレートにはプロトコルを指定しなければなりません。時間の値は、月 (months)、週 (weeks)、日 (days)、および時間 (hours) を任意に組み合わせて指定します。

## 形式

CacheDefaultExpiry *URL\_template expiration\_time*

## デフォルト

CacheDefaultExpiry ftp:\* 1 day  
CacheDefaultExpiry gopher:\* 2 days  
CacheDefaultExpiry http:\* 0 days

注: HTTP プロトコルのデフォルトの有効期限は 0 日です。多くのスクリプト・プログラムでは有効期限が指定されず、その出力が即時に有効期限切れとなるので、この値を保持されるようお勧めします。0 以外の値では、クライアントが古くなったコンテンツを表示する可能性があります。

## CacheDev - キャッシュ用のストレージを指定する

このディレクティブは、キャッシュ・ストレージを指定するときに使用します。ファイルまたはロー・ディスク区画のいずれかを指定できます。AIX プラットフォームでは、ロー論理ボリュームを指定できます。(メモリー・キャッシュを使用しない場合は、ロー・ディスク・キャッシュによって最良のパフォーマンスが得られます。)

キャッシュ・デバイスは、指定する前に準備する必要があることに注意してください。キャッシュ・デバイスの準備をするには、**htcformat** コマンドを使用してそれをフォーマットします。詳しくは、181 ページの『htcformat コマンド』を参照してください。

複数のキャッシュ・デバイスを指定できます。同じ CacheMemory 値と BlockSize 値に各デバイスが関連付けられます。しかし、プロキシ・サーバー・マシンで約 8 MB のメモリー・オーバーヘッドが、キャッシュ・デバイスごとに必要になります。

す。大きいデバイスを少数使用するほうが、小さいデバイスを数多く使用するよりも効率的です。 最高の効率を得るには、1 つのディスク全体を 1 つの大きい区画として使用し、そのディスクには他のものを何も入れないでください。キャッシュ・ストレージの詳細については、115 ページの『ディスク・キャッシュのパフォーマンスの最適化』を参照してください。

## 形式

CacheDev {*raw\_disk\_partition* | *file*}

## 例

**AIX:** CacheDev /dev/r1v02

**HP-UX:** CacheDev /dev/rdisk/clt15d0

**Linux:** CacheDev /opt/IBMWTE/filecache1

**Solaris:** CacheDev /dev/rdisk/clt3d0s0

**Windows:** CacheDev ¥¥.¥E:

## デフォルト

なし

## CacheExpiryCheck - サーバーが有効期限切れファイルに戻すかどうかを指定する

このディレクティブは、サーバーが有効期限の切れたキャッシュ・ファイルに戻すかどうかを指定するときに使用します。サーバーに有効期限切れのファイルに戻させたい場合には、この値を `off` に設定します。クライアントが有効期限切れのファイルを要求している場合に、プロキシがもっと最近のバージョンについて起点サーバーをチェックするようにしたい場合は、デフォルト値の `on` を使用します。一般に、管理者はサーバーが有効期限切れのファイルに戻すことを希望しません。ただし例外として、サーバーを実際に点検しているときなど、戻されるコンテンツについては特に関心がない場合があります。

## 形式

CacheExpiryCheck {*on* | *off*}

## デフォルト

CacheExpiryCheck `on`

## CacheFileSizeLimit - キャッシュに入れるファイルの最大サイズを指定する

このディレクティブは、キャッシュに入れるファイルの最大サイズを指定するときに使用します。このサイズより大きいファイルはキャッシュに格納されません。その値は、バイト (B)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定することができます。この指定で数値と測定単位 (B、K、M、G) の間にスペースが入っていても問題ありません。

## 形式

CacheFileSizeLimit *maximum* {B | K | M | G}

## デフォルト

CacheFileSizeLimit 4000 K

## CacheLastModifiedFactor - 有効期限を決定する値を指定する

このディレクティブは、特定の URL、またはテンプレートと一致するすべての URL に対する有効期限の計算に使用する値を指定する場合に使用します。

HTTP サーバーでは、ファイルの「最終変更」日時が提供されることはよくありますが、「有効期限」の日付は提供されません。同様に、FTP ファイルには、「最終変更」タイム・スタンプはあっても、有効期限がない場合があります。Caching Proxy は、最終変更日時に基づいてこれらのファイルの有効期限を計算します。サーバーは、最終変更日時を使用して、ファイルが変更されてからの時間の長さを判別し、それに CacheLastModifiedFactor ディレクティブの値を乗算します。この計算結果は、ファイルの存続時間、またはファイルが失効するまでの期間です。

また、off または -1 を指定して、このディレクティブを Off にし、有効期限を計算しないようにすることもできます。プロキシー・サーバーは、CacheLastModifiedFactor ディレクティブを構成ファイル内に表示される順番で読み取ります。プロキシー・サーバーは、キャッシュ・ファイルに適用できる最初のディレクティブを使用します。

## 形式

CacheLastModifiedFactor *url factor*

*url*

キャッシュされているファイルの、プロトコルを含む完全な URL を指定します。ワイルドカードとしてのアスタリスク (\*) が付いている URL テンプレートを、マスクを適用するために使用することもできます。

*factor*

計算に使用される係数を指定します。off または -1 の値も指定できます。

## 例

```
CacheLastModifiedFactor *://hosta/*    off
CacheLastModifiedFactor ftp://hostb/*  0.30
CacheLastModifiedFactor ftp://*        0.25
CacheLastModifiedFactor http://*       0.10
CacheLastModifiedFactor *              0.50
```

## デフォルト

```
CacheLastModifiedFactor http://* 0.10
CacheLastModifiedFactor http://*.htm* 0.20
CacheLastModifiedFactor http://*.gif 1.00
CacheLastModifiedFactor http://*.jpg 1.00
CacheLastModifiedFactor http://*.jpeg 1.00
CacheLastModifiedFactor http://*.png 1.00
CacheLastModifiedFactor http://*.tar 1.00
```

```
CacheLastModifiedFactor http://*.zip 1.00
CacheLastModifiedFactor http:* 0.15
CacheLastModifiedFactor ftp:* 0.50
CacheLastModifiedFactor * 0.10
```

デフォルトの 0.14 では、1 週間前に変更されたファイルが 1 日で有効期限切れになります。

## CacheLocalDomain - ローカル・ドメインをキャッシュに入れるかどうかを指定する

このディレクティブは、プロキシと同じドメイン内のホストからの URL をキャッシュに入れるかどうかを指定するために使用します。内部の帯域幅は URL を迅速にロードするのに十分であるため、イントラネット上のローカル・サイトでは、通常、キャッシュに入れる必要はありません。ローカル・サイトをキャッシュに入れないということは、検索に時間のかかる URL のためにキャッシュ・スペースを節約することになります。

### 形式

```
CacheLocalDomain {on | off}
```

### デフォルト

```
CacheLocalDomain on
```

## CacheMatchLanguage - 戻されるキャッシュ・コンテンツの言語プリファレンスを指定してください。

バックエンド・サーバーが同一の URL で多種の言語をお客様に戻す能力がある場合は、このディレクティブを使用して、同一の URL での異なる言語のキャッシングをサポートします。このディレクティブを使用すると、Caching Proxy が要求中の言語プリファレンスを、キャッシュされた応答の言語と比べて検証することができます。

CacheMatchLanguage が使用可能にされると、Caching Proxy がキャッシュされたコンテンツを読み込む前に、要求の Accept-Language ヘッダーにある言語プリファレンスを、キャッシュされたコンテンツの言語と比べます。Caching Proxy はまた、プリファレンスとの違いの大きさを比べます。プリファレンスとの違いの大きさが指定された限度内の場合は、キャッシュされたコピーを戻し、そうでない場合は、プロキシは要求をバックエンド・サーバーに転送し、要求した言語で新しいコピーを取得します。

### 形式

```
CacheMatchLanguage {on | off} lang-prefer-distance-limit special-id-for-all-lang
```

*lang-prefer-distance-limit*

0.001 から 0.9999 の範囲内の値を指定します。

*special-id-for-all-lang*

Content-Language ヘッダーでサーバーから戻される言語ストリングを指定します。この言語ストリングは、この応答がすべての言語プリファレンスで使用できることをプロキシに知らせます。

## 例

以下は、ディレクティブ、キャッシュ・オブジェクト、および要求の構成例です。

```
CacheMatchLanguage On 0.2
```

キャッシュ・オブジェクトが中国語 (簡体字、zh\_cn) であり、要求が次のものの場合、

```
GET / HTTP/1.1
```

```
...
```

```
Accept-Language: en_US;q=1.0, zh_cn;q=0.7, ja;q=0.3
```

```
....
```

この要求では、カスタマーは英語のページ (コードと品質は en\_US/1.0) を要求し、次に中国語 (簡体字) (コードと品質は zh\_cn/0.7) を要求し、次に日本語 (コードと品質は ja/0.3) を要求します。キャッシュされるオブジェクトは中国語 (簡体字) です。期待される最良の品質と一致する言語品質との間のプリファレンスの違いは、 $1.0 - 0.7 = 0.3$  です。CacheMatchLanguage ディレクティブで限度が 0.2 に設定されており、0.3 は限度を超えているため、プロキシはキャッシュ中のオブジェクトを戻すのではなく、サーバーにその URL の新規コピーを求めます。

サーバーが言語を指定していないか、応答を戻すときに Content-Language ヘッダーで special-id-for-all-lang を指定していない場合は、次の要求を受けるときにプロキシは言語プリファレンスの突き合わせをせず、キャッシュ中のコピーを戻します。

## デフォルト

```
CacheMatchLanguage off
```

## CacheMaxExpiry - キャッシュ・ファイルの最大存続時間を指定する

このディレクティブは、ファイルがキャッシュ内に留まっていられる時間の最大値を定義する場合に使用します。キャッシュ・ファイルの存続時間は、更新のために起点による検査を受けることなしに、それをキャッシュから提供できる時間の長さを定義します。場合によっては、キャッシュ・ファイルの計算された存続時間の方がユーザーがファイルを保持したい時間より長い場合があります。ファイルの存続時間 (起点によって指定されたかまたは Caching Proxy によって計算された) は、CacheMaxExpiry ディレクティブによって指定された限界を超えることはできません。

このディレクティブは、構成ファイル内で複数回使用することができます。テンプレートごとに別々のディレクティブを組み込んでください。

## 形式

```
CacheMaxExpiry URL lifetime
```

*URL*

キャッシュされているファイルの、プロトコルを含めて完全に指定される URL を指定します。ワイルドカードとしてのアスタリスク (\*) が付いている URL テンプレートを、マスクを適用するために使用することもできます。

*lifetime*

URL テンプレートと一致しているキャッシュ・ファイルの最大存続時間を指定



します。時間は、月 (months)、週 (weeks)、日 (days)、時間 (hours)、分 (minutes)、または秒 (seconds) を任意に組み合わせて指定することができます。

## 例

```
CacheMaxExpiry ftp:* 1 month
CacheMaxExpiry http://www.santaclaus.np/* 2 days 12 hours
```

## デフォルト

```
CacheMaxExpiry 1 month
```

## CacheMemory - キャッシュ RAM を指定する

このディレクティブは、キャッシュに関連付けるメモリーの量を指定するときに使用します。ディスク・キャッシュのパフォーマンスを最適にするためには、キャッシュ索引を含むキャッシュ・インフラストラクチャーのサポートには、キャッシュ・メモリーの値を最小 64 MB にすることをお勧めします。キャッシュ・サイズが増えると、キャッシュ索引が増加し、索引を保管するためにさらにキャッシュ・メモリーが必要になります。64 MB のキャッシュ・メモリー値は、キャッシュ・インフラストラクチャーのサポートを提供し、約 6.4 GB までのディスク・キャッシュ用のキャッシュ索引を保管するために十分な大きさです。もっと大きなディスク・キャッシュの場合、キャッシュ・メモリーは、キャッシュ・サイズの 1% にすべきです。

メモリー・キャッシングを使用している場合には、キャッシュそれ自体とキャッシュ索引に必要なメモリー量の両方を含めるよう、このディレクティブを設定してください。

このディレクティブの最大推奨値は 1600 MB です。この制限は、Caching Proxy が、32 ビット・アプリケーションとして最大 2 GB のメモリーを使用できることから決定されています。キャッシュに必要なメモリーの量に、ルーチン処理で使用するメモリーの量を加えた合計が 2 GB に近づくか、またはそれを超えると、Caching Proxy は正常に稼働しません。

その量は、以下の単位のいずれかで指定できます。バイト (B)、キロバイト (K)、メガバイト (M)、およびギガバイト (G)。

## 形式

```
CacheMemory amount {B | K | M | G}
```

## デフォルト

```
CacheMemory 64 M
```

## CacheMinHold - ファイルを使用可能に保つ期間を指定する

このディレクティブは、有効期限を上書きするファイルの URL を指定するときに使用します。一部のサイトでは、ファイルの存続時間の終了前に有効期限が切れるように設定されているので、サーバーはファイルをさらに頻繁に要求することが必要になります。CacheMinHold ディレクティブによって、有効期限切れのファイルは、それが再度要求されるまで、指定された時間の長さだけキャッシュに保持されます。このディレクティブは、複数回指定することができます。

注: 有効期限が上書きされると、キャッシュ内のファイルは廃止または旧式になる場合があります。

## 例

CacheMinHold http://www.cachebusters.com/\* 1 hour

## デフォルト

なし

## CacheNoConnect - スタンドアロン・キャッシュ・モードを指定する

このディレクティブは、プロキシ・サーバーがリモート・サーバーからファイルを検索するかどうかを指定するときに使用します。デフォルト値 (Off) では、サーバーはリモート・サーバーからファイルを検索できます。On の値は、サーバーをスタンドアロン・キャッシュ・モードで稼働するように設定します。これは、サーバーがそのキャッシュにすでに保管されているファイルしか戻すことができないことを意味します。通常は、サーバーがこのモードで稼働するときは、CacheExpiryCheck ディレクティブも Off に設定します。

サーバーをスタンドアロン・キャッシュ・モードで実行するのは、サーバーをデモのために使用する場合に便利です。デモに使用したいファイルがすべてキャッシュに保管されていることがわかっていれば、ネットワーク接続は不要です。

## 形式

CacheNoConnect {on | off}

## デフォルト

CacheNoConnect Off

## CacheOnly - テンプレートと一致する URL を持つファイルだけをキャッシュに入れる

このディレクティブは、指定したテンプレートと一致する URL を持つファイルだけをキャッシュに入れるよう指定するときに使用します。このディレクティブは、構成ファイル内で複数回使用することができます。テンプレートごとに別々のディレクティブを組み込んでください。URL テンプレートにはプロトコルを指定しなければなりません。このディレクティブに値を設定しなければ、NoCaching ディレクティブと一致しないすべての URL もキャッシュに入れることができます。CacheOnly と NoCaching のどちらのディレクティブも構成ファイルに組み込まない場合は、すべての URL をキャッシュに入れることができます。

## 形式

CacheOnly *url\_pattern*

## 例

CacheOnly http://realstuff/\*



## デフォルト

なし

## CacheQueries - 疑問符 (?) を含む URL へのキャッシュ応答を指定する

このディレクティブは、照会要求に対する応答をキャッシュに入れる URL を指定するときに使用します。 PUBLIC *url\_pattern* の値を使用すると、起点サーバーに `cache-control: public` ヘッダーが含まれ、その応答が別の方法でキャッシュ可能であれば、URL に疑問符の含まれる GET 要求に対する応答がキャッシュに入れます。 ALWAYS *url\_pattern* の値を指定すると、その応答が別の方法でキャッシュ可能である場合は、URL に疑問符を含む GET 要求に対する応答がキャッシュに入れます。

このディレクティブは、複数回指定することができます。

```
CacheQueries {ALWAYS | PUBLIC} url_pattern
```

### 例

```
CacheQueries ALWAYS http://www.hosta.com/*  
CacheQueries PUBLIC http://www.hostb.com/*
```

注: 逆方向の互換性のために、前の `CacheQueries {ALWAYS | PUBLIC | NEVER}` の構文は、以下のように扱われることになりました。

- `CacheQueries ALWAYS` および `CacheQueries PUBLIC` は、`CacheQueries ALWAYS *` および `CacheQueries PUBLIC *` として扱われます。
- `CacheQueries NEVER` は無視されます。
- `CacheQueries NEVER` と `CacheQueries url_pattern` の両方が指定されている場合には、`CacheQueries NEVER` は無視されますが、警告メッセージが出されます。

## デフォルト

なし

## CacheRefreshInterval - キャッシュ・オブジェクトの再妥当性検査の時間間隔を指定する

このディレクティブは、キャッシュ・ファイルが変更されているかどうかを判別するために起点サーバーが検査する時期を指定するときに使用します。

`CacheClean` ディレクティブは、このディレクティブと同じように見えますが、相違点があります。`CacheRefreshInterval` は、プロキシがファイルの妥当性を使用前に検査するよう指定するだけであるのに対して、`CacheClean` ディレクティブでは、指定した期間の後にファイルがキャッシュから除去されます。

### 形式

- 以下の形式は、URL パターンと一致するすべてのファイルのリフレッシュ間隔を指定します。

```
CacheRefreshInterval URL_pattern time_period
```

- 以下の形式は、URL パターンと一致しない すべてのファイルのリフレッシュ間隔を指定します。リフレッシュ間隔だけが指定されます。

`CacheRefreshInterval time_period`

## 例

`CacheRefreshInterval *.gif 8 hours`  
`CacheRefreshInterval 1 week`

## デフォルト

`CacheRefreshInterval 2 weeks`

## CacheRefreshTime - キャッシュ・エージェントをいつ開始するかを指定する

このディレクティブは、キャッシュ・エージェントを開始する時期を指定するときに使用します。キャッシュ・エージェントは、特定の時間に開始することができます。

## 形式

`CacheRefreshTime HH:MM`

## デフォルト

`CacheRefreshTime 03:00`

## CacheTimeMargin - ファイルをキャッシングする場合の最小存続時間を指定する

`CacheTimeMargin` ディレクティブは、ファイルをキャッシュに入れておくために必要なそのファイルの最小存続時間を指定するときに使用します。

`Caching Proxy` は、ファイルごとに有効期限を計算します。ほとんどないことですが、ファイルが有効期限切れとなる前にそのファイルに対する別の要求が受け取られた場合に、`Caching Proxy` は、ファイルをキャッシュに入れておくにはそのファイルの存続時間が短すぎると見なします。デフォルトでは、`Caching Proxy` は存続時間が 10 分より短いファイルはキャッシュに入れません。キャッシュがその最大容量に近くなければ、このディレクティブは初期値のままにしておきます。キャッシュがその容量近くまで埋め込まれた場合には、この最小存続時間の値を大きくすることを考慮してください。

## 形式

`CacheTimeMargin minimum_lifetime`

## デフォルト

`CacheTimeMargin 10 minutes`

注: このディレクティブを 4 時間以上に設定すると、キャッシュの効率が著しく落ちます。

## CacheUnused - 未使用キャッシュ・ファイルの保持期間を指定する

このディレクティブは、指定したテンプレートと一致する URL を持つ未使用のキャッシュ・ファイルをサーバーが保持する時間の最大長を指定するときに使用します。テンプレートと一致する URL を持つ未使用ファイルは、有効期限とは無関係に、指定の期間だけキャッシュに格納された後でサーバーにより削除されます。このディレクティブは、構成ファイル内で複数回使用することができます。テンプレートごとに別々のディレクティブを組み込んでください。URL テンプレートにはプロトコルを指定しなければなりません。時間の値は、月 (months)、週 (weeks)、日 (days)、および時間 (hours) を任意に組み合わせて指定します。

### 形式

```
CacheUnused url_template time_length
```

### 例

```
CacheUnused ftp:* 3 weeks
CacheUnused gopher:* 3 days 12 hours
CacheUnused * 4 weeks
```

### デフォルト

```
CacheUnused ftp:* 3 days
CacheUnused gopher:* 12 hours
CacheUnused http:* 2 days
```

## Caching - プロキシ・キャッシュを使用可能にする

このディレクティブは、ファイルのキャッシュを使用可能にするときに使用します。キャッシングが On になっていると、プロキシ・サーバーは、他のサーバーから検索したファイルをローカル・キャッシュに保管します。これで、プロキシ・サーバーは、同じファイルに対するこれ以降の要求があっても、他のサーバーから検索する必要なしに応答します。

### 形式

```
Caching {on | off}
```

### デフォルト

```
Caching On
```

注: Caching ディレクティブを変更した場合は、手動でサーバーを停止してから再始動しなければなりません。(15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。)

## CompressAge - ログをいつ圧縮するかを指定する

このディレクティブは、ログを圧縮するまでの経過時間を指定するときに使用します。ログは、CompressAge に設定された値より古くなると圧縮されます。

CompressAge を 0 に設定した場合は、ログが圧縮されることはありません。現在および直前の日のログは決して圧縮されません。

## 形式

CompressAge *number\_of\_days*

## デフォルト

CompressAge 1

## 関連ディレクティブ

- 214 ページの『CompressDeleteAge - ログをいつ削除するかを指定する』
- 『CompressCommand - 圧縮コマンドおよびパラメーターを指定する』
- 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 288 ページの『PurgeAge - ログの経過時間限度を指定する』
- 288 ページの『PurgeSize - ログ・アーカイブのサイズの限度を指定する』

## CompressCommand - 圧縮コマンドおよびパラメーターを指定する

このディレクティブは、ログの圧縮に使用される圧縮ユーティリティを識別し、パラメーターをそのユーティリティに渡すコマンドを作成するときに使用します。アーカイブ・ログのパスを組み込んでください。

圧縮ユーティリティは、そのマシンのパスにリストされているディレクトリーにインストールしなければなりません。

## 形式

CompressCommand *command*

*command*

単一の行に入力した、使用したいコマンドおよびパラメーターを含みます。典型的な例として、パラメーターは %%LOGFILES%% および %%DATE%% を含みます。

%%LOGFILES%%

特定の %%DATE%% に対して使用可能なログ・ファイルのリストを指定します。

%%DATE%%

ログ・ファイルの日付スタンプを指定します。

## 例

- **Linux および UNIX の場合:**

```
CompressCommand tar -cf /logarchs/log%%DATE%%.tar %%LOGFILES%% ;
gzip /logarchs/log%%DATE%%.tar
CompressCommand tar -cf /logarchs/log%%DATE%%.tar %%LOGFILES%% ;
compress /logarchs/log%%DATE%%.tar
CompressCommand zip -q /logarchs/log%%DATE%%.zip %%LOGFILES%%
```

**注:** コマンドおよびすべてのパラメーターは、一行に入力しなければなりません。上述の例では、最初の 2 つのコマンド例が読みやすくするために分割されています。

- **Windows の場合:**

```
CompressCommand pkzip -q d:\logarchs\log%%DATE%%.tar %%LOGFILES%%
```

## デフォルト

なし

## 関連ディレクティブ

- 212 ページの『CompressAge - ログをいつ圧縮するかを指定する』
- 『CompressDeleteAge - ログをいつ削除するかを指定する』
- 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 288 ページの『PurgeAge - ログの経過時間限度を指定する』
- 288 ページの『PurgeSize - ログ・アーカイブのサイズの限度を指定する』

## CompressDeleteAge - ログをいつ削除するかを指定する

このディレクティブは、ログの圧縮後、いつログを削除するかを指定する場合に使用します。ログは、CompressDeleteAge の値に設定された日数より古くなると削除されます。CompressDeleteAge を 0 に設定するか、あるいは CompressAge ディレクティブに設定された値より低い場合には、ログは削除されません。

注: この圧縮プラグインが、当日または前日のログを削除することは決してありません。

## 形式

```
CompressDeleteAge number_of_days
```

## デフォルト

```
CompressDeleteAge 7
```

## 関連ディレクティブ

- 212 ページの『CompressAge - ログをいつ圧縮するかを指定する』
- 213 ページの『CompressCommand - 圧縮コマンドおよびパラメーターを指定する』
- 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 288 ページの『PurgeAge - ログの経過時間限度を指定する』
- 288 ページの『PurgeSize - ログ・アーカイブのサイズの限度を指定する』

## CompressionFilterAddContentType - 圧縮したい HTTP 応答のコンテンツ・タイプを指定する

圧縮したい HTTP 応答のコンテンツ・タイプを指定するには、このディレクティブを使用します。

HTTP 応答を圧縮すると、ネットワーク負荷の軽減に役立ち、プロキシ・サーバーのパフォーマンスも改善されます。圧縮フィルター機能が有効になっている場合、ブラウザで HTTP 圧縮がサポートされていて、HTTP 応答が現在のところ圧縮されていなければ、Caching Proxy は HTTP 応答を圧縮し、圧縮したコンテンツをブラウザに戻します。

## 例

圧縮フィルター機能を有効にするには、次の 2 つのディレクティブを `ibmproxy.conf` ファイルに追加します。

- **HP-UX システムの場合:**

```
CompressionFilterEnable /opt/ibm/edge/cp/lib/mod_z.sl  
CompressionFilterAddContentType type-1[,type-n]
```

- **その他の UNIX® システムおよび Linux システムの場合:**

```
CompressionFilterEnable /opt/ibm/edge/cp/lib/mod_z.so  
CompressionFilterAddContentType type-1[,type-n]
```

- **Windows システムの場合:**

```
CompressionFilterEnable C:\ProgramFiles\IBM\edge\cp\Bin\mod_z.dll  
CompressionFilterAddContentType type-1[,type-n]
```

`CompressionFilterEnable` ディレクティブで参照された `mod_z` ライブラリーは、`zlib1.1.4` の動的バージョンです。

`type-n` 変数は、コンテンツ・タイプ・ヘッダーの有効な任意の値です。たとえば、`text/html` または `image/bmp` がこれに相当します。

注: 特定タイプの HTTP 応答、たとえば JPEG 画像やビデオ・ストリームなどは、すでにアプリケーションによって著しく圧縮されているため、この関数を使用した圧縮は行わないでください。

## デフォルト

なし

## CompressionFilterEnable - HTTP 応答を圧縮するための圧縮フィルターを有効にする

このディレクティブは、バックエンド・サーバーからの、またはプロキシ・サーバーのキャッシュからの、HTTP 応答を圧縮するための圧縮フィルターを有効にする場合に、使用します。

このディレクティブの使用法の例については、214 ページの

『`CompressionFilterAddContentType` - 圧縮したい HTTP 応答のコンテンツ・タイプを指定する』を参照してください。

## デフォルト

なし

## ConfigFile - 追加構成ファイルの名前を指定する

このディレクティブは、追加の構成ファイルの名前および場所を指定する場合に使用します。特定の構成ファイル内で見つかったディレクティブは、現在の構成ファイルの後で処理されます。

注: キャッシュ・エージェントがこのファイルを読み取れるように、追加の構成ファイルの許可がユーザー「nobody」に対して「Read」に設定されていることを確認してください。

### 例

- **Linux** および **UNIX:** ConfigFile /etc/rca.conf
- **Windows:** ConfigFile c:%WINNT%\rca.conf

### デフォルト

なし

## ConnThreads - 接続管理に使用する接続スレッドの数を指定

このディレクティブを使用して、接続の管理に使用する接続スレッドの数を定義します。

### 形式

ConnThreads *number*

### デフォルト

ConnThreads 5

### 関連ディレクティブ

- 256 ページの『MaxActiveThreads - アクティブ・スレッドの最大数を指定する』

## ContinueCaching - キャッシングに必要なファイルの大きさを指定する

クライアント接続が終了した場合にも、Caching Proxy がキャッシュ・ファイルを作成し終わるために、要求されたファイルのうちのどれだけの部分を転送しなければならないかを指定するとき、このディレクティブを使用します。この変数の有効な値は 0 から 100 までの範囲の整数です。

例えば、ContinueCaching 75 を指定すると、Caching Proxy がクライアント接続の終了を検出するまでにファイルの 75% 以上がすでに転送されていれば、Caching Proxy は、コンテンツ・サーバーからのファイルの転送を継続し、キャッシュ・ファイルを生成します。

### 形式

ContinueCaching *percentage*

### デフォルト

ContinueCaching 75



## DefinePicsRule - コンテンツのフィルター操作規則を提供する

このディレクティブは、レーティング・サービス情報を含めて、コンテンツの URL をフィルターに掛けるために必要な情報をプロキシに提供するときを使用します。このディレクティブは、複数回指定することができます。

### 形式

```
DefinePicsRule "filter_name" {
```

### デフォルト

```
DefinePicsRule "RSAC Example" {
```

## DefProt - テンプレートと一致する要求にデフォルトの保護セットアップを指定する

このディレクティブを使用して、デフォルトの保護セットアップを、テンプレートと一致する要求に関連付けます。

注: 保護が正しく機能するために、DefProt および Protect ディレクティブは、構成ファイル内の Pass または Exec ディレクティブの前になければなりません。

### 形式

```
DefProt request_template setup_name [FOR server_IP_address | host_name]
```

#### *request\_template*

デフォルトの保護セットアップに関連付けたい要求のためのテンプレートを指定します。サーバーは、受信したクライアント要求をテンプレートと比較し、一致する場合は保護セットアップに関連付けます。

要求がこのテンプレートと一致しても、後続の Protect ディレクティブのテンプレートと一致しなければ、この要求の保護は活動化されません。Protect ディレクティブを DefProt とともに使用方法の説明については、273 ページの『Protect - テンプレートと一致する要求の保護セットアップを活動化する』を参照してください。

#### *setup*

*request\_template* と一致する要求に関連付けたい、構成ファイル内に定義されている名前付き保護セットアップ。保護セットアップは、保護サブディレクティブで定義されます。このパラメーターは、次の 3 つのフォームのいずれでも構いません。

- 保護サブディレクティブが入っている個別のファイルを指定する全パス名とファイル名。
- すでに Protection ディレクティブで定義された名前と一致する保護セットアップ・ラベル名。Protection ディレクティブに、保護サブディレクティブが入っています。
- 実際の保護サブディレクティブ。サブディレクティブは、中括弧 ({} ) で囲むことが必要です。左の中括弧は、DefProt ディレクティブと同じ行の最後の文字でなければなりません。その後の行に、サブディレクティブを 1 行に 1 つずつ指定します。右の中括弧は、最後のサブディレクティブ行の後の行に

単独で置かなければなりません。中括弧で囲まれた中に、コメント行を置くことはできません。保護サブディレクティブの説明については、以下を参照してください。

- 279 ページの『AuthType - 認証タイプを指定する』
- 279 ページの『DeleteMask - ファイルを削除できるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『GetMask - ファイルを取得できるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『GroupFile - 関連グループ・ファイルの場所を指定する』
- 280 ページの『Mask - HTTP 要求を行うことができるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『PasswdFile - 関連するパスワード・ファイルの場所を指定する』
- 281 ページの『PostMask - ファイルを POST できるユーザー名、グループ、およびアドレスを指定する』
- 281 ページの『PutMask - ファイルを PUT できるユーザー名、グループ、およびアドレスを指定する』
- 281 ページの『ServerID - パスワード・ファイルに関連付けられる名前を指定する』

**[FOR Server\_IP\_address | host\_name]**

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、FOR 240.146.167.72) を指定するか、あるいはホスト名 (例えば、FOR hostA.bcd.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しないと、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

**注:**

1. このパラメーターは、*setup* パラメーターをパスおよびファイル名または保護セットアップ・ラベルの形式で指定した場合にのみ使用できます。*setup* パラメーターを中括弧で囲んだ実際の保護サブディレクティブの形式で指定した場合には、このパラメーターは使用できません。
2. このパラメーターを使用するには、FOR または他の任意の文字ストリング (ブランクは含まない) を *setup* パラメーターと *IP\_address* または *host\_name* の間に挿入する必要があります。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

**注:** ディレクティブは 1 行で入力しなければなりません。

## 例

- 以下の例では、保護サブディレクティブが入っている個々のファイルを識別しています。

```
DefProt /secret/* /server/protect/setup1.acc
```

- 以下の例では、ラベル名を使用して保護サブディレクティブを指しています。このラベル名は、Protection ディレクティブのラベル名と一致しなければなりません。Protection ディレクティブは、DefProt ディレクティブより前になければなりません。

```
DefProt /secret/* SECRET-PROT
```

- 以下の例には、DefProt ディレクティブの一部として保護サブディレクティブが含まれています。

```
DefProt {  
  AuthType Basic  
  ServerID restricted  
  PasswdFile /docs/etc/WWW/restrict.password  
  GroupFile /docs/etc/WWW/restrict.group  
  GetMask authors  
  PutMask authors  
}
```

- 以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーが /secret/ で始まる要求を受信すると、その要求が受け入れられたネットワーク接続の IP アドレスに基づいて、異なるデフォルト保護セットアップをその要求と関連付けます。0.67.106.79 に入ってくる要求の場合に、サーバーは、その要求を CustomerA-PROT というラベルの保護ディレクティブで定義されたデフォルトの保護と関連付けます。0.83.100.45 に入ってくる要求の場合、サーバーはその要求を CustomerB-PROT というラベルを持つ、Protection ディレクティブで定義されたデフォルトの保護と対応させます。

```
DefProt /secret/* CustomerA-PROT 0.67.106.79  
DefProt /secret/* CustomerB-PROT 0.83.100.45
```

- 以下の例では、オプションのホスト名パラメーターを使用しています。/secret/ で始まる要求を受信すると、サーバーは、URL のホスト名に基づいて、別のデフォルト保護セットアップをその要求に対応付けます。hostA に送信された要求に対して、サーバーは、CustomerA-PROT ラベルを付けて Protection ディレクティブで定義したデフォルト保護と、この要求を関連付けます。hostB に入ってくる要求の場合に、サーバーは、その要求を CustomerB-PROT というラベルの保護ディレクティブで定義されたデフォルトの保護と関連付けます。

```
DefProt /secret/* CustomerA-PROT hostA.bcd.com  
DefProt /secret/* CustomerB-PROT hostB.bcd.com
```

## デフォルト

なし

## DelayPeriod - 要求間の一時停止を指定する

このディレクティブは、キャッシュ・エージェントが宛先サーバーに要求を送信する間隔をあけるかどうかを指定するために使用します。要求間の遅延を指定すると、宛先サーバーの負荷だけでなく、プロキシ・マシンおよびユーザーのネットワーク・リンクの負荷も軽減されます。遅延を指定しない場合には、キャッシュ・

エージェントは最高速度で稼働します。低速のインターネット接続の場合には、ネットワークの最大使用を達成するために、遅延期間は指定しないことを考慮してください。

注: インターネットへの接続が 128 kbps より速い場合には、リフレッシュ中のサイトへのあまりに多くの要求の高速の送信を避けるために、DelayPeriod は On に設定してください。

### 形式

DelayPeriod {on | off}

### デフォルト

DelayPeriod On

## DelveAcrossHosts - ドメイン間のキャッシュへの格納を指定する

このディレクティブは、キャッシュ・エージェントがホスト間のハイパーテキスト・リンクをたどるかどうかを指定するために使用します。キャッシュに格納された URL に他のサーバーへのリンクが含まれている場合、サーバーは、そのリンクを無視することもたどることもできます。DelveInto ディレクティブが never に設定されている場合は、このディレクティブは適用されません。

### 形式

DelveAcrossHosts {on | off}

### デフォルト

DelveAcrossHosts Off

## DelveDepth - キャッシュへの格納中にリンクをどこまで追跡するかを指定する

このディレクティブは、キャッシュにロードするページの検索時にたどるリンク・レベルの数を指定するときに使用します。DelveInto ディレクティブが never に設定されている場合は、このディレクティブは適用されません。

### 形式

DelveDepth *number\_of\_levels*

### デフォルト

DelveDepth 2

## DelveInto - キャッシュ・エージェントがリンクをたどるかどうかを指定する

このディレクティブは、キャッシュ・エージェントがキャッシュに入れられた URL からのリンクをたどってページをロードするかどうかを指定するときに使用します。

### 形式

DelveInto {always | never | admin | topn}

**always**

キャッシュ・エージェントは、これより前にキャッシュに入れられたすべての URL からのリンクをたどります。

**never**

キャッシュ・エージェントは、URL にあるすべてのリンクを無視します。

**admin**

キャッシュ・エージェントは、LoadURL ディレクティブに指定された URL のリンクだけをたどります。

**topn**

キャッシュ・エージェントは、キャッシュ内で最も頻繁に検索されたファイルからのリンクだけをたどります。

**デフォルト**

DelveInto always

## DirBackgroundImage - 背景イメージをディレクトリー・リストに指定する

このディレクティブは、背景イメージをプロキシー・サーバーによって生成されたディレクトリー・リストに適用するのに使用します。ディレクトリー・リストは、プロキシー・サーバーが FTP サイトのブラウズに使用されたときに生成されます。

背景イメージの絶対パスを指定してください。イメージが別のサーバーにある場合は、背景イメージは完全な URL として指定しなければなりません。背景イメージが指定されていない場合は、プレーンな白い背景が使用されます。

**形式**

DirBackgroundImage */path/file*

**例**

DirBackgroundImage /images/corplogo.png

DirBackgroundimage http://www.somehost.com/graphics/embossed.gif

**デフォルト**

なし

## DirShowBytes - 小さなファイルのバイト・カウントをディレクトリー・リストに表示する

このディレクティブを使用して、1 KB より小さいファイルの正確なバイト・カウントをディレクトリー・リストに表示するかどうかを指定します。値が off の場合は、ディレクトリー・リストでは、サイズが 1 KB 以下のファイルはすべて、サイズが 1 KB と表示されます。

**形式**

DirShowBytes {on | off}

## デフォルト

DirShowBytes Off

## DirShowCase - ディレクトリー・リスト上のファイルのソート時に大/小文字を区別する

このディレクティブを使用して、ディレクトリー・リスト上のファイル名のソート時に大文字と小文字を区別するかどうかを指定します。

On の値は、ファイルのリスト中で大文字が小文字の前に置かれることを意味します。

## 形式

DirShowCase {on | off}

## デフォルト

DirShowCase On

## DirShowDate - ディレクトリー・リストに最終変更日を表示する

このディレクティブは、ディレクトリー・リストに各ファイルが最後に変更された日付を含めるかどうかを指定するときに使用します。

## 形式

DirShowDate {on | off}

## デフォルト

DirShowDate On

## DirShowDescription - ファイルの記述をディレクトリー・リストに表示する

このディレクティブを使用して、HTML ファイルの記述をディレクトリー・リストに表示するかどうかを指定します。記述は、ファイルの HTML <title> タグから取得されます。

FTP ディレクトリー・リストの記述は、判別できる場合は MIME タイプを示しています。

## 形式

DirShowDescription {on | off}

## デフォルト

DirShowDescription On

## DirShowHidden - 隠しファイルをディレクトリー・リストに表示する

このディレクティブを使用して、ディレクトリー中の隠しファイルをディレクトリー・リストに表示するかどうかを指定します。サーバーは、ピリオド (.) で始まる名前を持つすべてのファイルを隠しファイルと見なします。

### 形式

DirShowHidden {on | off}

### デフォルト

DirShowHidden On

## DirShowIcons - アイコンをディレクトリー・リストに表示する

このディレクティブは、サーバーがアイコンをディレクトリー・リストに組み込むかどうかを指定するときに使用します。アイコンを使用すれば、リスト内のファイルのコンテンツ・タイプをグラフィックで表示することができます。アイコンそのものは、AddBlankIcon、AddDirIcon、AddIcon、AddParentIcon、および AddUnknownIcon ディレクティブで定義されます。

### 形式

DirShowIcons {on | off}

### デフォルト

DirShowIcons On

## DirShowMaxDescrLength - ディレクトリー・リストの記述の最大長を指定する

このディレクティブを使用して、ディレクトリー・リストの記述フィールドに表示される文字の最大文字数を設定します。

### 形式

DirShowMaxDescrLength *number\_of\_characters*

### デフォルト

DirShowMaxDescrLength 25

## DirShowMaxLength - ディレクトリー・リストに表示するファイル名の最大長を指定する

このディレクティブを使用して、ディレクトリー・リストのファイル名に使用される文字の最大文字数を設定します。

### 形式

DirShowMaxDescrLength *number\_of\_characters*

### デフォルト

DirShowMaxLength 25



## DirShowMinLength - ディレクトリー・リストに表示するファイル名の最小長を指定する

このディレクティブを使用して、ディレクトリー・リストのファイル名用に常に確保される最小の文字数を設定します。ディレクトリー内のファイル名は、この数字を超えても構いません。ただし、ファイル名は、DirShowMaxLength ディレクティブで指定した数字を超えてはなりません。

### 形式

DirShowMinLength *number\_of\_characters*

### デフォルト

DirShowMinLength 15

## DirShowSize - ディレクトリー・リストにファイル・サイズを表示する

このディレクティブを使用して、ファイルのサイズをディレクトリー・リストに表示するかどうかを指定します。

### 形式

DirShowSize {on | off}

### デフォルト

DirShowSize On

## Disable - HTTP メソッドを使用不可にする

このディレクティブは、サーバーが受け入れない HTTP メソッドを指定するときに使用します。サーバーが拒否するメソッドごとに、別個の Disable ディレクティブを入力してください。

デフォルト構成ファイルでは、GET、HEAD、OPTIONS、POST、および TRACE メソッド が使用可能であり、サポートされているその他すべての HTTP メソッドは使用不可です。現在使用可能になっているメソッドを使用不可にするには、そのメソッドを Enable ディレクティブから削除し、Disable ディレクティブに追加します。

### 形式

Disable *method*

注: 「構成および管理」フォームは、POST メソッドを使用してサーバー構成を更新します。POST メソッドを使用不可にすると、「構成および管理」フォームを使用できなくなります。

### デフォルト

Disable PUT  
Disable DELETE  
Disable CONNECT

## DisInheritEnv - CGI プログラムによって継承放棄される環境変数を指定する

このディレクティブを使用して、どの環境変数を CGI プログラムに継承させないかを指定します (ただし、CGI 処理特有の CGI 環境変数は除きます)。

デフォルトでは、すべての環境変数が CGI プログラムによって継承されます。このディレクティブは、個々の環境変数を継承から除外するときに使用します。

### 形式

```
DisInheritEnv environment_variable
```

### 例

```
DisInheritEnv PATH
DisInheritEnv LANG
```

この例では、PATH と LANG を除くすべての環境変数が CGI プログラムによって継承されます。

### デフォルト

なし

## DNS-Lookup - クライアントのホスト名を検索するかどうかを指定する

このディレクティブは、サーバーが要求クライアントのホスト名を検索するかどうかを指定するときに使用します。

### 形式

```
DNS-Lookup {on | off}
```

使用する値は、サーバーの働き方に関する以下のものに影響を与えます。

- サーバーのパフォーマンス。デフォルト値 `off` を使用すると、サーバーは、ホスト名参照を実行するためのリソースを使用しないので、サーバーのパフォーマンスと応答時間が改善されます。
- サーバーがログ・ファイルを書き込むときにクライアントに関して記録する情報。
  - `off` — クライアントは IP アドレスによって識別されます。
  - `on` — クライアントはホスト名によって識別されます。
- 保護セットアップ、サーバー・グループ・ファイル、およびアクセス制御リスト (ACL) ファイル内のアドレス・テンプレートでホスト名が使用できるかどうか。
  - `off` — アドレス・テンプレートでホスト名は使用できません。IP アドレスを使用しなければなりません。
  - `on` — アドレス・テンプレートでホスト名を使用できます。IP アドレスは使用できません。

注: 使用している保護ルールでドメイン・ネームを使用するには、DNS-Lookup ディレクティブを `on` に設定しなければなりません。

## デフォルト

DNS-Lookup Off

## Enable - HTTP メソッドを使用可能にする

このディレクティブは、サーバーがどの HTTP メソッドを受け入れるかを指定するときに使用します。

必要に応じていくつでも HTTP メソッドを使用可能にすることができます。サーバーが受け入れるメソッドごとに、別個の Enable ディレクティブを入力してください。

### 形式

Enable *method*

特定の URL に Service ディレクティブがない場合は、Enable ディレクティブを使用して、任意の HTTP メソッドについてカスタマイズ済みプログラミングを行うことができます。このディレクティブに指定するプログラムは、そのメソッドの標準処理をオーバーライドします。

Enable *method* /path/fileDLL:function\_name

Enable CONNECT メソッドのフォーマットおよび使用可能なオプションについては、128 ページの『SSL トンネリングの構成』を参照してください。

## デフォルト

Enable GET  
Enable HEAD  
Enable POST  
Enable TRACE  
Enable OPTIONS

## EnableTcpNodelay - TCP NODELAY ソケット・オプションを使用可能にする

このディレクティブは、TCP NODELAY ソケット・オプションを使用可能にするために使用します。

EnableTcpNodelay ディレクティブは、小さな IP パケット (SSL ハンドシェークまたは短い HTTP 応答など) が Caching Proxy とクライアントの間に送信される場合にパフォーマンスを高めます。デフォルトでは、TCP NODELAY オプションはすべてのソケットで使用可能にされています。

### 形式

EnableTcpNodelay {All | HTTP | HTTPS | None}

## デフォルト

EnableTcpNodelay All

## Error - エラー・ステップをカスタマイズする

このディレクティブを使用して、エラー・ステップ実行中にサーバーで呼び出したいカスタマイズ済みアプリケーション関数を指定します。このコードは、エラーが起こった場合に実行され、カスタマイズ済みエラー・ルーチンを提供します。

### 形式

Error *request\_template* */path/file:function\_name*

*request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、`/front_page.html`、`http://www.ics.raleigh.ibm.com`、`/pub*`、`/*`、および `*` はすべて有効です。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

### 例

Error `/index.html /ics/api/bin/icsext05.so:error_rtns`

### デフォルト

なし

## ErrorLog - サーバー・エラーがログに記録される場所のファイルを指定する

このディレクティブを使用して、サーバーに内部エラーのログ記録に使用させたいファイルのパス名とファイル名を指定します。

**注:** サーバーのユーザー ID、グループ ID、あるいはログ・ディレクトリー・パスに対するデフォルトを変更する場合は、新規ディレクトリーを作成し、その許可および所有権を更新します。サーバーが情報をユーザー定義のログ・ディレクトリーに書き込むことができるようにするには、そのディレクトリーの許可を 755 として設定し、ユーザー定義のサーバー・ユーザー ID を所有者として設定します。例えば、サーバーのユーザー ID をデフォルトから `jdoue` に変更し、デフォルト・ログ・ディレクトリーを `server_root/account` に変更すると、`server_root/account` ディレクトリーの許可は 755 になり、`jdoue` によって所有されるはずです。

サーバーが稼働中であれば、毎日真夜中に新規ログ・ファイルを開始します。それ以外の場合には、サーバーは、その日におけるサーバーの最初の始動時に新規ログ・ファイルを開始します。ファイル作成時に、サーバーは、指定されたファイル名を使用し、日付接尾部を付加します。日付接尾部は、`Mmmddyyyy` という形式です。ここで、`Mmm` は月の最初の 3 文字を表し、`dd` は日を表し、また、`yyyy` は年を表します。

## 形式

ErrorLog */path/logs\_directory/file\_name*

## デフォルト

- **Linux** および **UNIX** システム: ErrorLog  
*/opt/ibm/edge/cp/server\_root/logs/error*
- **Windows** システム: ErrorLog *drive:¥Program Files¥IBM¥edge¥cp¥logs¥error*

## ErrorPage - 特定のエラー条件にカスタマイズされたメッセージを指定する

このディレクティブを使用して、サーバーに特定のエラー条件が起こったときに、要求側クライアントに送信するファイルの名前を指定します。エラー・キーワードとエラー・メッセージ・ファイルを関連付ける ErrorPage ディレクティブは、構成ファイル *ibmproxy.conf* により提供されます。

エラー・メッセージをカスタマイズする場合は、ErrorPage ディレクティブを変更してエラー・キーワードを異なるファイルと関連付けたり、または提供されているエラー・メッセージ・ファイルを変更したりすることができます。例えば、メッセージを変更して問題の原因に関するより多くの情報を含め、それを解決する可能な方法を示すことができます。内部ネットワークの場合は、ユーザーの連絡先となる担当者を示すことができます。

ErrorPage ディレクティブは、構成ファイルの任意の場所に入れることができます。エラーが起こると、このファイルは構成ファイルに定義されたマッピング規則に従って処理されます。このため、送信するファイルは、Fail、Map、NameTrans、Pass、Redirect、および Service の各ディレクティブによって定義されたマッピング規則を介して到達可能な場所になければなりません。少なくとも、サーバーがエラー・メッセージ・ファイルを渡すことができるようにする Pass ディレクティブが必要です。

## 形式

ErrorPage *keyword /path/filename.html*

*keyword*

エラー条件に関連したキーワードの 1 つを指定します。キーワードは、ファイル *ibmproxy.conf* の ErrorPage ディレクティブでリストされます。キーワードを変更することはできません。

*/path/filename.html*

これは、Web 上のクライアントによって表示される、エラー・ファイルの完全修飾 Web 名を指定します。デフォルトのエラー・メッセージ・ファイルは、*/HTML/errorpages/* にあります。

## 例

ErrorPage scriptstart */HTML/errorpages/scriptstart.htmls*

この例では、scriptstart 条件が生じると、サーバーは、*/HTML/errorpages/* ディレクトリーで検出される *scriptstart.htmls* ファイルをクライアントに送信します。

以下の HTML テキストは、このファイルに含まれることがあるテキストの例です。

```
<HTML>
<HEAD>
<TITLE>Message for SCRIPTSTART condition</TITLE>
</HEAD>
<BODY>
The CGI program could not be started.
<P>
<A HREF="mailto:admin@websvr.com">Notify the administrator</A>
of this problem.
</BODY>
</HTML>
```

サーバーの構成ファイル内の上記のパスと一致するディレクティブが  
PASS /\* /wwwhome/\* であれば、このメッセージ・ファイルの絶対パスは  
/wwwhome/HTML/errorpages/scriptstart.htmls となります。

## サーバーが戻すエラー・メッセージをカスタマイズする

エラー条件はそれぞれキーワードによって識別されます。どのエラー・メッセージをカスタマイズするかを決めるには、まず、Caching Proxy で提供されたエラー・メッセージ・ファイルを調べます。これは、/HTML/errorpages の中にあります。エラー・ページには、エラー番号、デフォルト・メッセージ、原因の説明、および該当するリカバリー・アクションが含まれています。

次に、エラー・メッセージを変更するには、以下のいずれかを実行してください。

- 既存の HTML または HTMLS ファイルを変更 (最初にバックアップ・コピーを作成する) か、あるいは所要のテキストで新規の HTML または HTMLS ファイルを作成します。HTML エディターまたは ASCII エディターを使用することができます。サーバー側インクルードを使用したい場合は、必ず HTMLS ファイルを使用しなければなりません。
- エラー・メッセージを異なる名前で (または異なるパスに) 作成した場合は、そのファイルを指すようにそのキーワードのための ErrorPage ディレクティブを変更してください。

## エラー条件、原因、およびデフォルト・メッセージ

すべてのキーワードおよびデフォルトのエラー・メッセージ・ファイルは、ファイル ibmproxy.conf の ErrorPage ディレクティブ・セクションにリストされています。エラー・メッセージ・ファイルには、エラー・メッセージ番号、キーワード、デフォルト・メッセージ、説明、およびユーザー応答 (アクション) が含まれています。

## デフォルト

ファイル ibmproxy.conf には、多数のデフォルトが組み込まれています。

ErrorPage ディレクティブをエラー条件に変更しないと、その条件に対するサーバーのデフォルトのエラー・ページが送信されます。

## EventLog - イベント・ログ・ファイルのパスを指定する

このディレクティブは、イベント・ログのパスとファイル名を指定するときに使用します。イベント・ログは、キャッシュ自体に関する通知メッセージを取り込みます。

注: サーバーのユーザー ID、グループ ID、あるいはログ・ディレクトリー・パスに対するデフォルトを変更する場合は、新規ディレクトリーを作成し、その許可および所有権を更新します。サーバーが情報をユーザー定義のログ・ディレクトリーに書き込むことができるようにするには、そのディレクトリーの許可を 755 として設定し、ユーザー定義のサーバー・ユーザー ID を所有者として設定します。例えば、サーバーのユーザー ID をデフォルトから jdoe に変更し、デフォルト・ログ・ディレクトリーを server\_root/account に変更すると、server\_root/account ディレクトリーの許可は 755 になり、jdoe によって所有されるはずですが。

サーバーが稼働中であれば、毎日真夜中に新規ログ・ファイルを開始します。それ以外の場合には、サーバーは、その日におけるサーバーの最初の始動時に新規ログ・ファイルを開始します。ファイル作成時に、サーバーは、指定されたファイル名を使用し、日付接尾部を付加します。日付接尾部は、*Mmmddyyyy* という形式です。ここで、*Mmm* は月の最初の 3 文字を表し、*dd* は日を表し、また、*yyyy* は年を表します。

### 形式

EventLog /path/logs\_directory/file\_name

### デフォルト

- **Linux および UNIX システム:** EventLog  
/opt/ibm/edge/cp/server\_root/logs/event
- **Windows システム:** EventLog drive:%Program Files%IBM%edge%cp%logs%event

## Exec - 一致する要求に対して CGI プログラムを実行する

このディレクティブを使用して、CGI プログラムの実行によって受け入れ、応答する要求のためのテンプレートを指定します。要求は、Exec ディレクティブのテンプレートに一致すると、後続のディレクティブの要求テンプレートとは比較されません。

### 形式

Exec request\_template program\_path [Server\_IP\_address | host\_name]

*request\_template*

サーバーが、CGI プログラムを実行することによって受け入れて応答する要求のためのテンプレート。

*request\_template* と *program\_path* の両方で、ワイルドカードとしてアスタリスク (\*) を使用する必要があります。*request\_template* のワイルドカードと一致する要求の一部は、CGI プログラムが入っているファイルの名前で始まっていないわけではありません。



要求には、PATH\_INFO 環境変数に入れて CGI プログラムに渡す追加データが含まれていることもあります。追加データは、要求にある CGI プログラム・ファイル名の後の最初のスラッシュ (/) の後に続きます。このデータは、CGI の指定に従って渡されます。

#### *program\_path*

要求を処理するためにサーバーが実行する CGI プログラムが入っているファイルへのパスを指定します。*program\_path* にもワイルドカードを含めなくてはなりません。このワイルドカードは、CGI プログラムが入っているファイルの名前に置き換えられます。

Exec ディレクティブは再帰的ディレクティブで、すべてのサブディレクトリーに適用されます。cgi-bin および admin-bin のそれぞれのディレクトリーごとに、別々の Exec ディレクティブを使用する必要はありません。

#### [Server\_IP\_address | host\_name]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) またはホスト名 (例えば、hostA.bcd.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

サーバー IP アドレスの指定にワイルドカード文字を使用することはできません。

### 例

以下の例において、サーバーが /idd/depts/plan/c92 という要求を受信すると、/depts/bin/plan.exe に入っている CGI プログラムを、c92 を入力としてそのプログラムに渡して実行します。

以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーが /cgi-bin/ で始まる要求を受信した場合には、その要求が入ってきたネットワーク接続の IP アドレスを基にして、別のディレクトリーからの要求にサービスします。130.146.167.72 に入ってくる要求では、サーバーは /CGI-BIN/customerA ディレクトリーを使用します。アドレス 0.83.100.45 の接続に入ってくる要求の場合には、サーバーは /CGI-BIN/customerB ディレクトリーを使用します。

```
Exec    /cgi-bin/*      /CGI-BIN/customerA/*  130.129.167.72
Exec    /cgi-bin/*      /CGI-BIN/customerB/*  0.83.100.45
```

以下の例では、オプションのホスト名パラメーターを使用しています。サーバーは、/cgi-bin で始まる要求を受信すると、URL 内のホスト名に基づいて、別のディレクトリーからの要求を処理します。hostA.bcd.com に送信された要求に対して、サーバーは /CGI-BIN/customerA ディレクトリーを使用します。hostB.bcd.com に送信された要求に対して、サーバーは /CGI-BIN/customerB ディレクトリーを使用します。

```
Exec    /cgi-bin/*      /CGI-BIN/customerA/*  hostA.bcd.com
Exec    /cgi-bin/*      /CGI-BIN/customerB/*  hostB.bcd.com
```

## デフォルト

- **Linux** および **UNIX** システム

```
Exec    /cgi-bin/*      /opt/ibm/edge/cp/server_root/cgi-bin/*
Exec    /admin-bin/*     /opt/ibm/edge/cp/server_root/admin-bin/*
```

- **Windows** システム

```
Exec    server_root/cgi-bin/*
Exec    server_root/admin-bin/*
Exec    server_root/DOCS/admin-bin/*
```

## ExportCacheImageTo - キャッシュ・メモリーをディスクにエクスポートする

このディレクティブを使用して、キャッシュのコンテンツをダンプ・ファイルにエクスポートします。再始動時にメモリー・キャッシュが破損したり、同じキャッシュを複数のプロキシに配置する場合などに役立つ機能です。

### 形式

`ExportCacheImageTo export_file_name`

### デフォルト

なし

## ExternalCacheManager - IBM WebSphere Application Server からの動的キャッシング用の Caching Proxy の構成

これは、リバース・プロキシ構成にのみ適用されます。

このディレクティブを使用して、動的リソースをキャッシングできる IBM WebSphere Application Server (Caching Proxy アダプター・モジュールで構成される) を認識するための Caching Proxy を構成します。Caching Proxy では、アプリケーション・サーバーの動的キャッシュにも保管されている JSP の結果のコピーが保管されます。Caching Proxy は、IBM WebSphere Application Server からそのグループ ID が ExternalCacheManager 項目と一致する内容だけをキャッシュします。

この機能を使用するために、Caching Proxy 構成ファイルに Service ディレクティブを追加する必要があることにも注意してください。アプリケーション・サーバーでも、追加の構成ステップが必要です。詳しくは、111 ページの『第 22 章 動的に生成されたコンテンツのキャッシング』を参照してください。

### 形式

`ExternalCacheManager External_Cache_Manager_ID Maximum_Expiry_Time`

*External\_Cache\_Manager\_ID*

プロキシにサービスする IBM WebSphere Application Server に割り当てられた ID。この ID は、Application Server の、dynacache.xml ファイルにある externalCacheGroup: group id 属性で設定された ID と一致しなければなりません。

### *Maximum\_Expiry\_Time*

外部キャッシュ・マネージャーのためにキャッシュされるリソースに対するデフォルトの有効期限時間。外部キャッシュ・マネージャーが指定された時間内にキャッシュされたリソースを無効にしないと、そのリソースは指定された時間に有効期限切れになります。この時間は分または秒で指定できます。

### 例

以下の項目は、www.xyz.com ドメイン内にある、そのリソースが 20 秒またはそれ以前に有効期限切れとなる外部キャッシュ・マネージャー (IBM WebSphere Application Server) を定義します。

```
ExternalCacheManager IBM-CP-XYZ-1 20 seconds
```

### デフォルト

なし

## Fail - 一致する要求を拒否する

このディレクティブは、サーバーが処理する必要のない要求のためのテンプレートを指定するときに使用します。要求は、Fail ディレクティブのテンプレートに一致すると、後続のディレクティブの要求テンプレートとは比較されません。

### 形式

```
Fail request_template [Server_IP_address | host_name]
```

#### *request\_template*

サーバーが拒否する必要がある要求のためのテンプレートを指定します。要求がこのテンプレートと一致すると、サーバーはエラー・メッセージを要求側に送信します。

テンプレートではアスタリスクをワイルドカードとして使用できます。スラッシュ (/) の直後の波形記号 (w) は明示的に一致しなければならず、このためにワイルドカードを使用することはできません。

#### [*Server\_IP\_address* | *host\_name*]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) またはホスト名 (例えば、hostA.bcd.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

## 例

以下の例で、サーバーは `/usr/local/private/` で始まるすべての要求を拒否します。

```
Fail /usr/local/private/*
```

以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーは、要求が IP アドレス 240.146.167.72 のネットワーク接続で入ってきている場合には、`/customerB/` で始まるすべての要求を拒否します。サーバーは、要求が IP アドレス 0.83.100.45 のネットワーク接続で入ってきている場合には、`/customerA/` で始まるすべての要求を拒否します。

```
Fail /customerB/* 240.146.167.72
Fail /customerA/* 0.83.100.45
```

以下の例では、オプションのホスト名パラメーターを使用しています。`hostA.bcd.com` に要求が出された場合は、サーバーは、`/customerB/` で始まるすべての要求を拒否します。`hostB.bcd.com` に要求が出された場合は、サーバーは、`/customerA/` で始まるすべての要求を拒否します。

```
Fail /customerB/* hostA.bcd.com
Fail /customerA/* hostB.bcd.com
```

## デフォルト

なし

## FIPSEnable - SSLV3 および TLS 用の Federal Information Processing Standard (FIPS) 承認済み暗号を使用可能にする

このディレクティブを使用して、SSL 接続における SSLV3 および TLS プロトコルの FIPS 承認済み暗号を使用可能にします。このディレクティブが使用可能になると、サポートされる SSLV3 用暗号仕様 (V3CipherSpecs ディレクティブ) のリストは無視されます。また、許可された TLS 暗号仕様は、352F0AFF09FE に設定され、SSLV3 暗号仕様は FFFE に設定されます。

## 形式

```
FIPSEnable {on | off}
```

## デフォルト

```
FIPSEnable off
```

## flexibleSocks - フレキシブルな SOCKS のインプリメントを使用可能にする

SOCKS 構成ファイルを使用して、確立する接続のタイプを決定するようプロキシに指示するとき、このディレクティブを使用します。

## 形式

```
flexibleSocks {on | off}
```

## デフォルト

```
flexibleSocks on
```

## FTPDDirInfo - ディレクトリーのウェルカム・メッセージまたは記述メッセージを生成する

このディレクティブは、FTP サーバーがディレクトリーのウェルカム・メッセージまたは記述メッセージを生成できるようにする場合に使用します。このメッセージは、オプションで FTP リストの一部として表示することができます。FTPDDirInfo ディレクティブを使用すると、メッセージを表示する場所が制御できるようになります。

### 形式

FTPDDirInfo {top | bottom | off}

#### top

ウェルカム・メッセージをページの一番上の、ディレクトリー内のファイルのリストの前に表示します。

#### bottom

ウェルカム・メッセージをページの一番下の、ディレクトリー内のファイルのリストの後に表示します。

#### off

ウェルカム・ページを表示しません。

### デフォルト

FTPDDirInfo top

## ftp\_proxy - FTP 要求のための別のプロキシ・サーバーを指定する

プロキシ・サーバーがプロキシ・チェーンの一部である場合は、このサーバーが FTP 要求のために接続する必要がある別のプロキシの名前を指定するために、このディレクティブを使用してください。末尾のスラッシュ文字 (/) を含めた完全な URL を指定しなければなりません。オプションのドメイン名またはテンプレートの使用については、263 ページの『no\_proxy - ドメインに直接接続するためのテンプレートを指定する』を参照してください。

これは、フォワード・プロキシ構成にのみ適用されます。

### 形式

ftp\_proxy *full\_URL* [*domain\_name\_or\_template*]

### 例

ftp\_proxy http:// outer.proxy.server/

### デフォルト

なし

## FTPUrlPath - FTP URL をどう解釈するかを指定する

このディレクティブは、FTP URL にあるパス情報が、ログイン・ユーザーの作業ディレクトリーと相対的と解釈するか、あるいはルート・ディレクトリーと相対的と解釈するかを指定するときに使用します。

## 形式

FTPUrlPath {relative | absolute}

FTPUrlPath ディレクティブを `absolute` に設定した場合は、ログイン・ユーザーの FTP 作業ディレクトリーを FTP URL パスに含める必要があります。FTPUrlPath `Relative` が指定されている場合は、ログイン・ユーザーの FTP 作業ディレクトリーは FTP URL から省略されていなければなりません。例えば、ログイン・ユーザーのための作業ディレクトリー `/export/home/user1` に含まれているファイル `test1.html` にアクセスするには、FTPUrlPath ディレクティブの設定によって異なりますが、以下のような URL パスが必要です。

- 設定が FTPUrlPath `absolute` であれば、必要な URL パスは `ftp://ftphost/export/home/user1/test1.html` です。
- 設定が FTPUrlPath `relative` であれば、必要な URL パスは `ftp://ftphost/test1.html` です。

## デフォルト

なし

## Gc - ガーベッジ・コレクションを指定する

このディレクティブは、ガーベッジ・コレクションが使用されるかどうかを指定するときに使用します。キャッシングが使用可能になっている場合に、サーバーは、ガーベッジ・コレクション・プロセスを使用して、キャッシュに格納してはならないファイルを削除します。ファイルは、有効期限および他の Proxy ディレクティブ値に基づいて削除されます。一般に、キャッシングが使用可能になっていれば、ガーベッジ・コレクションが使用されます。ガーベッジ・コレクションが使用されない場合は、プロキシー・キャッシュは効率的に使用されません。

## 形式

Gc {on | off}

## デフォルト

Gc On

## GCAdvisor - ガーベッジ・コレクション・プロセスをカスタマイズする

このディレクティブは、サーバーをガーベッジ・コレクションに使用したいカスタマイズ済みアプリケーションを指定するときに使用します。

## 形式

GCAdvisor */path/file:function\_name*

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。



## 例

GcAdvisor /api/bin/customadvise.so:gcadv

## GcHighWater - ガーベッジ・コレクションをいつ開始するかを指定する

このディレクティブは、ガーベッジ・コレクションのトリガーとなるために埋め込まれている必要がある全キャッシュ容量のパーセンテージを指定するときに使用します。このパーセンテージは、**最高水準点**と呼ばれます。最高水準点は、全キャッシュ容量に対するパーセンテージとして指定します。ガーベッジ・コレクションは、最低水準点に達するまで継続されます — この設定については、『GcLowWater - ガーベッジ・コレクションをいつ終了するかを指定する』を参照してください。最高水準点のパーセンテージは、50 と 95 間で設定できます。

### 形式

GcHighWater *percentage*

### デフォルト

GcHighWater 90

## GcLowWater - ガーベッジ・コレクションをいつ終了するかを指定する

このディレクティブは、ガーベッジ・コレクションの終了のトリガーとなる全キャッシュ容量のパーセンテージを指定するときに使用します。このパーセンテージは、**最低水準点**として知られています。最低水準点は、全キャッシュ容量に対するパーセンテージとして指定します。値は、最高水準点に設定した値より低い値に設定しなければなりません。最高水準点の設定については、『GcHighWater - ガーベッジ・コレクションをいつ開始するかを指定する』を参照してください。

### 形式

GcLowWater *percentage*

### デフォルト

GcLowWater 60

## gopher\_proxy - Gopher 要求のための別のプロキシー・サーバーを指定する

プロキシー・サーバーがプロキシー・チェーンの一部である場合は、このサーバーが Gopher 要求のために接続する必要がある別のプロキシーの名前を指定するために、このディレクティブを使用してください。末尾のスラッシュ (/) を含めた完全な URL を指定しなければなりません。オプションのドメイン名またはテンプレートの使用については、263 ページの『no\_proxy - ドメインに直接接続するためのテンプレートを指定する』を参照してください。

これは、フォワード・プロキシー構成にのみ適用されます。



## 形式

`gopher_proxy full_URL[domain_name_or_template]`

## 例

`gopher_proxy http://outer.proxy.server/`

## デフォルト

なし

## GroupId - グループ ID を指定する

このディレクティブは、サーバーがファイルにアクセスする前に変更する先のグループ名または番号を指定するときに使用します。

このディレクティブを変更した場合は、手動でサーバーを停止してから再始動しなければ、変更が有効になりません。サーバーを再始動しただけでは、変更は有効になりません。(15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。)

**注:** サーバーのユーザー ID、グループ ID、あるいはログ・ディレクトリー・パスに対するデフォルトを変更する場合は、新規ディレクトリーを作成し、その許可および所有権を更新します。サーバーが情報をユーザー定義のログ・ディレクトリーに書き込むことができるようにするには、そのディレクトリーの許可を 755 として設定し、ユーザー定義のサーバー・ユーザー ID を所有者として設定します。例えば、サーバーのユーザー ID をデフォルトから `jdoue` に変換し、デフォルト・ログ・ディレクトリーを `server_root/account` に変更すると、`server_root/account` ディレクトリーの許可は、755 になり、`jdoue` によって所有されます。

## 形式

`GroupId { group_name | group_number }`

## デフォルト

**AIX:** `GroupId nobody`

**HP-UX:** `GroupId other`

**Linux:**

**Red Hat:** `GroupId nobody`

**SUSE:** `GroupId nogroup`

**Solaris:** `GroupId nobody`

## HeaderServerName - HTTP ヘッダーに戻されるプロキシー・サーバーの名前を指定する

このディレクティブは、HTTP ヘッダーに戻されるプロキシー・サーバーの名前を指定するときに使用します。

## 形式

HeaderServerName *name*

## デフォルト

なし

## Hostname - サーバーの完全修飾ドメイン・ネームまたは IP アドレスを指定する

このディレクティブは、ファイル要求からクライアントに戻されるドメイン・ネームまたは IP アドレスを指定する場合に使用します。ドメイン・ネームを指定する場合、ドメイン・ネーム・サーバーは名前を IP アドレスに変換できなければなりません。IP アドレスを指定する場合、ドメイン・ネーム・サーバーは必要なく、またアクセスもされません。

注: 配列をセットアップするときには、その配列のすべてのメンバーで同じ

Hostname ディレクティブを構成しなければなりません。

## 形式

Hostname {*name* | *IP address*}

## デフォルト

デフォルトでは、このディレクティブは初期構成ファイルには指定されません。構成ファイルにこのディレクティブを指定しないと、値はデフォルトによってユーザーのドメイン・ネーム・サーバーに定義されたホスト名となります。

## http\_proxy - HTTP 要求のための別のプロキシ・サーバーを指定する

プロキシ・サーバーがプロキシ・チェーンの一部である場合は、このサーバーが HTTP 要求のために接続する必要がある別のプロキシの名前を指定するために、このディレクティブを使用してください。末尾のスラッシュ (/) を含めた完全な URL を指定しなければなりません。オプションのドメイン名またはテンプレートの使用については、263 ページの『no\_proxy - ドメインに直接接続するためのテンプレートを指定する』を参照してください。

## 形式

http\_proxy *full\_URL*[*domain\_name\_or\_template*]

## 例

http://outer.proxy.server/

## デフォルト

なし

## HTTPSCheckRoot - HTTPS 要求をフィルターに掛ける

このディレクティブは、Caching Proxy が URL のためのセキュアでないホーム・ページを検索し、その中でラベルを見つけようとするかどうかを指定するときに使用

します。ラベルが見つかり、それらはセキュア要求に適用されます。例えば、`https://www.ibm.com/` を要求した場合に、Caching Proxy は `http://www.ibm.com/` からラベルを検索し、見つかったラベルを使用して `https://www.ibm.com/` をフィルターに掛けます。

HTTPSCheckRoot を Off に設定した場合には、Caching Proxy はセキュアでないホーム・ページと、その中のラベルを検索しません。

## 形式

HTTPSCheckRoot {on | off}

## デフォルト

HTTPSCheckRoot on

## ICP\_Address - ICP 照会用の IP アドレスを指定する

このサブディレクティブは、ICP 照会の送信および受信に使用する IP アドレスを指定するために使用します。これは、`<MODULEBEGIN> ICP` ディレクティブと `<MODULEEND>` ディレクティブの間に入れる必要があります。

## 形式

ICP\_Address *IP\_address*

## デフォルト

デフォルトでは、このディレクティブは初期構成ファイルには指定されません。構成ファイルにこのディレクティブを指定しないと、デフォルトはすべてのインターフェースで ICP 照会を受け入れおよび送信する値になります。

## ICP\_MaxThreads - ICP 照会用の最大スレッド数を指定する

このサブディレクティブは、ICP 照会を listen するために作成されるスレッド数を指定するために使用します。これは、`<MODULEBEGIN> ICP` ディレクティブと `<MODULEEND>` ディレクティブの間に入れる必要があります。

注: Redhat Linux 6.2 以下では、プロセス当たりの作成できるスレッドの最大数が小さいので、この数を低くする必要があります。ICP で使用する大きなスレッド数を指定すると、要求へのサービスに使用できるスレッド数が制限されることがあります。

## 形式

ICP\_MaxThreads *number\_of\_threads*

## デフォルト

ICP\_MaxThreads 5

## Occupier - ICP クラスターのメンバーを指定する

プロキシ・サーバーが ICP クラスターの一部である場合に、このサブディレクティブは ICP ピアを指定するために使用します。これは、`<MODULEBEGIN> ICP` ディレクティブと `<MODULEEND>` ディレクティブの間に入れる必要があります。

ICP クラスターに新しいピアを追加する場合には、既存のすべてのピアの構成ファイルに ICP ピア情報を追加する必要があります。それぞれのピアに 1 行を使用してください。ピア・リストには、現行のホストも含めることができます。ICP の初期化時には、現行のホスト項目は無視されます。これにより、構成ファイルを編集して現行のホストを除去しなくても、他のピア・マシンにコピーできる単一の構成ファイルを持つことができます。

## 形式

```
ICP_Peer hostname http_port icp_port
```

### hostname

ピアの名前

### http\_port

ピアのプロキシ・ポート

### icp\_port

ピアの ICP サーバー・ポート

## 例

以下の行は、プロキシ・ポートが 80 で ICP ポートが 3128 のホスト abc.xcompany.com をピアとして追加します。

```
ICP_Peer abc.xcompany.com 80 3128
```

## デフォルト

なし

## ICP\_Port - ICP 照会用のポート番号を指定する

このサブディレクティブは、ICP サーバーが ICP 照会を listen するポート番号を指定するために使用します。これは、<MODULEBEGIN> ICP ディレクティブと <MODULEEND> ディレクティブの間に入れる必要があります。

## 形式

```
ICP_Port port_number
```

## デフォルト

```
ICP_Port 3128
```

## ICP\_Timeout - ICP 照会に対する最大待機時間を指定する

このサブディレクティブは、Caching Proxy が ICP 照会に対する応答を待機する最長時間を指定するために使用します。この時間はミリ秒で指定します。これは、<MODULEBEGIN> ICP ディレクティブと <MODULEEND> ディレクティブの間に入れる必要があります。

## 形式

```
ICP_Timeout timeout_in_milliseconds
```

## デフォルト

```
ICP_Timeout 2000
```

## IgnoreURL - リフレッシュしない URL を指定する

このディレクティブは、キャッシュ・エージェントがロードしない URL を指定するときに使用します。このディレクティブは、キャッシュ・エージェントが、キャッシュに格納されている URL からのリンクをたどってページをロードするときに便利です。IgnoreURL ディレクティブを複数回使用して、異なる URL または URL マスクを指定することができます。このディレクティブの値には、マスクに適用するワイルドカードとしてアスタリスク (\*) を含めることができます。

### 形式

IgnoreURL URL

### 例

IgnoreURL http://www.yahoo.com/  
IgnoreURL http://\*.ibm.com/\*

### デフォルト

IgnoreURL \*/cgi-bin/\*

## imbeds - サーバー側インクルード処理が使用されるかどうかを指定する

このディレクティブは、ファイル・システム、CGI プログラム、またはその両方から提供されたファイルについて、サーバー側インクルード処理を実行したいかどうかを指定するときに使用します。サーバー側インクルード処理は、コンテンツ・タイプが `ext/x-ssi-html` のファイルに対して行われます。オプションで、コンテンツ・タイプが `text/html` のファイルについてもサーバー側インクルード処理が実行されるように指定することができます。コンテンツ・タイプの詳細については、194 ページの『AddType - 特定の接尾部を持つファイルのデータ・タイプを指定する』を参照してください。

サーバー側インクルード処理を使用すれば、戻されているファイルに情報を動的に挿入することができます。このような情報には、日付、ファイルのサイズ、ファイルの最終変更日、CGI またはサーバー側インクルード環境変数、およびテキスト・ファイルを含めることができます。サーバー側インクルード処理は、送信元がローカルであるファイルについてのみ実行されます。Caching Proxy は、プロキシまたはキャッシュ・オブジェクトに対してはサーバー側インクルード処理を実行しません。

サーバー側インクルードを使用すると、サーバーは、特殊コマンドが使用されるつど、それらのコマンドをファイルの中から探し出します。このため、サーバーのパフォーマンスに影響が出てクライアントに対する応答時間が遅くなることがあります。

### 形式

imbeds {on | off | files | cgi | noexec} {SSIOnly | html}

**on** サーバー側インクルード処理は、ファイル・システムからのファイルおよび CGI プログラムからのファイルに対して行われます。

**off**

サーバー側インクルード処理は、どのファイルに対しても行われません。

**files**

サーバー側インクルード処理は、ファイル・システムからのファイルに対してのみ行われます。

**cgi**

サーバー側インクルード処理は、CGI プログラムから戻されたファイルに対してのみ行われます。

**noexec****SSIOnly**

サーバー側インクルード処理は、`text/x-ssi-html` のコンテンツ・タイプを持つファイルに対して行われます。

**html**

サーバー側インクルード処理は、`text/html` のコンテンツ・タイプと `text/x-ssi-html` のコンテンツ・タイプを持つファイルに対して行われます。

サーバーは、検索した各ファイルのコンテンツ・タイプと、処理した各 CGI プログラムの出力をチェックします。

通常、サーバー側インクルード処理は、`text/x-ssi/html` のコンテンツ・タイプを持つファイルに対してのみ行われます。ただし、`text/html` のコンテンツ・タイプを持つファイルをサーバー側インクルード処理するように指定することができます。

注: サーバーは、`html` として、`html`、`.html`、および `.htm` を処理します。これ以外は、`SSIOnly` として扱います。

各接尾部には、正しいコンテンツ・タイプで `AddType` ディレクティブを定義する必要があります。`.htm` や `.html` 以外の接尾部を使用する場合は、`AddType` ディレクティブが `text/x-ssi/html` のコンテンツ・タイプで定義されていることを確認してください。

## デフォルト

`imbeds on SSIOnly`

## ImportCacheImageFrom - ファイルからキャッシュ・メモリーをインポートする

このディレクティブを使用して、キャッシュのコンテンツをダンプ・ファイルからインポートします。再始動時にメモリー・キャッシュが破損したり、同じキャッシュを複数のプロキシに配置する場合などに役立つ機能です。

## 形式

`ImportCacheImageFrom import_file_name`

## デフォルト

なし

## InheritEnv - CGI プログラムによって継承される環境変数を指定する

このディレクティブを使用して、どの環境変数を CGI プログラムに継承させるかを指定します (ただし、CGI 処理特有の CGI 環境変数は除きます)。

InheritEnv ディレクティブを含めないと、すべての環境変数が CGI プログラムによって継承されます。InheritEnv ディレクティブを含めると、InheritEnv ディレクティブに指定された環境変数だけが、CGI 特有の環境変数と一緒に継承されます。このディレクティブを使用すると、継承した変数の値をオプションで初期化することができます。

### 形式

`InheritEnv environment_variable`

### 例

```
InheritEnv PATH
InheritEnv LANG=ENUS
```

この例では、PATH および LANG 環境変数だけが CGI プログラムによって継承され、LANG 環境変数は ENUS の値で初期化されます。

### デフォルト

なし。デフォルトでは、すべての環境変数が CGI プログラムによって継承されます。

## InputTimeout - 入力タイムアウトを指定する

このディレクティブを使用して、クライアントがサーバーに接続した後に要求を送信できる時間を設定します。まずクライアントはサーバーに接続し、次に要求を送信します。このディレクティブによって指定された時間内にクライアントが要求を送信しなければ、サーバーは接続をクローズします。時間の値は、時間 (hours)、分 (minutes または mins)、および秒 (seconds または secs) を任意に組み合わせて指定します。

### 形式

`InputTimeout time`

### 例

```
InputTimeout 3 mins 30 secs
```

### デフォルト

```
InputTimeout 2 minutes
```



## JunctionReplaceUrlPrefix - JunctionRewrite プラグインと併用時に、接頭部を挿入する代わりに URL を置き換える

このディレクティブは JunctionRewrite プラグインのデフォルトのアクションをオーバーライドし、プロキシが HTML ページ中の特定の URL リンクを訂正することを可能にしています。このディレクティブは JunctionRewrite ディレクティブと組み合わせて使用されます。

これは、リバース・プロキシ構成にのみ適用されます。

JunctionReplaceUrlPrefix ディレクティブは、JunctionRewrite プラグインが URL の先頭に接頭部を挿入するのではなく、`url_pattern_1` から `url_pattern_2` へと URL を置き換えるように指示します。

### 形式

```
JunctionReplaceUrlPrefix url_pattern_1 url_pattern_2
```

### 例

```
JunctionReplaceUrlPrefix /server1.internaldomain.com/* /server1/*
```

この例では、URL が `/server1.internaldomain.com/notes.nsf` であるとし、接頭部は `/server1` であるとし、URL を `/server1/server1.internaldomain.com/notes.nsf` に再書き込みするために接頭部を挿入するのではなく、JunctionRewrite プラグインは URL を `/server1/notes.nsf` に変更します。

### デフォルト

なし

## JunctionRewrite - URL 再書き込みを使用可能にする

このディレクティブにより、Caching Proxy 内のジャンクション再書き込みルーチンは、発信元サーバーからの応答を再書き込みして、ジャンクションが使用された場合に、サーバーの相対 URL が適切な発信元サーバーにマップされるようにします。

これは、リバース・プロキシ構成にのみ適用されます。

UseCookie オプションを使用しないで「**JunctionRewrite on**」を設定した場合は、ジャンクション再書き込みプラグインも使用可能にする必要があります。ジャンクションは、プロキシ・マッピング・ルールによって定義されます。

JunctionRewrite に関する追加情報については、49 ページの『JunctionRewrite に代わる UseCookie』および 50 ページの『JunctionRewrite 機能を拡張する Transmogrier プラグインのサンプル』を参照してください。

### 形式

```
JunctionRewrite {on | on UseCookie | off}
```

### デフォルト

```
JunctionRewrite off
```

## JunctionRewriteSetCookiePath - JunctionRewrite プラグインとの併用時に Set-Cookie ヘッダーのパス・オプションを再書き込みする

このディレクティブは、Cookie 名が一致する際に、プロキシーが Set-Cookie ヘッダーのパス・オプションを再書き込みすることを可能にします。応答がジャンクションを必要としており、ジャンクションの接頭部が定義されている場合は、各パスの先頭に接頭部が挿入されます。このディレクティブは JunctionRewrite プラグインと併せて使用でき、また RewriteSetCookieDomain ディレクティブと併せて使用できます。

これは、リバース・プロキシー構成にのみ適用されます。

### 形式

JunctionRewriteSetCookiePath *cookie-name1 cookie-name2...*

*cookie-name*

Set-Cookie ヘッダー中の Cookie 名です。

### デフォルト

なし

## JunctionSkipUrlPrefix - JunctionRewrite プラグインと併用時に接頭部を既に含んでいる URL の再書き込みをスキップする

このディレクティブは JunctionRewrite プラグインのデフォルトのアクションをオーバーライドし、プロキシーが URL パターンに既に一致している場合は URL の再書き込みをスキップさせます。 JunctionRewrite プラグインと併せて使用し、HTML ページ内の一部の URL リンクを訂正する方法を提供します。通常、このディレクティブは既に接頭部を含む URL をスキップするために使用されます。

これは、リバース・プロキシー構成にのみ適用されます。

### 形式

JunctionSkipUrlPrefix *url\_pattern*

### 例

JunctionSkipUrlPrefix /server1/\*

この例では、URL が /server1/notes.nsf であるとし、ジャンクション・接頭部は /server1/ であるとし、URL を /server1/server1/notes.nsf に再書き込みする代わりに、 JunctionRewrite プラグインは URL の再書き込みをスキップし、URL は /server1/notes.nsf のまま変更されません。

### デフォルト

なし

## KeepExpired - リソースがプロキシで更新済みである場合、期限切れのリソースのコピーを戻すように指定する

このディレクティブを使用して、キャッシュ・オブジェクトが再検証されている間、バックエンド・サーバーが要求であふれてしまうのを防ぐ助力をします。

キャッシュ・オブジェクトがバックエンド・サーバー上のコンテンツで再検証されている時、同じリソースに対する要求は、バックエンド・サーバーに代理要求されます。同じ要求のあふれが、バックエンド・サーバーをダウンさせる原因になることもあります。このディレクティブを使用可能にすると、このような状態が発生するのを防ぐ助けとすることが可能です。そのディレクティブが使用可能になると、リソースがプロキシで更新済みである場合、有効期限が切れた、または失効したリソースのコピーは戻されます。

### 形式

KeepExpired {on | off}

### デフォルト

KeepExpired off

## KeyRing - 鍵リング・データベースへのファイル・パスを指定する

このディレクティブは、サーバーが SSL 要求に使用する鍵リング・データベースへのファイル・パスを指定するときに使用します。鍵リング・ファイルは iKeyman 鍵管理ユーティリティを介して生成されます。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

### 形式

KeyRing *filename*

### 例

**Windows:** KeyRing c:¥Program Files¥IBM¥edge¥cp¥¥key.kdb

**Linux および UNIX:** KeyRing /etc/key.kdb

### デフォルト

なし

## KeyRingStash - 鍵リング・データベースのパスワード・ファイルへのファイル・パスを指定する

このディレクティブは、鍵リング・データベースのパスワード・ファイルへのファイル・パスを指定するときに使用します。パスワード・ファイルは、鍵リング・データベース・ファイルの構築時に、iKeyman 鍵管理ユーティリティを介して生成されます。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

## 形式

`KeyRingStash file_path`

## 例

**Windows:** `KeyRingStash c:¥Program Files¥IBM¥edge¥cp¥key.sth`

**Linux および UNIX:** `KeyRingStash /etc/key.sth`

## デフォルト

なし

## LimitRequestBody - PUT 要求または POST 要求の最大ボディ・サイズを指定する

このディレクティブは、PUT 要求または POST 要求の最大ボディ・サイズを制御するときに使用します。LimitRequest ディレクティブは、アタックからプロキシを保護するために使用されます。

その値は、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定することができます。

## 形式

`LimitRequestBody max_body_size {K | M | G}`

## デフォルト

`LimitRequestBody 10 M`

## LimitRequestFields - クライアント要求のヘッダーの最大数を指定する

このディレクティブを使用して、クライアント要求に送信できるヘッダーの最大数を指定します。LimitRequest ディレクティブは、アタックからプロキシを保護するために使用されます。

## 形式

`LimitRequestFields number_headers`

## デフォルト

`LimitRequestFields 32`

## LimitRequestFieldSize - 最大ヘッダー長および最大要求行を指定する

このディレクティブを使用して、要求行の最大長および各要求内のヘッダーの最大長を指定します。LimitRequest ディレクティブは、アタックからプロキシを保護するために使用されます。

その値は、バイト (B)、キロバイト (K) で指定することができます。

## 形式

`LimitRequestFieldSize max_hdr_length {B | K}`

## デフォルト

`LimitRequestFieldSize 4096 B`

## ListenBacklog - サーバーが持てる listen バックログ・クライアント接続の数を指定する

このディレクティブは、サーバーが接続拒否を示すメッセージをクライアントに送信する前に持つ `listen` バックログ・クライアント接続の数を指定するときに使用します。この数は、サーバーが数秒で処理できる要求の数によって異なります。これは、クライアントがタイムアウトとなり、接続を打ち切るまでにサーバーが処理できる数より高い値にしないようにしてください。

注: `ListenBacklog` 値が TCP/IP のサポートする `SOMAXCONN` 値より大きい場合には、その `SOMAXCONN` 値が代わりに使用されます。

## 形式

`ListenBacklog number_of_requests`

## デフォルト

`ListenBacklog 128`

## LoadInlineImages - 組み込みイメージのリフレッシュを制御する

このディレクティブは、キャッシュ・エージェントがインライン・イメージを検索するかどうかを指定するために使用します。 `LoadInlineImages` を `On` に設定すると、キャッシュに格納中のページに組み込まれているイメージもキャッシュに入れます。 `Off` に設定した場合には、組み込みイメージはキャッシュに入れられません。

## 形式

`LoadInlineImages {on | off}`

## デフォルト

`LoadInlineImages on`

## LoadTopCached - リフレッシュを実行する頻繁にアクセスされるページの数指定する

このディレクティブは、前夜のキャッシュ・アクセス・ログにアクセスし、要求された回数の最も多い URL をロードするようキャッシュ・エージェントに指示するときに使用します。

`LoadTopCached` ディレクティブに値を設定する場合には、`Caching` ディレクティブを `On` に設定し、`CacheAccessLog` ディレクティブの値を設定する必要があります。

## 形式

`LoadTopCached number_of_pages`

## デフォルト

LoadTopCached 100

## LoadURL - リフレッシュする URL を指定する

このディレクティブは、キャッシュ・エージェントがキャッシュにロードする URL を指定するために使用します。 構成ファイルには複数の LoadURL ディレクティブを組み込むことができますが、ワイルドカードは使用できません。

### 形式

LoadURL *url*

### 例

LoadURL http://www.ibm.com/

## デフォルト

なし

## Log - ログ・ステップをカスタマイズする

このディレクティブは、ログ・ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードによって、接続がクローズされた後のログ記録およびその他の処理が提供されます。

### 形式

Log *request\_template* */path/file:function\_name*

*request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。 この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、/front\_page.html、http://www.ics.raleigh.ibm.com、/pub\*、/\*、および \* はすべて有効です。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。 open、write、および close の各関数の名前を指定する必要があります。

### 例

Log /index.html /api/bin/icsextpgm.so:log\_url

## デフォルト

なし

## LogArchive - ログ・アーカイブの動作を指定する

このディレクティブは、アーカイブ・ルーチンの動作を指定するときに使用します。このディレクティブは、グローバル設定を持つすべてのログに影響します。これは、ログが圧縮されるか、パージされるか、あるいはログに何も実行されないかを指定します。

**Compress** を指定した場合は、**CompressAge** および **CompressDeleteAge** ディレクティブを使用して、ログがいつ圧縮または削除されるかを指定します。使用するコマンドとそのパラメーターの指定には、**CompressCommand** ディレクティブを使用します。

**Purge** を指定した場合は、**PurgeAge** および **PurgeSize** ディレクティブを使用して、ログがいつパージされるかを指定します。

### 形式

```
LogArchive {Compress | Purge | none}
```

#### Compress

アーカイブ・ルーチンではログが圧縮されることを指定します。

#### Purge

アーカイブ・ルーチンではログが消去されることを指定します。

#### none

アーカイブ・ルーチンは何も実行しないことを指定します。

### デフォルト

```
LogArchive Purge
```

### 関連ディレクティブ

- 212 ページの『**CompressAge** - ログをいつ圧縮するかを指定する』
- 214 ページの『**CompressDeleteAge** - ログをいつ削除するかを指定する』
- 213 ページの『**CompressCommand** - 圧縮コマンドおよびパラメーターを指定する』
- 260 ページの『**Midnight** - ログのアーカイブに使用される API プラグインを指定する』
- 288 ページの『**PurgeAge** - ログの経過時間限度を指定する』
- 288 ページの『**PurgeSize** - ログ・アーカイブのサイズの限度を指定する』

## LogFileFormat - アクセス・ログの形式を指定する

このディレクティブは、アクセス・ログ・ファイルの形式を指定するときに使用します。

### 形式

```
LogFileFormat {common | combined}
```

デフォルトでは、ログは、NCSA 共通ログ形式で表示されます。代わりに NCSA 結合ログ形式でログを表示するには、**combined** を指定します。結合形式では、参照



URL (Referring URL)、ユーザー・エージェント (User Agent)、および Cookie (要求の中にある場合) のフィールドが追加されます。

## デフォルト

LogFileFormat common

## LogToGUI (Windows only) - サーバー・ウィンドウにログ項目を表示する

**Windows** システム専用です。コマンド行からプロキシを実行するときには、このディレクティブを使用して、アクセス・ログに出力します。サーバーのパフォーマンスを最適化するために、このディレクティブはデフォルトで off (使用不可) に設定されています。

注: プロキシをサービスとして実行するときには、このディレクティブは無効になります。

## 形式

LogToGUI {on | off}

## デフォルト

LogToGUI off

## LogToSyslog - アクセス情報をシステム・ログに送信するかどうかを指定する (Linux と UNIX 専用)

**Linux** および **UNIX** システム専用です。このディレクティブは、サーバーが、アクセス要求とエラーをアクセスおよびエラー・ログ・ファイルだけでなく、システム・ログにも記録するかどうかを指定するときに使用します。

## 形式

LogToSyslog {on | off}

システム・ログ・ファイルがサーバー上に存在していなければ、そのファイルにエラー・ログ情報を書き込むように指定してはなりません。アクセス情報またはエラー情報、あるいはその両方をログに記録する選択をすることができます。

エラー情報のみをシステム・ログに送るには、/etc/syslog.conf ファイルに以下の行を追加してください。

```
user.err syslog_output_file_for_error_information
```

アクセス情報のみをシステム・ログに送るには、/etc/syslog.conf ファイルに以下の行を追加してください。

```
user.info syslog_info_file_for_access_information
```

エラー情報とアクセス情報の両方をシステム・ログに送るには、/etc/syslog.conf ファイルに上記の両方の行を追加してください。

*syslog\_output\_file* および *syslog\_info\_file* は、以下の形式で指定します。

- **AIX:** /var/adm/name\_of\_syslog\_file

- **HP-UX:** /var/adm/syslog/syslog.log
- **Linux:** /var/adm/messages
- **Solaris:** /var/adm/messages

システム・ログ・ファイルを作成した後に、以下のコマンドを使用してそれを再始動できます。

```
kill -HUP 'cat /etc/syslog.pid'
```

## デフォルト

LogToSyslog Off

## Map - ルールの突き合わせを行うため、要求パス・ストリングを使用して、マッチング要求を新規要求ストリングに変更する

このディレクティブを使用して、新しい要求ストリングに変更される要求のためのテンプレートを指定します。 サーバーは、要求を変更すると新しい要求ストリングを使用し、それを後続のディレクティブの要求テンプレートと比較します。

Map ディレクティブでは、着信要求パス・ストリングを使用して、ルールの突き合わせを行います。 254 ページの『MapQuery - ルールの突き合わせを行うため、要求パスおよび照会ストリングを使用して、マッチング要求を新規要求ストリングに変更する』も参照してください。

## 形式

```
Map request_template new_request [server_IP_address | host_name]
```

### *request\_template*

サーバーが変更し、新しい要求ストリングと他のテンプレートとの比較を続ける要求のためのテンプレートを指定します。

テンプレートではアスタリスク (\*) をワイルドカードとして使用できます。 スラッシュ (/) の直後の波形記号 (~) は明示的に一致しなければならず、このためにワイルドカードを使用することはできません。

### *new\_request*

サーバーが使用して、後続ディレクティブの要求テンプレートの比較を継続する、新規要求ストリングを指定します。 *new\_request* で指定するストリングには、*request\_template* に含まれていれば、ワイルドカードを含めることができます。要求の *request\_template* ワイルドカードと一致する部分が *new\_request* のワイルドカードの代わりに挿入されます。

### *[server\_IP\_address | host\_name]*

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) を指定するか、またはホスト名 (例えば、hostA.raleigh.ibm.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

## 例

- 以下の例において、サーバーは `/stuff/` で始まるすべての要求を取り入れ、その要求の `/stuff/` 部分を `/good/stuff/` に変更します。新規の要求ストリングには、元の要求の `/stuff/` より後の部分も組み込まれます。したがって、`/stuff/whatsup/` は `/good/stuff/whatsup/` に変更されます。サーバーは新規の要求ストリングを取り入れ、それと後続ディレクティブの要求テンプレートとの比較を継続します。

```
Map /stuff/* /good/stuff/*
```

- 以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーは、`/stuff/` で始まる要求を受信すると、要求が入ってきたネットワーク接続の IP アドレスを基にして、その要求を別の要求ストリングに変更します。240.146.167.72 に入ってきた要求の場合には、サーバーはその要求の `/stuff/` 部分を `/customerA/good/stuff/` に変更します。アドレスが 0.83.100.45 の接続に入ってきた要求の場合には、サーバーはその要求の `/stuff/` 部分を `/customerB/good/stuff/` に変更します。

```
Map /stuff/* /customerA/good/stuff/* 240.146.167.72
Map /stuff/* /customerB/good/stuff/* 0.83.100.45
```

- 以下の例では、オプションのホスト名パラメーターを使用しています。サーバーは、`/stuff/` で始まる要求を受信すると、URL のホスト名に基づいて、その要求を別の要求ストリングに変更します。hostA に送信された要求の場合、サーバーはその要求の `/stuff/` 部分を `/customerA/good/stuff/` に変更します。hostB に送信された要求の場合、サーバーはその要求の `/stuff/` 部分を `/customerB/good/stuff/` に変更します。

```
Map /stuff/* /customerA/good/stuff/* hostA.bcd.com
Map /stuff/* /customerB/good/stuff/* hostB.bcd.com
```

## デフォルト

なし

## MapQuery - ルールの突き合わせを行うため、要求パスおよび照会ストリングを使用して、マッチング要求を新規要求ストリングに変更する

このディレクティブを使用して、新しい要求ストリングに変更される要求のためのテンプレートを指定します。サーバーは、要求を変更すると新しい要求ストリングを使用し、それを後続のディレクティブの要求テンプレートと比較します。

このディレクティブの機能は、Map ルール（253 ページの『Map - ルールの突き合わせを行うため、要求パス・ストリングを使用して、マッチング要求を新規要求ストリングに変更する』）とほとんど同じです。ただし、照会ストリング付きの URL を処理するために、MapQuery は、パス・ストリングと照会ストリングの両

方を使用してルールของ突き合わせをします。着信 URL が MapQuery ルールで一致すると、残りのルールとの突き合わせには、変換済みの URL が使用されます。

また MapQuery は、照会ストリング付きの URL を、異なるパス・ストリングまたは異なる照会ストリングを持つ、別の URL に変換することもできます。ただし、他のすべてのマッピング・ディレクティブでは、要求パスのみが使用されるため、要求パスが一致したときは、変換済みの URL に変更済みの照会ストリングが付加されるだけになります。パターンを突き合わせるために、変更済みの照会ストリングが使用されることはありません。

## 形式

MapQuery *request\_template new\_request* [*server\_IP\_address* | *host\_name*]

### *request\_template*

サーバーが変更し、新しい要求ストリングと他のテンプレートとの比較を続ける要求のためのテンプレートを指定します。

テンプレートではアスタリスク (\*) をワイルドカードとして使用できます。スラッシュ (/) の直後の波形記号 (~) は明示的に一致しなければならず、このためにワイルドカードを使用することはできません。

### *new\_request*

サーバーが使用して、後続ディレクティブの要求テンプレートの比較を続ける、新規要求ストリングを指定します。*new\_request* で指定するストリングには、*request\_template* に含まれていれば、ワイルドカードを含めることができます。要求の *request\_template* ワイルドカードと一致する部分が *new\_request* のワイルドカードの代わりに挿入されます。

### [*server\_IP\_address* | *host\_name*]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) を指定するか、またはホスト名 (例えば、hostA.raleigh.ibm.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

## 例

着信 URL が以下であるとしします。

/getsomthing?type=1

また、MapQuery ルールは以下であるとしします。

MapQuery /getsomthing?type=\* /gettype/\*

変換済みの URL は /gettype/1 になり、これが次のルール・マッピングで使われます。

```
Proxy /gettype/* http://server/gettype/*
```

変換済みの URL は http://server/gettype/1 になります。

## デフォルト

なし

## MaxActiveThreads - アクティブ・スレッドの最大数を指定する

このディレクティブを使用して、同時にアクティブにしておきたいスレッドの最大数を設定します。最大数に達すると、サーバーは、別の要求が終了してスレッドが使用可能になるまで、新しい要求を保留します。一般に、マシンの能力が高いほど、このディレクティブに設定する値も高くなります。マシンが、メモリー・スワップなどのオーバーヘッド・タスクにあまりに長く時間がかかるようになった場合は、この値を小さくしてみてください。

### 形式

```
MaxActiveThreads number_of_threads
```

### デフォルト

```
MaxActiveThreads 100
```

## MaxContentLengthBuffer - 動的データのためのバッファのサイズを指定する

このディレクティブは、サーバーが生成した動的データのためのバッファのサイズを設定するときに使います。動的データは、CGI プログラム、サーバー側インクルード、および API プログラムからの出力です。

その値は、バイト (B)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定することができます。 数字と値 (B、K、M、G) の間にスペースが入っていても構いません。

### 形式

```
MaxContentLengthBuffer size
```

### デフォルト

```
MaxContentLengthBuffer 100 K
```

## MaxLogFileSize - 各ログ・ファイルの最大サイズを指定する

このディレクティブは、各ログ・ファイルの最大サイズを指定するときに使います。各ログ・ファイルは、このディレクティブで定義されたサイズを超えることはできません。ログ・ファイルが定義された最大サイズに達すると、現在のログ・ファイルがクローズされ、次のインクリメンタル整数値を付加した新規ログ・ファイルが、同じ名前で作成されます。

注:

1. Caching Proxy は 32 ビット・アプリケーションであるため、32 ビット機能によってそのログ・ファイルを開きます。この制約のため、2 GB 以上の MaxLogFileSize を指定しないでください。まだ要求を処理している最中に、Caching Proxy がログ・ファイルに書き込もうとして、ログ・ファイルのサイズが 2GB を超えると、Caching Proxy が停止する場合があります。
2. Linux および Unix プラットフォームでは、ログ・ファイルがあるディレクトリーの許可が、少なくとも ibmproxy デーモンが実行されるグループに対して書き込み許可を持っていないと、ログ・ファイルが作成されません。つまり、ibmproxy.conf ファイルにあるロギング・ディレクティブ用のログ・ファイルのロケーションは、少なくとも ibmproxy.conf ファイル内の GroupId ディレクティブで定義されたグループに対して、書き込み許可を持っている必要があります。これは、ログ・ファイルのデフォルトのロケーションが変更された、あるいはデフォルトの UserId ディレクティブまたは GroupId ディレクティブが ibmproxy.conf ファイルで変更された場合のみ問題となります。

MaxLogFileSize ディレクティブを設定するための推奨値は、最小で 10 M、ただし 200 M より小さい値です。実際のログ・ファイル・サイズは、ここで設定したサイズより若干大きくなります。この値を低く設定しすぎると、プロキシー・サーバーがログ・ファイルをクローズおよびオープンする頻度が高くなるため、プロキシーのパフォーマンスに好ましくない影響が生じます。一部のプラットフォームでは、この値を高く設定しすぎると、プロキシーが入出力バッファのために、より多くのメモリーを使用する原因になります。ログ・ファイル・サイズがより大きくなると、入出力バッファはオペレーティング・システムによって制御されるようになりますが、プロキシーがメモリー不足になったり、メモリー・リークのように見える原因となる可能性があります。

その最大サイズは、バイト (B)、キロバイト (K)、メガバイト (M)、およびギガバイトのいずれかの単位で指定できます。

## 形式

MaxLogFileSize *maximum* {B | K | M | G}

## デフォルト

MaxLogfileSize 128 M

## MaxPersistRequest - 持続接続で受信する要求の最大数を指定する

このディレクティブは、サーバーが持続接続で受け取る要求の最大数を指定するときに使用します。この数値を決定するときは、ページで使用するイメージの数を考慮に入れてください。イメージごとに別々の要求が必要です。

## 形式

MaxPersistRequest *number*

## デフォルト

MaxPersistRequest 5



## MaxQueueDepth - キューに入れる URL の最大数を指定する

このディレクティブは、キャッシュ・エージェントの未解決ページ検索要求のキューの最大の項目数を指定するために使用します。大量のメモリーを備えた大きいシステムの場合には、使用可能なすべてのメモリーを使用しないで、ページ検索要求のキューをより大きく定義することができます。

キャッシュに入れる URL のキューは、キャッシュ・エージェントが実行されるたびに最初に判別されます。キャッシュ・エージェントが他の URL へのハイパーテキスト・リンクをたどるよう指示してある場合は、これら他の URL はキャッシュのキュー項目数には入りません。MaxURLs ディレクティブに指定された値に達すると、キューにそれ以上の URL があっても、キャッシュ・エージェントは停止します。

### 形式

MaxQueueDepth *maximum\_depth*

### デフォルト

MaxQueueDepth 250

## MaxRuntime - キャッシュ・エージェントが実行する最大時間数を指定する

このディレクティブは、特定の実行中にキャッシュ・エージェントが URL を検索する最大時間数を指定するときに使用します。値が 0 の場合、完了するまでキャッシュ・エージェントが稼働することを意味します。

### 形式

MaxRuntime {0 | *maximum\_time*}

### 例

MaxRuntime 2 hours 10 minutes

### デフォルト

MaxRuntime 2 hours

## MaxSocketPerServer - サーバーのオープン・アイドル状態のソケットの最大数を指定する

このディレクティブを使用して、オープン・アイドル状態のソケットの最大数を設定し、1 つの起点サーバーで保持できるようにします。このディレクティブは、ServerConnPool ディレクティブを on に設定した場合にのみ使用してください。

### 形式

MaxSocketPerServer *num*

### 例

MaxSocketPerServer 10



## デフォルト

MaxSocketPerServer 5

## MaxUrls - リフレッシュする URL の最大数を指定する

このディレクティブは、特定の実行中にキャッシュ・エージェントが検索する URL の最大数を指定するときに使用します。値が 0 の場合は、制限がないことを意味します。自動モードのキャッシュ・エージェントを使用している場合は、LoadURL および LoadTopCached ディレクティブが MaxURLs より優先します。

### 形式

MaxURLs *maximum\_number*

## デフォルト

MaxURLs 2000

## Member - 配列のメンバーを指定する

このディレクティブは、リモート・キャッシュ・アクセスを使用しているサーバーが共有する配列のメンバーを指定するために使用します。

注: 配列をセットアップするときには、その配列のすべてのメンバーで同じ Hostname ディレクティブを構成してください。

### 形式

```
Member name {  
  subdirective  
  subdirective  
  .  
  .  
}
```

以下のサブディレクティブが組み込まれています。

#### RCAAddr

この必須サブディレクティブは、RCA 通信の IP アドレスかホスト名を識別します。

#### RCAPort

この必須サブディレクティブは、RCA 通信のポートを識別します。ポート番号は、1024 以上 65535 以下でなければなりません。

#### CacheSize {*n bytes* | *n Kbytes* | *n Mbytes* | *n Gbytes*}

この必須サブディレクティブは、このメンバーのキャッシュのサイズを識別するもので、正の値でなければなりません。

#### [Timeout *n milliseconds* | *n seconds* | *n hours* | *n days* | *n months* | *n years* | forever]

このメンバーに対する待ち時間を識別します。*n* は、正の整数でなければなりません。Timeout はオプションで、デフォルトは 1000 ミリ秒です。Timeout の値は、通常はミリ秒で設定されます。

#### [BindSpecific {on | off}]

セキュリティ測定を提供し、プライベート・サブネット上で通信できるようにします。BindSpecific はオプションで、デフォルトは On です。

#### [ReuseAddr {on | off}]

配列の再結合を高速化できます。これを On に設定すると、他の処理がポートをスチールできますが、未定義の動作の原因となることがあります。ReuseAddr はオプションで、デフォルトは Off です。

### 例

```
Member bittersweet.chocolate.ibm.com {
  RCAAddr      127.0.0.1
  RCAPort      6294
  CacheSize    25G
  Timeout      500 milliseconds
  BindSpecific On
  ReuseAddr    Off
}
```

### デフォルト

なし

## Midnight - ログのアーカイブに使用される API プラグインを指定する

このディレクティブは、真夜中にログのアーカイブを実行するアプリケーション・プラグインを指定するために使用します。このディレクティブは、インストールの実行中に初期化されます。このディレクティブが構成ファイルに含まれていない場合には、アーカイブは実行されません。

### 形式

Midnight */path/file:function\_name*

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

### デフォルト

- **Linux** および **UNIX**: Midnight /usr/lib/archive.so:begin
- **Windows**: Midnight C:\Program Files\IBM\edge\cp\bin\archive.dll:begin

## NameTrans - 名前変換ステップをカスタマイズする

このディレクティブは、名前変換中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、要求内の仮想パスをサーバー上の物理パスに変換して、URL を特定のオブジェクトにマッピングするための機構を提供します。

注: これは、ターミナル・マッピング規則ではありません。変換された URL は、引き続きターミナル・マッピング規則ディレクティブ (Exec、Fail、Map、Pass、Redirect、および Service) のいずれかと一致しなければなりません。

## 形式

NameTrans *request\_template* /path/file:function\_name  
[Server\_IP\_address | host\_name]

### *request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、

ば、/front\_page.html、http://www.ics.raleigh.ibm.com、/pub\*、/\*、および \* はすべて有効です。

### */path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

### *function\_name*

プログラム内でのアプリケーション関数名を指定します。

### [Server\_IP\_address | host\_name]

複数の IP アドレスまたは仮想ホストを使用している場合は、アプリケーション機能が、特定の IP アドレスに送信される要求についてのみ、または特定のホストに対してのみ呼び出されるかを判別します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

注: ディレクティブは 1 行に入力する必要があります。上記では、読みやすくするために 2 行にしてあります。

## 例

NameTrans /index.html /api/bin/icsextpgm.so:trans\_url

## デフォルト

なし

## NoBG - Caching Proxy プロセスをフォアグラウンドで実行する

Linux および UNIX プラットフォームでは、このディレクティブは、Caching Proxy サーバー・プロセスが自動的にバックグラウンドで実行されないようにするために使用します。デフォルトでは off に設定されるこのディレクティブは、次の形式となります。

NoBG [on | off]

注: **ibmproxy** コマンドに対する **-nobg** オプションは、Windows システムには無効です。

## 例

NoBG on

## デフォルト

NoBG off

## NoCaching - URL がテンプレートと一致したファイルはキャッシュに入れないことを指定する

このディレクティブは、指定したテンプレートと URL が一致したファイルはサーバーがキャッシュに入れないことを指定するときに使用します。このディレクティブは、構成ファイル内で複数回使用することができます。テンプレートごとに別々のディレクティブを組み込んでください。URL テンプレートにはプロトコルを指定しなければなりません。

CacheOnly ディレクティブまたは NoCaching ディレクティブのどちらも設定されていない場合は、任意の URL がキャッシュの対象になります。

### 形式

NoCaching *URL\_pattern*

### 例

NoCaching http://joke/\*

### デフォルト

なし

## NoLog - テンプレートと一致する特定のホストまたはドメインのログ項目を抑制する

このディレクティブは、特定のホストまたは指定したテンプレートと一致するドメインからのアクセス要求はログに記録しないことを指定するときに使用します。例えば、ローカル・ホストからのアクセス要求をログに記録しないようにすることができます。

このディレクティブは、構成ファイル内で複数回使用することができます。また、テンプレートを 1 つまたはそれ以上のスペースで区切ると、同一のディレクティブで複数のテンプレートを指定することもできます。テンプレートでは、ホスト名または IP 番号アドレスを使用することができます。

注: ホスト名テンプレートを使用するには、DNS-Lookup ディレクティブを on に設定する必要があります。DNS-Lookup ディレクティブが off に設定されている場合 (デフォルト) は、IP アドレス・テンプレートのみ使用できます。

### 形式

NoLog {*host\_name* | *IP\_address*} [...]

### 例

NoLog 128.0.\* \*.edu localhost.\*

### デフォルト

なし

## no\_proxy - ドメインに直接接続するためのテンプレートを指定する

プロキシ・チェーニングのためにディレクティブ `http_proxy`、`ftp_proxy`、または `gopher_proxy` を使用している場合には、このディレクティブを使用して、サーバーがプロキシ経由ではなく、直接接続するドメインを指定することができます。

値は、ドメイン・ネームまたはドメイン・ネーム・テンプレートのストリングとして指定します。ストリング内の各項目はコンマ (,) で区切ってください。ストリング内にはスペースを使用しないでください。

このディレクティブ上のテンプレートの入力は、他のディレクティブ上のテンプレートとは異なります。最も重要な点は、ワイルドカード文字 (\*) を使用できないことです。ドメイン・ネームの最後の部分だけを使用してテンプレートを指定できます。サーバーは、指定されたテンプレートと一致するストリングで終わるドメインに直接接続します。このディレクティブは、プロキシ・チェーニングにのみ適用され、SOCKS 構成ファイル内のダイレクト @/= 行と同等です。

### 形式

```
no_proxy domain_name_or_template[,...]
```

### 例

```
no_proxy www.someco.com,.raleigh.ibm.com,.some.host.org:8080
```

この例では、以下の要求の場合はサーバーはプロキシ経由で接続されません。

- `www.someco.com` で終わるドメインへの要求
- `blugrass.raleigh.ibm.com` または `keystone.raleigh.ibm.com` など、`.raleigh.ibm.com` で終わるドメインへの要求
- `myname.some.host.org:8080` など、`.some.host.org` で終わるドメインのポート 8080 への要求 (これには、同じドメインのその他のポートへの要求 (`myname.some.host.org` など、デフォルトのポート 80 を想定するもの) は含まれません。)

### デフォルト

なし

## NoCacheOnRange - Range 要求でキャッシングなしを指定する

デフォルトでは、ブラウザからの Range 要求を受信した際、Caching Proxy は、バックエンド・サーバーからのフル応答を必要とします。Caching Proxy は、要求の Range ヘッダーを除去し、その要求をバックエンド・サーバーに転送します。応答がプロキシ・サーバーのキャッシュに入ると、同じリソースに対する以降の要求は、その要求が Range 要求であるかどうかを問わず、プロキシ・サーバーからサービスを受けます。通常、Caching Proxy のデフォルトのアクションにより、パフォーマンスが向上し、クライアントの応答時間が短縮します。しかし、応答をキャッシュに入れることができない場合や、応答が非常に大きい場合には、デフォルトのアクションではパフォーマンスを低下させることがあります。

このデフォルト構成の使用時の問題を解決するには、NoCacheOnRange ディレクティブを使用します。このディレクティブで、Range 要求のキャッシングなしを指定します。

このディレクティブを、ibmproxy.conf ファイルでグローバルに使用可能にした場合、または PROXY マッピング・ルールのオプションとして使用可能にした場合、Caching Proxy は Range 要求ヘッダーをバックエンド・サーバーに転送します。ただし、Caching Proxy は、バックエンド・サーバーからの 206 (部分コンテンツ) 応答をキャッシュに入れることはしません。

NoCacheOnRange ディレクティブを使用可能に設定すると、以下の場合にプロキシのパフォーマンスが改善されます。

- 応答を頻繁にキャッシュに入れたり、更新したりできない場合。
- アプリケーションにとって、応答時間が重要である場合。

## 形式

NoCacheOnRange [on | off]

## 例

また、プロキシ・マッピング・ルールで NoCacheOnRange を有効にすることもできます。

```
Proxy /not-cachable/* http://server.com/no-cachable-resources/* NoCacheOnRange
```

## デフォルト

NoCacheOnRange off

# NoProxyHeader - ブロックするクライアント・ヘッダーを指定する

このディレクティブは、ブロックするクライアント URL ヘッダーを指定するために使用します。クライアントによって送信された任意の HTTP ヘッダーを、必須ヘッダーを含めてブロックすることができます。ヘッダーをブロックしているときは、十分な注意が必要です。共通ヘッダーには、以下のものが含まれます。

- **Pragma:** — 通常は、キャッシュ付きのブラウザおよびサーバーに、ファイルが要求されるたびに元のサーバーからファイルを取り出すように指示するために使用されます。
- **Referer:** — Request-URI が取得されたファイルの URL。

上記およびその他のヘッダーの詳細については、HTTP のプロトコル仕様書を参照してください。このディレクティブは、複数回指定することができます。

## 形式

NoProxyHeader *header*

## 例

```
NoProxyHeader Referer:
```

## デフォルト

なし

## NumClients - 使用するキャッシュ・エージェントのスレッドの数を指定する

このディレクティブは、キャッシュ・エージェントがキュー内のページを検索するときに使用するスレッドの数を指定するために使用します。スレッドの数を、内部ネットワークおよびインターネットへの接続の速度を基にします。可能な範囲は、1 から 100 までです。

注: 6 つを超えるスレッドを使用すると、コンテンツ・サーバーでは過剰な高速要求がもたらされる可能性があります。

### 形式

`NumClients number`

### デフォルト

`NumClients 4`

## ObjectType - オブジェクト・タイプ・ステップをカスタマイズする

このディレクティブは、オブジェクト・タイプ・ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、ファイル・システム内の要求されたオブジェクトを探し出して、その MIME タイプを識別します。

### 形式

`ObjectType request_template /path/file:function_name`

*request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、/front\_page.html、http://www.ics.raleigh.ibm.com、/pub\*、/\*、および \* はすべて有効です。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

### 例

`ObjectType /index.html /api/bin/icsextpgm.so:obj_type`

### デフォルト

なし



## OptimizeRuleMapping - ルールの数が増加したときに、着信要求のためのルール・マッピング・プロセスを最適化する

このディレクティブは、ルールの数が増加したときに、着信要求のルール・マッピング・プロセスを高速化します。

OptimizeRuleMapping ディレクティブを有効にすると、プロキシは、各ルールごとに 1 つずつ着信 URI 要求をマッピングする代わりに、接頭部ツリーに対して URI をマップします。接頭部ツリーは、プロキシがマッピング・ルール間の冗長なストリング比較を除去するのに役立ちます。その結果、構成内のルール数が 300 より大きくなったときに、Caching Proxy はより良いパフォーマンスが得られます。

### 形式

OptimizeRuleMapping [on | off]

### デフォルト

OptimizeRuleMapping off

## OutputTimeout - 出力タイムアウトを指定する

このディレクティブを使用して、サーバーがクライアントに出力を送信できる時間の最大値を設定します。時間制限は、ローカル・ファイルに対する要求、およびサーバーがプロキシとして機能する要求に適用されますが、ローカル CGI プログラムを開始する要求には適用されません。

このディレクティブに設定された時間制限内にサーバーが完全な応答を送信しなければ、サーバーは接続を終了します。時間の値は、時間 (hours)、分 (minutes または mins)、および秒 (seconds または secs) を任意に組み合わせて指定します。

### 形式

OutputTimeout *time*

### デフォルト

OutputTimeout 30 minutes

## PacFilePath - PAC ファイルを含むディレクトリーを指定する

このディレクティブは、リモートの構成 PAC ファイル形式を使用して生成されたプロキシの自動構成ファイルが入っているディレクトリーを指定するときに使用します。

### 形式

PacFilePath *directory\_path*

### デフォルト

- **Windows:** PacFilePath c:%Program Files%\IBM\edge\cp%\HTML\pacfiles
- **Linux および UNIX:** PacFilePath /opt/ibm/edge/cp/server\_root/pub/pacfiles

## Pass - 要求を受け入れるためのテンプレートを指定する

このディレクティブを使用して、サーバーからのファイルによって受け入れ、応答する要求のためのテンプレートを指定します。要求は、Pass ディレクティブのテンプレートに一致すると、後続のディレクティブの要求テンプレートとは比較されません。

### 形式

```
Pass request_template [file_path [server_IP_address | host_name]]
```

#### *request\_template*

サーバーに受け入れさせ、ファイルを使用して応答させたい要求のためのテンプレートを指定します。

テンプレートではアスタリスク (\*) をワイルドカードとして使用できます。スラッシュ (/) の直後の波形記号 (〰) は明示的に一致しなければならず、このためにワイルドカードを使用することはできません。

#### [*file\_path*]

サーバーが戻すファイルのパスを指定します。*file\_path* には、*request\_template* に含まれていれば、ワイルドカードを含めることができます。要求の *request\_template* ワイルドカードと一致する部分が *file\_path* のワイルドカードの代わりに挿入されます。

このパラメーターはオプションです。パスを指定しないと、要求そのものがパスとして使用されます。

#### [*server\_IP\_address* | *host\_name*]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) を指定するか、またはホスト名 (例えば、hostA.raleigh.ibm.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

### 例

- 以下の例において、サーバーは、オペレーティング・システムに応じて、リストされたパスからのファイルを使用して、/updates/parts/ で始まる要求に応答します。/updates/parts/ より後の部分も、また、すべてファイルを指定するために使用されます。

**Linux および UNIX システム:** Pass /updates/parts/\*  
/opt/ibm/edge/cp/server\_root/pub/\*

**Windows システム:** Pass /updates/parts/\* c:¥Program  
Files¥IBM¥edge¥cp¥pub¥\*

- 以下の例において、サーバーは、ディレクトリー /gooddoc からファイルを使用して、/gooddoc/ で始まる要求に応答します。したがって、サーバーは、ファイル /gooddoc/volume1/issue2/newsletter4.html 内の文書を使用して、要求 /gooddoc/volume1/issue2/newsletter4.html に応答します。

Pass /gooddoc/\*

- 以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーは、/parts/ で始まる要求を受信すると、要求が入ってきたネットワーク接続の IP アドレスを基にして、別のディレクトリーからのファイルを戻します。240.146.167.72 に入ってくる要求の場合、サーバーは /customerA/catalog/ からファイルを戻します。アドレス 0.83.100.45 の接続に入ってくる要求の場合には、サーバーは /customerB/catalog/ からファイルを戻します。

Pass /parts/\* /customerA/catalog/\* 240.146.167.72

Pass /parts/\* /customerB/catalog/\* 0.83.100.45

- 以下の例では、オプションのホスト名パラメーターを使用しています。サーバーは、/parts/ で始まる要求を受信すると、URL のホスト名に基づいて異なるディレクトリーのファイルを戻します。hostA に送信された要求の場合、サーバーは /customerA/catalog/ のファイルを戻します。hostB に送信された要求の場合、サーバーは /customerB/catalog/ のファイルを戻します。

#### AIX システム

Pass /Admin/\* /usr/lpp/internet/server\_root/Admin/\*  
Pass /Docs/\* /usr/lpp/internet/server\_root/Docs/\*  
Pass /errorpages/\* /usr/lpp/internet/server\_root/pub/errorpages/\*  
Pass /\* /usr/lpp/internet/server\_root/pub/\*

#### Solaris、HP-UX、および Linux システム

Pass /Admin/\* /opt/ibm/edge/cp/server\_root/Admin/\*  
Pass /Docs/\* /opt/ibm/edge/cp/server\_root/Docs/\*  
Pass /errorpages/\* /opt/ibm/edge/cp/server\_root/pub/errorpages/\*  
Pass /\* /opt/ibm/edge/cp/server\_root/pub/\*

## デフォルト

#### AIX システム

Pass /Admin/\* /usr/lpp/internet/server\_root/Admin/\*  
Pass /Docs/\* /usr/lpp/internet/server\_root/Docs/\*  
Pass /errorpages/\* /usr/lpp/internet/server\_root/pub/errorpages/\*  
Pass /\* /usr/lpp/internet/server\_root/pub/\*

#### HP-UX、Linux、および Solaris システム

Pass /Admin/\* /opt/ibm/edge/cp/server\_root/Admin/\*  
Pass /Docs/\* /opt/ibm/edge/cp/server\_root/Docs/\*  
Pass /errorpages/\* /opt/ibm/edge/cp/server\_root/pub/errorpages/\*  
Pass /\* /opt/ibm/edge/cp/server\_root/pub/\*

#### Windows システム

Pass /icons/\* C:¥Program Files¥IBM¥edge¥cp¥icons¥\*  
Pass /Admin/\* C:¥Program Files¥IBM¥edge¥cp¥Admin¥\*  
Pass /Docs/\* C:¥Program Files¥IBM¥edge¥cp¥Docs¥\*  
Pass /errorpages/\* C:¥Program Files¥IBM¥edge¥cp¥pub¥errorpages¥\*  
Pass /\* C:¥Program Files¥IBM¥edge¥cp¥pub¥\*

## PersistTimeout - クライアントが別の要求を送信するのを待機する時間を指定する

このディレクティブは、サーバーが持続接続を取り消す前に、クライアント要求間で待機する時間の長さを指定するときに使用します。この時間は、有効な任意の時間増分で指定できますが、通常は秒単位または分単位の数です。

サーバーは、別のタイムアウト・ディレクティブ `InputTimeout` を使用して、接続が確立された後に、クライアントが最初の要求を送信するまでどれだけ長く待機するかを決定します。入力タイムアウトについて詳しくは、244 ページの『`InputTimeout` - 入力タイムアウトを指定する』を参照してください。

サーバーは、最初の応答を送信した後に、`PersistTimeout` ディレクティブの値を使用して、持続接続を取り消すまでにそれぞれの後続要求ごとにどれだけ長く待機するかを決定します。

### 形式

`PersistTimeout time`

### デフォルト

`PersistTimeout 4 seconds`

## PICSDatabaseLookup - PICS ラベル検索ステップをカスタマイズする

このディレクティブは、指定した URL の PICS ラベルを検索するためにサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。関数では、要求されたファイルの PICS ラベルを動的に作成したり、あるいは代替ファイルまたはデータベースの中で PICS ラベルを検索することができます。

### 形式

`PICSDatabaseLookup /path/file:function_name`

`/path/file`

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

`function_name`

プログラム内のアプリケーション関数の名前を指定します。

### 例

`PICSDatabaseLookup /api/bin/icsext05.so:get_pics`

### デフォルト

なし

## PidFile (Linux および UNIX 専用) - Caching Proxy のプロセス ID を保管するファイルを指定する

Linux および UNIX 専用です。このディレクティブは、Caching Proxy のプロセス ID を入れるファイルの場所を指定するときに使用します。サーバー・プロセスが開始されると、そのプロセス ID (PID) をファイルに記録します。単一システム上で

サーバーの複数インスタンスを実行している場合は、各インスタンスに固有の PidFile ディレクティブがなければなりません。

## 形式

PidFile *path\_to\_pid\_file\_info*

## 例

PidFile /usr/pidinfo

## デフォルト

- ServerRoot ディレクティブが指定される場合: PidFile *server\_root* /ibmproxy-pid
- ServerRoot ディレクティブは指定されない場合: PidFile /tmp/ibmproxy-pid

## PKCS11DefaultCert、PKCS11DriverPath、PKCS11TokenPassword - IBM 4960 PCI 暗号アクセラレーター・カードをサポートする (AIX のみ)

AIX システムでは、IBM 4960 PCI 暗号アクセラレーター・カードをサポートするため、追加のディレクティブが提供されています。

これらの 3 つのディレクティブを使用すると、プロキシは、デバイス・ドライバを読み込む、トークン・デバイスをオープンする、およびデバイスに保管されている証明書にアクセスする、などが可能になります。デバイス・ドライバーがロードされると、プロキシ・サーバーは自動的にそのデバイスを使用して、SSL 通信速度を高速にします。

303 ページの『SSLCryptoCard - インストール済み暗号カードを指定する』も参照してください。

## 形式

PKCS11DefaultCert *default\_cert\_label*

トークン・デバイスに保管されている、デフォルトの SSL 証明書ラベルを指定します。

PKCS11DriverPath *absolute\_path\_to\_the\_card\_driver*

暗号アクセラレーター・カード用の、デバイス・ドライバーの絶対パスを指定します。

PKCS11TokenPassword *password*

トークン・デバイスをオープンするためのパスワードを指定します。

## 例

PKCS11DefaultCert MyDefaultCertInTheToken  
PKCS11DriverPath /usr/lib/pkcs11/PKCS11\_API.so  
PKCS11TokenPassword MyPasswordToOpenTheToken

## デフォルト

なし

## プラグイン・モジュールのディレクティブ

新しい機能やプラグインを使用可能にするために、以下にリストするディレクティブが Caching Proxy `ibmproxy.conf` ファイルに追加されました。これらのほとんどのディレクティブの編集には、「構成および管理」フォームは使用できません。それらの編集には、`vi` や `emacs` などの標準テキスト・エディターを使用する必要があります。本書では、この新しいディレクティブについて、それぞれの詳細をアルファベット順で示してあります。

- 232 ページの『ExternalCacheManager - IBM WebSphere Application Server からの動的キャッシング用の Caching Proxy の構成』
- 240 ページの『ICP\_Address - ICP 照会用の IP アドレスを指定する』
- 241 ページの『ICP\_Port - ICP 照会用のポート番号を指定する』
- 241 ページの『ICP\_Timeout - ICP 照会に対する最大待機時間を指定する』
- 240 ページの『Occupier - ICP クラスターのメンバーを指定する』
- 240 ページの『ICP\_MaxThreads - ICP 照会用の最大スレッド数を指定する』
- 300 ページの『SignificantURLTerminator - URL 要求の終了コードを指定する』
- 301 ページの『SSLCertificate - 証明書の鍵ラベルを指定する』
- 304 ページの『SSLOnly - HTTP 要求のリスナー・スレッドを使用不可にする』

`ibmproxy.conf` ファイルには、Caching Proxy プラグイン・モジュールの構成に使用するディレクティブは、以下の形式で入力する必要があります。

```
<MODULEBEGIN> plugin name
subdirective1
subdirective2

<MODULEEND>
```

それぞれのプラグイン・プログラムは、`ibmproxy.conf` ファイルを解析して、それぞれのサブディレクティブのブロックのみを読み取ります。Caching Proxy パーサーは、`<MODULEBEGIN>` と `<MODULEEND>` の間にあるものはすべて無視します。

Caching Proxy プラグイン・モジュールと一部の新しい機能では、API ディレクティブを `ibmproxy.conf` ファイルに追加する必要があります。プロキシ・サーバーは、リストされた順序でプラグイン・モジュールと対話するので、プロキシ構成ファイルの中でディレクティブを順序付けする際には十分注意してください。プロトタイプのディレクティブ (コメント形式の) は、`ibmproxy.conf` ファイルの API セクションに追加されています。この API ディレクティブは目的別の順序で配列されています。API ディレクティブを追加して新しい機能やプラグイン・モジュールを使用できるようにするには、各ディレクティブを構成ファイルのプロトタイプ・セクションに示されているように配列してください。あるいは、必要に応じて API ディレクティブをアンコメントして編集し、それぞれ必要な機能やプラグインに対するサポートを組み込んでください。ユーザー生成のプラグイン・モジュールは、製品と一緒に提供されたモジュールの後に追加してください。

## Port - サーバーが要求を listen するポートを指定する

このディレクティブは、サーバーが要求を `listen` するポートの番号を指定するときに使用します。HTTP の標準のポート番号は、80 です。1024 未満の他のポート番



号は、他の TCP/IP アプリケーション用に予約されているので使用できません。  
Proxy Web サーバー用に使用される共通ポートは、8080 と 8008 です。

80 以上のポート番号を使用する場合、クライアントはサーバーへの要求の際、特定のポート番号を指定しなければなりません。ポート番号はコロン (:) の後に付き、URL のホスト名の後に記述します。例えば、ブラウザから、URL `http://www.turfco.com:8008/` は、ポート 8008 で listen している `www.turfco.com` という名前のホストからのデフォルトのウェルカム・ページを要求します。

**ibmproxy** コマンドで **-p** オプションを指定すると、サーバーの始動時にこの設定値をオーバーライドすることができます。

## 形式

Port *number*

このディレクティブを変更した場合は、手動でサーバーを停止してから再始動しなければ、変更が有効になりません。再始動しただけでは、サーバーは変更を認識しません。(15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。)

## デフォルト

Port 80

## PostAuth - PostAuth ステップをカスタマイズする

このディレクティブは、PostAuth ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、前のステップまたは他の PostAuth ハンドラーからの戻りコードに関係なく実行されます。このディレクティブを使用すると、要求を処理するために割り振られたすべてのリソースをクリーンアップすることができます。

## 形式

PostAuth */path/file:function\_name*

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

## 例

AuthExit */ics/api/bin/icsext05.so:post\_exit*

## デフォルト

なし

## PostExit - PostExit ステップをカスタマイズする

このディレクティブは、PostExit ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、前のステップまたは他の PostExit ハンドラーからの戻りコードに関係なく実行されます。こ



のディレクティブを使用すると、要求を処理するために割り振られたすべてのリソースをクリーンアップすることができます。

## 形式

`PostExit /path/file:function_name`

`/path/file`

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

`function_name`

プログラム内でのアプリケーション関数名を指定します。

## 例

`PostExit /ics/api/bin/icsext05.so:post_exit`

## デフォルト

なし

## PreExit - PreExit ステップをカスタマイズする

このディレクティブは、PreExit ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、クライアント要求が読み取られてから、他の処理が行われるまでの間に実行されます。このステップの実行中に GoServe モジュールを呼び出すことができます。

## 形式

`PreExit /path/file:function_name`

`/path/file`

拡張子を含む、コンパイル済み DLL の完全修飾ファイル名を指定します。

`function_name`

プログラム内でのアプリケーション関数名を指定します。

## 例

`PreExit /ics/api/bin/icsext05.so:pre_exit`

## デフォルト

なし

## Protect - テンプレートと一致する要求の保護セットアップを活性化する

このディレクティブを使用して、テンプレートと一致する要求のための保護セットアップ規則を活性化します。

**注:** 保護が正しく機能するために、DefProt および Protect ディレクティブは、構成ファイル内のすべての Pass、Exec、または Proxy ディレクティブの前になければなりません。

保護セットアップは、保護サブディレクティブで定義されます。このディレクティブの形式は、保護サブディレクティブが入っているラベルまたはファイルを指す

か、あるいは保護サブディレクティブを **Protect** ディレクティブの一部に組み込む必要があるかどうかによって異なります。

## 形式

このパラメーターは、以下のどの形式であっても構いません。

- **Protect** ディレクティブは、保護サブディレクティブが入っている別個のファイルの絶対パスおよびファイル名として指定することができます。これは、また、前の **Protection** ディレクティブで定義された名前と一致する保護セットアップ・ラベル名によって指定することもできます。この場合は、保護ディレクティブに保護サブディレクティブが含まれています。次の形式を使用します。

```
Protect request_template [setup_file | label[
    [FOR Server_IP_address | host_name]
```

注: ディレクティブは、2 行に表示されている場合でも、1 行に入力しなければなりません。

- **Protect** ディレクティブの一部として組み込む実際の保護サブディレクティブを指定することができます。サブディレクティブは、中括弧 ({} ) で囲むことが必要です。 左の中括弧は、**Protect** ディレクティブと同じ行の最後の文字でなければなりません。その後の行に、サブディレクティブを 1 行に 1 つずつ指定します。右の中括弧は、最後のサブディレクティブ行の後の行に単独で置かなければなりません。 中括弧で囲まれた中に、コメント行を置くことはできません。保護サブディレクティブを **Protect** ディレクティブの一部として組み込むには、形式は次のようになります。

```
Protect request_template [FOR Server_IP_address | hhost_name]
    subdirective value
    subdirective value
    .
    .
    .
}
```

以下のパラメーターが使用されます。

### *request\_template*

保護を活動化する対象の要求のテンプレートを指定します。サーバーは、受信したクライアント要求をテンプレートと比較し、一致する場合は保護を活動化します。

### *[setup\_file | label]*

保護サブディレクティブが入っているラベルまたはファイルを指している場合に、このパラメーターは、*request\_template* に一致する要求について活動化する保護セットアップを指定します。

このパラメーターはオプションです。このパラメーターを省略すると、保護セットアップは、一致するテンプレートが入っている最新の **DefProt** ディレクティブにより定義されます。

### *[FOR server\_IP\_address | host\_name]*

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワ

ーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。IP アドレスを保護すると、IP アドレスと完全修飾ホスト名の両方が保護されます。ただし、サーバーが、完全修飾ホスト名以外の名前（例えば、ホスト名ファイル内の項目など）を使用してそのネットワーク内から呼び出された場合は、保護されません。

例:

```
Protect http://x.x.x.x PROT-ADMIN
```

Web ブラウザー内:

- `http://x.x.x.x` は保護される
- `http://hostname.example.com` は保護される
- `http://hostname` は保護されない

例:

```
Protect http://hostname.example.com PROT-ADMIN
```

Web ブラウザー内:

- `http://x.x.x.x` は保護されない
- `http://hostname.example.com` は保護される
- `http://hostname` は保護されない

IP アドレス（例えば、FOR 240.146.167.72）を指定するか、あるいはホスト名（例えば、FOR hostA.bcd.com）を指定することができます。

サーバー IP アドレスの指定にワイルドカード文字を使用することはできません。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや URL のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

注: `[server_IP_address | host_name]` パラメーターは、`[setup_file | label]` パラメーターか `subdirective value` パラメーターのどちらかとともに使用されます。

- `[server_IP_address | host_name]` を `[setup_file | label]` とともに使用するには、FOR または他の任意の文字ストリング（ブランクは含まない）を `[setup_file | label]` パラメーターと `[server_IP_address | host_name]` パラメーターの間に挿入する必要があります。
- `[server_IP_address | host_name]` を `subdirective value` パラメーターとともに使用するには、FOR を `IP_address` または `host_name` の前に組み込まない ようにしてください。

#### *subdirective value*

保護サブディレクティブを Protect ディレクティブの一部として組み込むには、このパラメーターを使用してください。保護サブディレクティブの説明については、次を参照してください。

- 279 ページの『AuthType - 認証タイプを指定する』

- 279 ページの『DeleteMask - ファイルを削除できるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『GetMask - ファイルを取得できるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『GroupFile - 関連グループ・ファイルの場所を指定する』
- 280 ページの『Mask - HTTP 要求を行うことができるユーザー名、グループ、およびアドレスを指定する』
- 280 ページの『PasswdFile - 関連するパスワード・ファイルの場所を指定する』
- 281 ページの『PostMask - ファイルを POST できるユーザー名、グループ、およびアドレスを指定する』
- 281 ページの『PutMask - ファイルを PUT できるユーザー名、グループ、およびアドレスを指定する』
- 281 ページの『ServerID - パスワード・ファイルに関連付けられる名前を指定する』

## 例

- 以下の例では、サーバーは次のように保護を活動化します。
  - /secret/scoop/ で始まる要求は保護を活動化します。保護セットアップは、/server/protect/setup1.acc 保護セットアップ・ファイルで定義されています。Protect ディレクティブに保護セットアップが指定されていないため、前もって一致している DefProt ディレクティブの保護セットアップが使用されます。
  - /secret/business/ で始まる要求は保護を活動化します。保護セットアップは、BUS-PROT というラベルが付いている Protection ディレクティブに定義されています。
  - /topsecret/ で始まる要求は保護を活動化します。保護セットアップは、Protect ディレクティブに直接含まれます。

これらの例では、IP アドレスを使用しています。サーバーは、/secret/ または /topsecret/ で始まる要求を受信すると、その要求が入ってきたネットワーク接続の IP アドレスに基づいて、要求に対して異なる保護セットアップを活動化します。

- 0.67.106.79 に入ってくる /secret/ 要求の場合には、サーバーは CustomerA-PROT のラベルで、Protection ディレクティブで定義した保護セットアップを活動化します。0.67.106.79 に入ってくる /topsecret/ 要求の場合には、サーバーは Protect ディレクティブでインライン定義した /topsecret/ の保護セットアップを活動化します。
- 0.83.100.45 に入ってくる /secret/ 要求の場合には、サーバーは CustomerB-PROT のラベルで、Protection ディレクティブで定義した保護セットアップを活動化します。0.83.100.45 に入ってくる /topsecret/ 要求の場合には、サーバーは Protect ディレクティブでインライン定義した /topsecret/ の保護セットアップを活動化します。

```
Protection BUS-PROT {
    UserID      busybody
    GroupID     webgroup
}
```

```

AuthType Basic
ServerID restricted
PasswdFile /docs/WWW/restrict.pwd
GroupFile /docs/WWW/restrict.grp
GetMask authors
PutMask authors
}
DefProt /secret/* /server/protect/setup1.acc
Protect /secret/scoop/*
Protect /secret/business/* BUS-PROT
Protect /topsecret/* {
    AuthType Basic
    ServerID restricted
    PasswdFile /docs/WWW/restrict.pwd
    GroupFile /docs/WWW/restrict.grp
    GetMask topbrass
    PutMask topbrass
}
Pass /secret/scoop/* /WWW/restricted/*
Pass /secret/business/* /WWW/confidential/*
Pass /topsecret/* /WWW/topsecret/*

Protect /secret/* CustomerA-PROT FOR 0.67.106.79
Protect /secret/* CustomerB-PROT FOR 0.83.100.45
Protect /topsecret/* 0.67.106.79 {
    AuthType Basic
    ServerID restricted
    PasswdFile /docs/WWW/customer-A.pwd
    GroupFile /docs/WWW/customer-A.grp
    GetMask A-brass
    PutMask A-brass
}
Protect /topsecret/* 0.83.100.45 {
    AuthType Basic
    ServerID restricted
    PasswdFile /docs/WWW/customer-B.pwd
    GroupFile /docs/WWW/customer-B.grp
    GetMask B-brass
    PutMask B-brass
}

```

- 以下の例では、仮想ホストを使用しています。サーバーは、`/secret/` または `/topsecret/` で始まる要求を受信すると、URL のホスト名に基づいて、要求に対して異なる保護セットアップを活動化します。
  - `hostA.bcd.com` に送信された `/secret/` 要求に対して、サーバーは、`CustomerA-PROT` のラベルを付けて `Protection` ディレクティブで定義した、保護セットアップを活動化します。`hostA.bcd.com` に送信された `/topsecret/` 要求に対して、サーバーは、`Protect` ディレクティブでインラインに定義した、`/topsecret/` 用の保護セットアップを活動化します。
  - `hostB.bcd.com` に送信された `/secret/` 要求に対して、サーバーは、`CustomerB-PROT` のラベルを付けて `Protection` ディレクティブで定義した、保護セットアップを活動化します。`hostB.bcd.com` に送信された `/topsecret/` 要求に対して、サーバーは、`Protect` ディレクティブでインラインに定義した、`/topsecret/` 用の保護セットアップを活動化します。
  - 代行された要求に対して、サーバーは `proxy-prot` のラベルを付けて `Protection` ディレクティブで定義した、保護セットアップを活動化します。例えば、次のとおりです。

```
Protect http://host1/* proxy-prot
```

```

Protect    /secret/*    CustomerA-PROT    FOR    hostA.bcd.com
Protect    /secret/*    CustomerB-PROT    FOR    hostB.bcd.com
Protect    /topsecret/*    hostA.bcd.com    {
    AuthType Basic
    ServerID restricted
    PasswdFile /docs/WWW/customer-A.pwd
    GroupFile  /docs/WWW/customer-A.grp
    GetMask A-brass
    PutMask A-brass
}
Protect    /topsecret/*    hostB.bcd.com {
    AuthType Basic
    ServerID restricted
    PasswdFile /docs/WWW/customer-B.pwd
    GroupFile  /docs/WWW/customer-B.grp
    GetMask B-brass
    PutMask B-brass
}

```

## デフォルト

デフォルトで、「構成および管理」フォームのための保護は、/admin-bin/\* という要求テンプレートを指定した Protect ディレクティブによって提供されます。

## Protection - 名前付き保護セットアップを構成ファイル内に定義する

このディレクティブを使用して、保護セットアップを構成ファイル内に定義します。保護セットアップに名前を付け、保護サブディレクティブを使用して保護のタイプを定義します。

注:

1. 構成ファイルでは、Protection ディレクティブは、このディレクティブを指す DefProt または Protect ディレクティブのどれよりも前に置いてください。
2. 使用している保護ルールでドメイン・ネームを使用するには、DNS-Lookup ディレクティブを on に設定してください。

## 形式

```

Protection label_name {
    subdirective value
    subdirective value
    .
    .
    .
}

```

*label\_name*

この保護セットアップに関連付ける名前を指定します。後続の DefProt および Protect ディレクティブでは、この名前を使用してこの保護セットアップを指すことができます。

*subdirective value*

サブディレクティブは中括弧 ( { } ) で囲まれます。左の中括弧は、*label\_name* と同じ行の最後の文字でなければなりません。その後の行に、サブディレクティブを 1 行に 1 つずつ指定します。右の中括弧は、最後のサブディレクティブ行の後の行に単独で置かなければなりません。中括弧で囲まれた中に、コメント行を置くことはできません。

保護サブディレクティブの説明については、『Protection subdirectives - 一連のリソースの保護方法を指定する』を参照してください。

## 例

```
Protection NAME-ME {
    AuthType Basic
    ServerID restricted
    PasswdFile /WWW/password.pwd
    GroupFile /WWW/group.grp
    GetMask groupname
    PutMask groupname
}
```

## デフォルト

```
Protect /admin-bin/* {
    ServerId      Private_Authorization
    AuthType      Basic
    GetMask       All@(*)
    PutMask       All@(*)
    PostMask      All@(*)
    Mask          All@(*)
    PasswdFile    /opt/ibm/edge/cp/server_root/protect/webadmin.passwd
}
```

## Protection subdirectives - 一連のリソースの保護方法を指定する

保護セットアップ内で使用できる保護サブディレクティブについて、以下に説明します。 サブディレクティブは、アルファベット順に並んでいます。

保護セットアップは、別個のファイルとしたり、あるいは DefProt、Protect、または Protection ディレクティブの一部として構成ファイルに組み込むことができます。

### AuthType - 認証タイプを指定する

この保護サブディレクティブは、ユーザー名とパスワードに基づいてアクセスを制限するときに使用します。パスワードがクライアントからサーバーに送信されたときに使用される認証のタイプを指定します。基本認証 (AuthType Basic) では、パスワードは非暗号化テキストとしてサーバーに送信されます。パスワードはエンコードされますが、暗号化はされません。

#### デフォルト:

```
AuthType Basic
```

### DeleteMask - ファイルを削除できるユーザー名、グループ、およびアドレスを指定する

この保護サブディレクティブは、保護ディレクトリーに対する DELETE 要求を出ることができるユーザー名、グループ、およびアドレス・テンプレートを指定するときに使用します。

#### 例:

```
DeleteMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```



## GetMask - ファイルを取得できるユーザー名、グループ、およびアドレスを指定する

この保護サブディレクティブは、保護ディレクトリーに対する GET 要求を出すことができるユーザー名、グループ、およびアドレス・テンプレートを指定するときに使用します。

例:

```
GetMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

デフォルト:

```
GetMask All@(*)
```

## GroupFile - 関連グループ・ファイルの場所を指定する

この Protection サブディレクティブは、保護セットアップで使用するサーバー・グループ・ファイルのパスおよびファイル名を指定するときに使用します。これによって、サーバー・グループ・ファイル内で定義されているグループを以下の場所で使用することができます。

- 保護セットアップの一部として使用されているマスク・サブディレクティブ。(マスク・サブディレクティブは、DeleteMask、GetMask、Mask、PostMask、および PutMask です。)
- 保護セットアップで保護されているディレクトリーにある ACL ファイル。

例:

```
GroupFile /docs/etc/WWW/restrict.group
```

## Mask - HTTP 要求を行うことができるユーザー名、グループ、およびアドレスを指定する

このサブディレクティブは、他のマスク・サブディレクティブでは扱われない HTTP 要求を出すことが許可されたユーザー名、グループ、およびアドレス・テンプレートを指定するときに使用します。

例:

```
Mask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

注: Mask ディレクティブを使用する場合は、大/小文字の区別があることに十分注意してください。以下は、ユーザー ID に指定される Mask 保護の例です。

```
MASK WEBADM,webadm
```

## PasswdFile - 関連するパスワード・ファイルの場所を指定する

この保護サブディレクティブは、ユーザー名とパスワードに基づいてアクセスを制限するときに使用します。この保護セットアップが使用するパスワード・ファイルのパス名とファイル名を指定します。

一部のブラウザーは、ホスト内のセキュリティ・レルム (ServerID) によってユーザー ID やパスワードをキャッシュに入れるため、ServerID およびパスワード・ファイルを指定する際は、以下のガイドラインに従ってください。

- 同じパスワード・ファイルを使用する保護セットアップの場合は、同じ ServerID を使用します。

- 異なるパスワード・ファイルを使用する保護セットアップの場合は、異なる ServerID を使用します。

例:

```
PasswdFile /docs/etc/WWW/restrict.password
```

注: パスワード・ファイルのパスまたはファイル名に、埋め込まれたブランクが含まれている場合は、そのパスとファイル名全体を引用符 (") で囲む必要があります。

```
PasswdFile "c:¥test this¥admin.pwd"
```

## PostMask - ファイルを POST できるユーザー名、グループ、およびアドレスを指定する

セキュア・サーバーの場合、保護ディレクトリーに対する POST 要求を出すことができるユーザー、グループ、およびアドレス・テンプレートを指定するとき、この保護サブディレクティブを使用します。

例:

```
PostMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

## PutMask - ファイルを PUT できるユーザー名、グループ、およびアドレスを指定する

この保護サブディレクティブは、保護ディレクトリーに対する PUT 要求を出すことができるユーザー、グループ、およびアドレス・テンプレートを指定するときに使用します。

例:

```
PutMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

## ServerID - パスワード・ファイルに関連付けられる名前を指定する

この保護サブディレクティブは、ユーザー名とパスワードに基づいてアクセスを制限するときに使用します。使用されるパスワード・ファイルに関連付けたい名前を指定します。名前は実際のコンピュータの名前でなくても構いません。

この名前は、要求側に送信される ID として使用されます。それぞれの保護セットアップは別々のパスワード・ファイルを使用できるので、保護セットアップに名前に関連付けておくと、クライアントはどのパスワードを送信すればよいかがわかります。ほとんどのクライアントは、ユーザー名とパスワードの入力を求めるプロンプトを表示するときに、この名前を表示します。

一部のブラウザーは、ホスト内のセキュリティ・レルム (ServerID) によってユーザー ID やパスワードをキャッシュに入れるため、ServerID およびパスワード・ファイルを指定する際は、以下のガイドラインに従ってください。

- 同じパスワード・ファイルを使用する保護セットアップの場合は、同じ ServerID を使用します。
- 異なるパスワード・ファイルを使用する保護セットアップの場合は、異なる ServerID を使用します。

例:

## Proxy - プロキシ・プロトコルまたはリバース・プロキシを指定する

このディレクティブは、Caching Proxy がどのプロトコルを処理すべきかを指示し、要求をサーバーにマップするときに使用します。有効なプロトコルは、http、ftp、および gopher です。

このプロキシ・ディレクティブは、要求をリモート・サーバーに渡します。例えば、次のディレクティブの場合は、すべての要求を指定の URL に転送します。

```
Proxy /* http://proxy.server.name/*
```

セキュア・リバース・プロキシ・サーバーの場合は、次のディレクティブを使用してください。

```
Proxy /* https://proxy.server.name/*
```

プロキシ・サーバーの制限を抑えたい場合は、構成ファイルから以下のディレクティブをアンコメントします。ただし、プロキシがリバース・プロキシとして構成されると、これらのディレクティブにセキュリティ上の問題が発生する場合があります。

```
Proxy http:*
Proxy ftp:*
Proxy gopher:*
```

オプション・パラメーター:

- UseSession

これは、リバース・プロキシ構成にのみ適用されます。

このオプションは、クライアント・サイドのソケットと発信側のソケットの間で、1 対 1 のマッピングを維持するよう Caching Proxy に指示します。このオプションは、接続ベースの認証など、一部のアプリケーションの場合に有効です。例えば、プロキシがサーバー・サイドのソケットを生かしたまま、同じクライアント・サイド・ソケットからの要求のために、そのソケットを再利用する必要があるようなアプリケーションの場合です。

- NoCaching

プロキシ・ルールが一致すると、このオプションは、対応する応答をキャッシュに入れないよう、プロキシに指示します。

- NoCacheOnRange

プロキシ・ルールが一致し、かつ要求に Range ヘッダーがある場合、このオプションは、対応する応答をキャッシュに入れないよう、プロキシに指示します。詳しくは、263 ページの『NoCacheOnRange - Range 要求でキャッシングなしを指定する』を参照してください。

- NoJunction

これは、リバース・プロキシ構成にのみ適用されます。

このオプションは、ジャンクション再書き込みプラグインが有効である場合に使用します。このオプションでは、着信 URL が一致した場合、対応する応答をプロキシーが再書き込みすることを許可しません。詳細については、47 ページの『ジャンクション再書き込みの使用可能化 (オプション)』と 48 ページの『JunctionPrefix オプションを使用するジャンクションの定義 (推奨される方法)』を参照してください。

- JunctionPrefix

これは、リバース・プロキシー構成にのみ適用されます。

このオプションは、ジャンクション再書き込みプラグインが有効である場合に使用します。このオプションでは、プロキシー・ルール内の最初の URL パターンからジャンクションの接頭部を推論する代わりに、明示的にジャンクションの再書き込み接頭部を宣言します。詳細については、47 ページの『ジャンクション再書き込みの使用可能化 (オプション)』と 48 ページの『JunctionPrefix オプションを使用するジャンクションの定義 (推奨される方法)』を参照してください。

## 形式

```
Proxy request_template target_server_path [[ip]:port]  
[UseSession | NoCaching | NoCacheOnRange | NoJunction | JunctionPrefix:url_prefix]
```

## 例

以下は、Proxy ディレクティブの UseSession オプションの例です。

```
Proxy /abc/* http://server1/default/abc/* :80 UseSession
```

着信クライアント要求がポート 80 から来るときに、クライアント要求の URL が /abc/\* というパターンに一致すると、URL は http://server1/default/abc/\* にマップされます。

## デフォルト

なし。

## ProxyAccessLog - プロキシー・アクセス・ログ・ファイルのパスを指定する

このディレクティブは、サーバーにプロキシー要求のアクセス統計をログに記録させたい場所のファイルのパスおよびファイル名を指定するときに使用します。デフォルトでは、クライアント要求に対してプロキシーとして振る舞うたびに、サーバーがこのログに項目を書き込みます。特定のクライアントからの要求をログに記録したくない場合は、NoLog ディレクティブを使用することができます。

サーバーは、午前 0 時に新規ログ・ファイルを開始します (サーバーが稼働している場合)。午前 0 時にサーバーが稼働していない場合は、その日における最初のサーバー始動時に新規ログ・ファイルの記録を開始します。ログ・ファイル作成時に、サーバーは、指定されたファイル名を使用し、日付接尾部または拡張子を付加します。日付接尾部または拡張子は、Mmmddyyyy という形式です。Mmm は月の最初の 3 文字、dd は日、yyyy は年です。

古いログ・ファイルは、ハード・ディスク上で大量のスペースを使用する可能性があるため、これらのファイルは除去するようにしてください。

## 形式

ProxyAccessLog *path/file*

## デフォルト

- **Linux** および **UNIX** システム: ProxyAccessLog  
/opt/ibm/edge/cp/server\_root/logs/proxy
- **Windows** システム: ProxyAccessLog *drive:¥Program*  
*Files¥IBM¥edge¥cp¥logs¥proxy*

## ProxyAdvisor - プロキシ要求のサービスをカスタマイズする

このディレクティブは、Proxy Advisor ステップ中にサーバーで呼び出したいカスタマイズ済みアプリケーションを指定するときに使用します。このコードは、要求に対応します。

## 形式

ProxyAdvisor */path/file:function\_name*

*/path/file*

コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

例:

ProxyAdvisor /api/bin/customadvise.so:proxyadv

## デフォルト

なし

## ProxyForwardLabels - PICS フィルター操作を指定する

ProxyForwardLabels ディレクティブは、プロキシ・サーバーおよびクライアントでか、あるいはプロキシ階層内の 2 つのプロキシで、PICS フィルター操作を指定するときに使用します。

ProxyForwardLabels を On に設定した場合に、プロキシ・サーバーは、起点サーバー、ラベル・ビューロー、Caching Proxy のラベル・キャッシュ、およびラベル提供側プラグインからのラベルを含め、見つかったすべての PICS ラベルについて PICS-Label: HTTP ヘッダーを生成します。

ProxyForwardLabels が Off の場合は、PICS-Label: HTTP ヘッダーは生成されません。

## 形式

ProxyForwardLabels {on | off}

## デフォルト

ProxyForwardLabels Off

## ProxyFrom - From: ヘッダーのクライアントを指定する

このディレクティブは、From: ヘッダーを生成するために使用します。これは、通常、プロキシ管理者の電子メール・アドレスを指定する場合に使用されます。

### 形式

ProxyFrom *e-mail\_address*

### 例

ProxyFrom webmaster@proxy.ibm.com の設定の結果として、以下のヘッダーが変更されます。

#### 元のヘッダー

Location: http://www.ibm.com/  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
Pragma: no-cache

#### 変更後のヘッダー

Location: http://www.ibm.com/  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
**From: webmaster@proxy.ibm.com**  
Pragma: no-cache

### デフォルト

なし

## ProxyIgnoreNoCache - 再ロード要求を無視する

このディレクティブは、ユーザーがブラウザで「再読み込み」をクリックした場合のサーバーの反応を指定するときに使用します。ProxyIgnoreNoCache ディレクティブを on に設定すると、高負荷の間中は、サーバーは宛先サーバーにページを要求せず、ファイルのキャッシュ・コピーが使用できればこれを提供します。サーバーは、本質的に、ブラウザから送信された "Pragma: no-cache" ヘッダーを無視します。

### 形式

ProxyIgnoreNoCache {on | off}

### デフォルト

ProxyIgnoreNoCache off

## ProxyPersistence - 持続接続機能を許可する

このディレクティブは、クライアントとの持続接続を維持するかどうかを指定するときに使用します。持続接続機能によって、ユーザーの待ち時間が短縮されてプロキシ・サーバー上の CPU の負荷が軽減される一方で、より多くのリソースが必要とされます。持続接続機能は、より多くのスレッドと、そのためにより多くのプロキシ・サーバー上のメモリーを必要とします。

プロキシのいずれかが HTTP 1.1 に準拠していない場合、持続接続機能をマルチレベルのプロキシ・サーバー・セットアップに使用しないでください。

### 形式

ProxyPersistence {on | off}

## デフォルト

ProxyPersistence on

## ProxySendClientAddress - Client - IP: ヘッダーを生成する

このディレクティブは、プロキシがクライアントの IP アドレスを宛先サーバーに転送するかどうかを指定するときに使用します。

### 形式

ProxySendClientAddress {*Client\_IP*: | OFF}

### 例

ディレクティブ ProxySendClientAddress *Client-IP*: の結果として、以下のヘッダーが変更されます。

#### 元のヘッダー

Location: http://www.ibm.com/  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
Pragma: no-cache

#### 変更後のヘッダー

Location: http://www.ibm.com  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
**Client-IP: 0.67.199.5**  
Pragma: no-cache

## デフォルト

なし

## ProxyUserAgent - User Agent スtringを変更する

このディレクティブは、クライアントが送信したStringを置き換える User Agent Stringを指定するときに使用します。これにより、Web サイトを訪問中の匿名性がさらに高まります。しかし、User Agent Stringに基づいてページをカスタマイズしてあるサイトもあります。ProxyUserAgent ディレクティブを使用することで、このようなカスタム・ページは表示されません。

### 形式

ProxyUserAgent *product\_name/version*

### 例

ディレクティブ ProxyUserAgent Caching Proxy/6.1 の結果として、以下のヘッダーが変更されます。

#### 元のヘッダー

Location: http://www.ibm.com/  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
**User Agent: Mozilla/ 2.02 OS2**  
Pragma: no-cache

#### 変更後のヘッダー

Location: http://www.ibm.com  
Last Modified: Tue 5 Nov 1997 10:05:39  
GMT  
**User Agent: Caching Proxy/6.1**  
Pragma: no-cache

## デフォルト

なし



## ProxyVia - HTTP ヘッダーの形式を指定する

このディレクティブは、HTTP ヘッダーの形式を制御するときに使用します。このディレクティブに使用できる 4 つの値があります。ProxyVia が Full に設定されると、Caching Proxy は Via ヘッダーを要求または応答の中に追加します。Via ヘッダーがすでにストリーム内にある場合には、Caching Proxy はその終わりにホスト情報を追加します。Set に設定されると、Caching Proxy は Via ヘッダーをホスト情報に設定し、Via ヘッダーがすでにストリーム内にある場合は、Caching Proxy がそれを除去します。Pass に設定されると、Caching Proxy はすべてのヘッダー情報をそのまま転送します。Block に設定されると、Caching Proxy は Via ヘッダーを転送しません。

### 形式

```
ProxyVia {Full | Set | Pass | Block}
```

### 例

```
ProxyVia Pass
```

### デフォルト

```
ProxyVia Full
```

## ProxyWAS - 要求が WebSphere Application Server に送信されることを指定する

これは、リバース・プロキシー構成にのみ適用されます。

ProxyWAS マッピング・ディレクティブは Proxy ディレクティブと同様に作動しますが、一致する要求を WebSphere Application Server に送ることも Caching Proxy に指示します。このディレクティブの使用法の例については、282 ページの『Proxy - プロキシー・プロトコルまたはリバース・プロキシーを指定する』を参照してください。

### 形式

```
ProxyWAS request_template target_server_path [[ip]:port]  
[UseSession | NoCaching | NoCacheOnRange | NoJunction | JunctionPrefix:url_prefix]
```

### デフォルト

なし

## PureProxy - 専用プロキシーを使用不可にする

このディレクティブは、サーバーがプロキシーとして機能するか、またはプロキシーとコンテンツ・サーバーの両方として機能するかを指定するために使用します。Caching Proxy をプロキシーとしてだけ使用することをお勧めします。

### 形式

```
PureProxy {on | off}
```

### デフォルト

```
PureProxy on
```

## PurgeAge - ログの経過時間限度を指定する

このディレクティブは、ログがパージされるまでの経過時間 (日数) を指定するとき 사용합니다。PurgeAge が 0 の場合、ログが削除されることはありません。

注: プラグインが、その日の、または前日のログを削除することはありません。

### 形式

PurgeAge *number*

### デフォルト

PurgeAge 7

### 関連ディレクティブ

- 212 ページの『CompressAge - ログをいつ圧縮するかを指定する』
- 214 ページの『CompressDeleteAge - ログをいつ削除するかを指定する』
- 213 ページの『CompressCommand - 圧縮コマンドおよびパラメーターを指定する』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- 『PurgeSize - ログ・アーカイブのサイズの限度を指定する』

## PurgeSize - ログ・アーカイブのサイズの限度を指定する

このディレクティブは、ログ・アーカイブがパージされるまでにどれだけの大きさ (メガバイト単位) になり得るかを指定するとき 사용합니다。PurgeSize ディレクティブが 0 に設定されると、サイズに限界はなく、ファイルは削除されません。

PurgeSize の設定は、ログ・タイプのログのすべて を参照します。例えば、エラーをログに記録していて (すなわち、構成ファイルに ErrorLog 項目が作成されていて)、PurgeSize が 10 MB として定義されている場合には、Caching Proxy はすべてのエラー・ログのサイズを計算して合算し、次に、合計サイズが 10 MB 未満になるまでログを削除します。

注: プラグインが、その日の、または前日のログを削除することはありません。ログ・ファイルが削除されると、各ログ・タイプ・ファイルが PurgeSize によって定義された値 (メガバイト単位) より小か等しくなるまで、最も古いログから順に削除されます。

### 形式

PurgeSize *number\_of\_MB*

### デフォルト

PurgeSize 0

### 関連ディレクティブ

- 212 ページの『CompressAge - ログをいつ圧縮するかを指定する』
- 214 ページの『CompressDeleteAge - ログをいつ削除するかを指定する』

- 213 ページの『CompressCommand - 圧縮コマンドおよびパラメーターを指定する』
- 251 ページの『LogArchive - ログ・アーカイブの動作を指定する』
- 260 ページの『Midnight - ログのアーカイブに使用される API プラグインを指定する』
- 288 ページの『PurgeAge - ログの経過時間限度を指定する』

## RCAConfigFile - ConfigFile の別名を指定する

このディレクティブは、リモート・キャッシュ・アクセス構成ファイルの名前および場所を指定する場合に使用します。

注: RCA 構成ファイルは `ibmproxy.conf` にマージされています。逆方向の互換性については、ConfigFile の別名として RCAConfigFile がサポートされています。

### 形式

RCAConfigFile */etc/file\_name*

### 例

RCAConfigFile */etc/user2rca.conf*

### デフォルト

RCAConfigFile */etc/rca.conf*

## RCAThreads - ポート当たりのスレッドの数を指定する

このディレクティブは、RCA ポート上で作動するスレッドの数を指定するときに使用します。

### 形式

RCAThreads *number\_of\_threads*

### 例

RCAThreads 50

### デフォルト

MaxActiveThreads x [(ArraySize -1) / (2 x ArraySize -1)]

## ReadTimeout - 接続の時間制限を指定する

このディレクティブを使用して、接続が取り消されるまでにネットワーク活動をせずにいられる時間の制限を指定します。

### 形式

ReadTimeout *time*

### デフォルト

ReadTimeout 5 minutes

## Redirect - 別のサーバーに送信される要求のテンプレートを指定する

このディレクティブは、受け入れて、別のサーバーに送信したい要求のテンプレートを指定するときに使用します。要求が `Redirect` ディレクティブ上のテンプレートと一致すると、その要求は構成ファイル内の他のディレクティブ上のテンプレートとは比較されません。

### 形式

`Redirect request_template URL [server_IP_address | host_name]`

#### *request\_template*

サーバーが別のサーバーに送信する要求のためのテンプレートを指定します。

テンプレートではアスタリスク (\*) をワイルドカードとして使用できます。スラッシュ (/) の直後の波形記号 (~) は明示的に一致しなければならない、このためにワイルドカードを使用することはできません。

#### *URL*

サーバーが別のサーバーに送信する `URL` 要求を指定します。この要求に対する応答は、それがサーバーからのものでないということは一切示さずに、元の要求側に送信されます。

`URL` には、プロトコル指定と、要求の送信先であるサーバーの名前が含まれていなければなりません。パス名またはファイル名を入れることもできます。

`request_template` でワイルドカードを使用した場合は、`URL` のパス名またはファイル名にもワイルドカードを使用することができます。元の要求の `request_template` のワイルドカードに一致する部分が、`URL` のワイルドカードの代わりに挿入されます。

#### `[server_IP_address | host_name]`

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) またはホスト名 (例えば、hostA.bcd.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを指定しない場合、サーバーは、要求を受信する IP アドレスや `URL` のホスト名に関係なく、このディレクティブを使用してすべての要求を処理します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

### 例

- 以下の例において、サーバーは `/chief/stuff/` で始まるすべての要求を `www.other.org` サーバーの `wahoo` ディレクトリーに送信します。

```
Redirect /chief/stuff/* http://www.other.org/wahoo/*
```

- 以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーは、/stuff/ で始まる要求を受信すると、要求が入ってきたネットワーク接続の IP アドレスに基づいて、その要求を別のサーバーにリダイレクトします。240.146.167.72 に入ってくる要求の場合、サーバーはその要求を

www.chief.org サーバーの wahoo ディレクトリーに送ります。アドレスが 0.83.100.45 の接続に入ってくる要求の場合には、サーバーはその要求を www.dawg.com サーバーの pound ディレクトリーに送ります。

```
Redirect /stuff/* http://www.chief.org/wahoo/* 240.146.167.72
Redirect /stuff/* http://www.dawg.com/pound/* 0.83.100.45
```

- 以下の例では、オプションの IP アドレス・パラメーターを使用しています。サーバーは、/stuff/ で始まる要求を受信すると、URL のホスト名に基づいて、その要求を別のサーバーへリダイレクトします。hostA に送信された要求の場合、サーバーはその要求を www.chief.org サーバーの wahoo ディレクトリーに送信します。hostB に送信された要求の場合、サーバーはその要求を www.dawg.com サーバーの pound ディレクトリーに送信します。

```
Redirect /stuff/* http://www.chief.org/wahoo/* hostA.bcd.com
Redirect /stuff/* http://www.dawg.com/pound/* hostB.bcd.com
```

## デフォルト

なし

## RegisterCacheIdTransformer - Cookie ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる

このディレクティブを使用し、Caching Proxy によって、Cookie ヘッダーを基にしたリソースの複数のバリエーション (URI) をキャッシュに入れることができます。

注: Cookie がクライアント・ブラウザで使用不可の場合、複数のクライアントが同じキャッシュ・オブジェクトにアクセスできます。

詳しくは、306 ページの『SupportVaryHeader - HTTP Vary ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる』を参照してください。

## 形式

RegisterCacheIdTransformer Cookie *cookie-name*

*cookie-name* は、クライアントの要求の Cookie ヘッダーにおける名前です。

## 例

RegisterCacheIdTransformer Cookie Usergroup

SupportVaryHeader と結合しているこのディレクティブの使用例については、306 ページの『SupportVaryHeader - HTTP Vary ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる』を参照してください。

## デフォルト

なし

## ReversePass - 自動的にリダイレクトされた要求をインターセプトする

これは、リバース・プロキシー構成にのみ適用されます。

ReversePass マッピング・ディレクティブは、サーバー応答ストリームを検査して、自動リダイレクトの結果として再書き込みされる要求を検出します。通常、サーバーが 3xx クラス (例えば、301 (moved permanently)、または 303 (see other)) で HTTP コードを戻す時に、サーバーは、要求側クライアントが以後の要求を正しい URL および IP アドレスに送信するように指示する応答を付けて、メッセージを送信します。リバース・プロキシー・セットアップの場合、起点サーバーからのリダイレクト・メッセージは、以後の要求に関してクライアント・ブラウザにプロキシー・サーバーをバイパスさせることが可能です。クライアントが起点サーバーと直接に連絡を取ることを避けるために、ReversePass ディレクティブを使用して、起点サーバーに特定して送信される要求をインターセプトします。

要求ストリームを処理する他のマッピング・ディレクティブとは異なり、ReversePass は、そのテンプレートを応答ストリームと突き合わせます。応答ストリームとは、プロキシー・サーバーが起点サーバーから取得し、クライアントへ送信する応答のことです。

### 形式

ReversePass *rewritten\_URL proxy\_URL* [*host:port*]

*host:port* オプションは、プロキシーが、バックエンド・サーバーのホスト名およびポートに基づいた異なる ReversePass 規則を適用することを可能にします。

### 例

- 次のステートメント例では、起点サーバーへの直接の要求が防止されます。

```
ReversePass http://backend.company.com:9080/* http://edge.company.com/*
```

ポート 9080 は エッジでのアプリケーション・サービスのデフォルトのポートです。このタイプの要求は、起点アプリケーション・サーバーが 3xx コードをクライアントに戻す場合に生成されます。

- 次のステートメント例では、エッジ・アプリケーション・サーバーから 301 コードでリダイレクトされる要求が検出されます。

```
ReversePass http://edge.company.com:9080/* http://edge.company.com/*
```

注: *proxy\_URL* パターンのコンテンツは、バックエンド・サーバーが送信したロケーション・ヘッダーに対して、ワイルドカード (\*) の前の部分に関しては完全に一致する必要があります。そうでないと、ディレクティブは失敗します。

### デフォルト

なし

## RewriteSetCookieDomain - 再書き込みする必要のあるドメイン・パターンを指定する

これは、リバース・プロキシー構成にのみ適用されます。

このディレクティブは、再書き込みする必要のあるドメイン・パターンを指定するために使用します。このディレクティブは、ドメイン *domain\_pattern1* から *domain\_pattern2* に変換します。

## 形式

`RewriteSetCookieDomain domain_pattern1 domain_pattern2`

## 例

`RewriteSetCookieDomain .internal.com .external.com`

## デフォルト

なし

## 関連ディレクティブ

- 246 ページの『JunctionRewriteSetCookiePath - JunctionRewrite プラグインとの併用時に Set-Cookie ヘッダーのパス・オプションを再書き込みする』

# RTSPEnable - RTSP リダイレクトを使用可能にする

これは、リバース・プロキシ構成にのみ適用されます。

このディレクティブは、RTSP リダイレクトを使用可能または使用不可にします。オプションは、on または off です。

## 形式

`RTSPEnable {on | off}`

## 例

`RTSPEnable on`

## デフォルト

なし

# rtsp\_proxy\_server - リダイレクト用のサーバーを指定する

これは、リバース・プロキシ構成にのみ適用されます。

このディレクティブは、リダイレクトされる要求を受信する RTSP プロキシ・サーバーを指定するときに使用します。タイプの異なるストリームに合わせて各種サーバーを指定できます。このディレクティブの形式は、次のとおりです。

`rtsp_proxy_server server dns address[:port] default rank [list of mime types]`

## 例

<code>rtsp_proxy_server</code>	<code>rproxy.mycompany.com:554</code>	<code>1</code>
<code>rtsp_proxy_server</code>	<code>fw1.mycompany.com:554</code>	<code>2</code>
<code>rtsp_proxy_server</code>	<code>fw1.mycompany.com:555</code>	<code>3</code>
<code>rtsp_proxy_server</code>	<code>fw2.mycompany.com:557</code>	<code>4</code>

## デフォルト

なし



## rtsp\_proxy\_threshold - キャッシュへのリダイレクトまでの要求の数を指定する

これは、リバース・プロキシー構成にのみ適用されます。

このディレクティブは、起点サーバーではなく、プロキシー・サーバーに RTSP 要求がリダイレクトされるまでに受信される要求の数を指定します。RealNetworks プロキシーは、最初の要求のストリームをキャッシュに格納し、キャッシングには、当初、ストリームの受信の 2 倍の帯域幅となります。しきい値 2 以上の値を指定すると、1 回だけの要求はキャッシュに格納されません。このディレクティブの形式は、次のとおりです。

```
rtsp_proxy_threshold number_of_hits
```

### 例

```
rtsp_proxy_threshold 5
```

### デフォルト

なし

## rtsp\_url\_list\_size - プロキシー・メモリー中の URL の数を指定する

これは、リバース・プロキシー構成にのみ適用されます。

このディレクティブは、リダイレクトのためにメモリーに保持される固有の URL の数を指定します。プロキシーは、特定の URL が以前に検索されているかどうかを判別するために、このリストを参照します。リストのサイズが大きいと、直前の要求を受信したのと同じプロキシー・サーバーに後続の要求をプロキシー・サーバーが送信できる可能性が高まります。しかし、リストの項目ごとに約 16 バイトのメモリーが消費されます。

### 形式

```
rtsp_url_list_size size_of_list
```

### 例

```
rtsp_url_list_size 8192
```

### デフォルト

なし

## RuleCaseSense - 大/小文字を区別しないアプリケーション URL からの要求をマップする

デフォルトでは、ibmproxy.conf ファイルに定義されているルールに対して Caching Proxy が要求をマップする際、マッチング・プロセスで大/小文字を区別します。ただし、一部のアプリケーション URL は、大/小文字を区別していません。これらの要求を正しく処理するために、RuleCaseSense ディレクティブが用意されています。このディレクティブを off に設定すると、プロキシーは大/小文字の区別をせずに、要求のマッチングを行います。

注: このディレクティブはグローバル・ディレクティブであり、定義されたすべてのマッピング・ルールに適用されます。

## 形式

`RuleCaseSense {on | off}`

## デフォルト

`RuleCaseSense on`

## ScriptTimeout - スクリプトのタイムアウト設定値を指定する

このディレクティブは、サーバーによって開始された CGI プログラムが終了するまでの時間を設定します。時間が期限切れになると、サーバーはプログラムを終了します。Linux および UNIX プラットフォームでは、これは KILL シグナルによって実行されます。

時間の値は、時間 (hours)、分 (minutes または mins)、および秒 (seconds または secs) を任意に組み合わせて入力します。

## 形式

`ScriptTimeout timeout`

## デフォルト

`ScriptTimeout 5 minutes`

## SendHTTP10Outbound - プロキシ要求のプロトコル・バージョンを指定する

このディレクティブは、Caching Proxy からダウンストリーム・サーバーへ送信される要求は HTTP バージョン 1.0 プロトコルを使用する必要があることを指定するときに使用します。(ダウンストリーム・サーバーとは、要求を処理するプロキシのチェーン内の別のプロキシ・サーバーあるいは起点サーバーです。)

このディレクティブが使用されると、Caching Proxy は HTTP 1.0 を要求行の中のプロトコルとして識別します。ほとんどの HTTP 1.0 サーバーでサポートされている Cache-control ヘッダーのような、HTTP 1.0 に特有の機能、およびある特定の HTTP 1.1 機能は、ダウンストリーム・サーバーへ送られます。このディレクティブは、HTTP 1.1 要求を正しく処理しないダウンストリーム・サーバーがあった場合に使用してください。

SendHTTP10Outbound ディレクティブが指定されていない 場合には、Caching Proxy は HTTP 1.1 を要求行の中のプロトコルとして識別します。この要求では、持続接続機能のような HTTP 1.1 機能も使用することができます。

## 形式

`SendHTTP10Outbound url_pattern`

## 例

このディレクティブは、複数回指定することができます。例えば、次のとおりです。

```
SendHTTP10Outbound http://www.hosta.com/*  
SendHTTP10Outbound http://www.hostb.com/*
```

逆方向の互換性のために、前の `SendHTTP10Outbound` の構文は、以下のように取り扱われることになりました。

- `SendHTTP10Outbound on` は `SendHTTP10Outbound *` が指定されたものとして取り扱われます。
- `SendHTTP10Outbound off` は無視されます。

注: `SendHTTP10Outbound off` と `SendHTTP10Outbound url_pattern>` の両方が指定されている場合には、`SendHTTP10Outbound off` は無視されますが、警告メッセージが出されます。

## デフォルト

なし

## SendRevProxyName - HOST ヘッダーの Caching Proxy ホスト名を指定する

これは、リバース・プロキシー構成にのみ適用されます。

リバース・プロキシーとして機能する場合、Caching Proxy はクライアントから HTTP 要求を受信し、その要求を起点サーバーに送信します。デフォルトで、Caching Proxy は、起点サーバーに送信される要求の HOST ヘッダーに起点サーバーのホスト名を書き込みます。SendRevProxyName ディレクティブが yes に設定された場合、Caching Proxy は、代わりに自身のホスト名を HOST ヘッダーに書き込みます。バックエンド・サーバーから別のバックエンド・サーバーへ要求がリダイレクトされる場合であっても、起点サーバーへの要求が常にプロキシー・サーバーから来るように見せることが可能なので、このディレクティブはバックエンド・サーバー用の特別な構成を使用可能にするために使用されます。

このディレクティブは、次の点で ReversePass マッピング・ディレクティブとは異なります。ReversePass ディレクティブは指定された構文によって要求をインターセプトし、ユーザーが指定した別の要求コンテンツで置き換えます。一方、SendRevProxyName ディレクティブは起点サーバーのホスト名を Caching Proxy のホスト名に置き換えるためにのみ設定されます。このディレクティブは、エッジでのアプリケーション・サービスの構成には利用できません。

## 形式

```
SendRevProxyName {yes | no}
```

## ServerConnGCRun - ガーベッジ・コレクション・スレッドを実行する間隔を指定する

このディレクティブは、ガーベッジ・コレクション・スレッドでサーバー接続がタイムアウト (ServerConnTimeout ディレクティブによる設定) になったかどうかを検査する間隔を設定します。このディレクティブは、ServerConnPool ディレクティブを On に設定した場合にのみ使用してください。

## 形式

`ServerConnGCRun time_interval`

## 例

`ServerConnGCRun 2 minutes`

## デフォルト

`ServerConnGCRun 2 minutes`

## ServerConnPool - 起点サーバーへの接続のプールを指定する

このディレクティブによってプロキシでは、起点サーバーへのその発信接続を 1 つにまとめてプールすることができます。このディレクティブを `On` に設定すると、パフォーマンスが向上し、持続接続が可能な起点サーバーの利点を活用できます。また、`ServerConnTimeout` ディレクティブを通じて未使用接続をどれだけ長い期間保持するかを指定することもできます。

注: このディレクティブは、制御された環境で最良の使用可能状態が得られます。フォワード・プロキシ状態あるいは起点サーバーが HTTP 1.1 に準拠していない状態では、パフォーマンスを損なう恐れがあります。

## 形式

`ServerConnPool {on | off}`

## デフォルト

`ServerConnPool off`

## ServerConnTimeout - 最大活動停止中期間を指定する

このディレクティブは、接続が取り消されるまでネットワーク活動をしないでいられる時間を制限するときに使用します。このディレクティブは、`ServerConnPool` ディレクティブを `On` に設定した場合にのみ使用してください。

## 形式

`ServerConnTimeout time-spec`

## 例

`ServerConnTimeout 30 seconds`

## デフォルト

`ServerConnTimeout 10 seconds`

## ServerInit - サーバー初期化ステップをカスタマイズする

このディレクティブは、その初期化ルーチンの実行中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、クライアント要求が読み込まれる前、およびサーバーが再始動されるたびに実行されます。

GoServe モジュールを PreExit または Service ステップで使用する場合は、ここで gosclone モジュールを呼び出す必要があります。

## 形式

`ServerInit /path/file:function_name [initialization_string]`

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

*initialization\_string*

オプションで、アプリケーション関数に渡すテキスト・ストリングを指定します。

## 例

`ServerInit /ics/api/bin/icsext05.so:svr_init`

## デフォルト

なし

## ServerRoot - サーバー・プログラムのインストール先ディレクトリーを指定する

このディレクティブは、サーバー・プログラムのインストール先ディレクトリー (サーバーの現行作業ディレクトリー) を指定するときに使用します。ディレクティブをログに記録する際には、相対パス名が使用されているときのデフォルト・ルートとしてこの現行作業ディレクトリーが使用されます。

Windows システムでは、このディレクトリーはインストール時に識別されます。

## 形式

`ServerRoot directory_path`

## デフォルト

- **Linux** および **UNIX** システム: `ServerRoot /opt/ibm/edge/cp/server_root/`
- **Windows** システム: `C:%Program Files%IBM%edge%cp%bin%`

注: デフォルトを変更しても構いませんが、変更しても、サーバーが要求を処理する方法には影響を与えません。

注: PASS および EXEC 規則は、このディレクトリーとは独立させることができます。

## ServerTerm - サーバー終了ステップをカスタマイズする

このディレクティブは、サーバー終了ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、正常シャットダウンが行われたとき、およびサーバーが再始動されるごとに実行されます。このディレクティブを使用すると、PreExit アプリケーション関数によって割り振られたリソースを解放することができます。

## 形式

ServerTerm */path/file:function\_name*

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

## 例

ServerTerm */ics/api/bin/icsext05.so:shut\_down*

## デフォルト

なし

## Service - サービス・ステップをカスタマイズする

このディレクティブは、サービス・ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、クライアント要求に対応します。例えば、ファイルを送信したり、CGI プログラムを実行します。

このディレクティブにはデフォルトはありません。要求が Service 規則と一致する (Service ディレクティブで指定されたアプリケーション関数が実行される) が、関数が HTTP\_NOACTION を戻した場合は、サーバーはエラーを生成し、要求は失敗します。

## 形式

Service *request\_template/path/file:function\_name*  
[*server\_IP\_address* | *host\_name*]

*request\_template*

アプリケーション関数が呼び出されるかどうかをさらに判別する要求のためのテンプレートを指定します。この指定には、プロトコル、ドメイン、およびホストを組み込むことができ、前にスラッシュ (/) を付けたり、ワイルドカードとしてアスタリスク (\*) を使用することができます。例えば、*/front\_page.html*、*http://www.ics.raleigh.ibm.com*、*/pub\**、*/\**、および *\** はすべて有効です。

*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でのアプリケーション関数名を指定します。

[*Server\_IP\_address* | *host\_name*]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターによって、アプリケーション関数が特定の IP アドレスに入ってきた要求についてだけ呼び出されるか、あるいは特定のホストについてだけ呼び出されるかが決定されます。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

## 例

```
Service /index.html /ics/api/bin/icsext05.so:serve_req
Service /cgi-bin/hexcalc* /ics/api/calculator:HEXcalc*
```

注: 絶対パス変換 (*query\_string* を含む) が必要な場合には、2 番目の例に示されているように、*request\_template* と *path/file:function\_name* の両方にアスタリスク (\*) が必要です。

## デフォルト

なし

## SignificantURLTerminator - URL 要求の終了コードを指定する

このディレクティブは、URL 要求の終了コードを指定するために使用します。要求の中で終了コードを使用すると、Caching Proxy は、要求の処理時および結果がすでにキャッシングされたかどうかの評価時に、終了コードの前の文字だけを評価します。複数の終了コードが定義されている場合には、Caching Proxy は着信 URL を ibmproxy.conf ファイルに定義されている順序での終了コードと比較します。

## 形式

SignificantURLTerminator *terminating\_string*

## 例

```
SignificantURLTerminator &.
```

この例では、以下の 2 つの要求は同一として処理されます。

```
http://www.exampleURL.com/tx.asp?id=0200&. ;x=004;y=001
http://www.exampleURL.com/tx.asp?id=0200&. ;x=127;y=034
```

## デフォルト

なし

## SMTPServer (Windows のみ) - sendmail ルーチン用に SMTP サーバーを設定する

このディレクティブは、Caching Proxy for Windows 内の内部 sendmail ルーチンによって使用される SMTP サーバーを設定するために使用します。313 ページの『WebMasterEMail - 選ばれたサーバー報告書を受け取るための電子メール・アドレスを設定する』および 313 ページの『WebMasterSocksServer (Windows のみ) - sendmail ルーチン用に Socks サーバーを設定する』で説明する 2 つのディレクティブも、このルーチンのために設定しなければなりません。

## 形式

SMTPServer *IP address or hostname of SMTP server*

## 例

```
SMTPServer mybox.com
```

## デフォルト

なし



## SNMP - SNMP サポートを使用可能および使用不可にする

このディレクティブを使用して、SNMP サポートを使用可能または使用不可にします。

### 形式

SNMP {on | off}

### デフォルト

SNMP off

## SNMPCommunity - SNMP のセキュリティー・パスワード・を指定する

このディレクティブは、Web サーバー分散プロトコル・インターフェース (DPI) サブエージェントと SNMP エージェント間のパスワードを定義するときに使用します。SNMP コミュニティー名は、サーバーの指定されたコミュニティについて SNMP がモニターするパフォーマンス変数をユーザーが表示するのを許可します。システム管理者は、パスワードが入力された場合に表示できる変数を定義します。SNMP コミュニティー名を変更するときは、`/etc/snmpd.conf` という名前のファイルに指定されたコミュニティ名も必ず変更してください。

### 形式

SNMPCommunity *name*

### デフォルト

SNMPCommunity public

## SSLCaching - セキュア要求のキャッシュを使用可能にする

このディレクティブは、リバース・プロキシの使用時にセキュア要求の内容をキャッシュに入れるときに使用します。このディレクティブは、プロキシ・サーバーへのすべての接続 (クライアント接続とバックエンド・コンテンツ接続の両方) のためのキャッシュを構成します。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

### 形式

SSLCaching {on | off}

### デフォルト

SSLCaching off

## SSLCertificate - 証明書の鍵ラベルを指定する

このディレクティブは、Caching Proxy が独自の SSL 証明書を発行する複数ドメインの単一リバース・プロキシとして活動しているときに、クライアントにどの証明書を送るかを決定し、プロキシ・サーバーにクライアント認証のためにクライアント側の PKI 証明書を検索するかどうかをプロキシが指示できる鍵ラベルを指定するために使用します。

Caching Proxy は、SSLCertificate ディレクティブを使用して、認証局 (CA) 発行の証明書と、自己割り当ての証明書を区別することができます。ただし、このディレクティブを使用して、なんらかの CA 発行の証明書を受け入れる (ClientAuthRequired オプション) ことにより、無効なユーザーにもプロキシ・サーバーへのアクセスを獲得することを許可できます。SSLCertificate ディレクティブで ClientAuthRequired オプションを使用する場合、どの正当なユーザーが SSL チャネルにアクセスできるかを判別するために、論理式オプションを使用することができます。

追加の論理式を SSLCertificate ディレクティブに追加すると、Caching Proxy は、クライアント証明書から値を取り出して、その論理式を計算します。クライアント証明書の値でその式が満たされた場合、Caching Proxy は、SSL 接続の使用についてクライアントを認可します。満たされない場合、接続はシャットダウンされクローズされます。

## 形式

SSLCertificate *serverIP/hostname CertificateLabel*  
[NoClientAuth | ClientAuthRequired *logic-expression*]

*serverIP/hostname*

SSL 要求の送信先のサーバーの IP アドレス (例えば、204.146.167.72) を指定するか、あるいはホスト名 (例えば、hostA.raleigh.ibm.com) を指定することができます。

*CertificateLabel*

指定した IP アドレスまたはホスト名に送信される SSL 要求についてクライアント認証が必要な場合に使用する証明書の名前。

[NoClientAuth | ClientAuthRequired *logic-expression*]

クライアント側の PKI 証明書を検索するかどうかの、プロキシ・サーバーに対する指示。

この論理式オプションが有効なのは、ClientAuthRequired オプションと共に使用された場合のみです。追加の論理式を SSLCertificate ディレクティブに追加すると、Caching Proxy は、クライアント証明書から値を取り出して、その論理式を計算します。クライアント証明書の値でその式が満たされた場合、Caching Proxy は、SSL 接続の使用についてクライアントを認可します。満たされない場合、接続はシャットダウンされクローズされます。

- この論理式で利用できる属性名は以下のとおりです:
  - IST、ICN、IOU、IC、IL、IO、IE、ST、CN、OU、C、L、O、E。
  - 属性名は、クライアント証明書の次のフィールドにマップされます:
    - IssuerStateOrProvince (IST) IssuerCommonName (ICN) IssuerOrgUnit (IOU)
    - IssuerCountry (IC) IssuerLocality (IL) IssuerOrg (IO) IssuerEmail (IE)
    - StateOrProvince (ST) CommonName (CN) OrgUnit (OU) Country (C)
    - Locality (L) Org (O) Email (E)
- 属性名の値は、引用符で区切る必要があります。
- 有効な論理演算子は以下のとおりです: && (AND)、|| (OR)、! (NOT)、= (EQUAL)

## 例

```
SSLCertificate www.abc.com ABCCert
SSLCertificate 204.146.167.72 intABCCert
SSLCertificate www.xyz.com XYZCert ClientAuthRequired
SSLCertificate www.xyz.com XYZCert ClientAuthRequired
    CN="valid.user.common.name.pattern" && (L="accepted.location.pattern" ||
    C!="not.valid.country.pattern")
```

## デフォルト

なし

## SSLCryptoCard - インストール済み暗号カードを指定する

これは、リバース・プロキシ構成にのみ適用されます。

このディレクティブは、インストール済みの暗号カードがあることをプロキシ・サーバーに知らせ、そのカードを指定するために使用します。

AIX で、IBM 4960 PCI 暗号アクセラレーター・カードをサポートするためには、270 ページの『PKCS11DefaultCert、PKCS11DriverPath、 PKCS11TokenPassword - IBM 4960 PCI 暗号アクセラレーター・カードをサポートする (AIX のみ)』を参照してください。

## 形式

```
SSLCryptoCard {rainbowcs | nciphernfast} {on | off}
```

## 例

```
SSLCryptoCard rainbowcs on
```

## デフォルト

なし

## SSLEnable - セキュア要求をポート 443 で listen するように指定する

このディレクティブは、Caching Proxy がセキュア要求をポート 443 で listen するように指定するために使用します。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

## 形式

```
SSLEnable {on | off}
```

## デフォルト

```
SSLEnable off
```

## SSLForwardPort - HTTP SSL のアップグレードのためにアドレス指定するポートを指定する

このディレクティブは、Caching Proxy が SSL をインプリメントすることによって HTTPS 要求にアップグレードする HTTP 要求をアドレス指定するポートを指定するときに使用します。メイン HTTP ポート 80 または メイン SSL ポート 443 以外のポートを指定します。

### 形式

SSLForwardPort *port number*

### 例

SSLForwardPort 8888

### デフォルト

なし

## SSLOnly - HTTP 要求のリスナー・スレッドを使用不可にする

このディレクティブは、SSL (通常はポート 443) が使用可能なときに、標準 HTTP 要求 (通常はポート 80 および 8080) に対するリスナー・スレッドを使用不可にするために使用します。

### 形式

SSLOnly {on | off}

### デフォルト

SSLOnly off

## SSLPort - デフォルト以外の HTTPS listen ポートを指定する

このディレクティブを使用して、ibmproxy のデフォルト HTTPS ポート 443 以外の HTTPS listen ポートを指定します。

注: ibmproxy は、それぞれのインスタンスごとに 1 つの HTTPS ポートをサポートしますので、ディレクティブを複数の HTTPS ポートの指定に使用しないでください。複数の HTTPS ポートをサポートするには、別々の ibmproxy.conf ファイルを持つ複数の ibmproxy インスタンスを開始する必要があります。

### 形式

SSLPort *port value*

ここで、*port value* は 0 より大きい整数値。さらに、*port value* がオペレーティング・システムに許可されており、他のアプリケーションに使用されていないことが必要です。

### 例

SSLPort 8443

## デフォルト

443

## SSLTunneling - SSL トンネリングを使用可能にする

これは、フォワード・プロキシー構成にのみ適用されます。

このディレクティブを `on` に設定すると、宛先サーバー上の任意のポートへの SSL トンネリングが許可されます。このディレクティブを `off` に設定すると、Proxy 規則で指定されているポートへのみの SSL トンネリングが許可されます。SSL トンネリングに対する Proxy 規則がなく、SSLTunneling ディレクティブが `off` の場合は、SSL トンネリングは許可されません。SSLTunneling ディレクティブが `on` の場合、Enable ディレクティブを使用して、CONNECT メソッドも使用可能にしなければなりません。

Caching Proxy をフォワード・プロキシーとして使用する場合、このディレクティブを使用可能にする必要があります。ただし、Caching Proxy をリバース・プロキシーとして使用する場合は、このディレクティブを使用不可にしておく（デフォルト）と、SSL トンネリングのぜい弱性に対するアタックから保護されます。

詳しくは、127 ページの『SSL トンネリング』を参照してください。

注: Proxy ディレクティブを使用して、宛先ホスト上の特定のポートへの SSL トンネリングを使用可能にしてください。

### 形式

SSLTunneling {on | off}

### デフォルト

SSLTunneling off

## SSLVersion - SSL のバージョンを指定する

このディレクティブは、使用する SSL のバージョン (V2、V3、またはすべてのバージョン) を指定するときに使用します。SSL バージョン 3 をサポートできないサーバーを使用している場合は、このディレクティブを V2 に設定してください。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

### 形式

SSLVersion {SSLV2 | SSLV3 | all}

### デフォルト

SSLVersion SSLV3

## SSLV2Timeout - SSLV2 セッションが有効期限切れになるまでの待ち時間を指定する

このディレクティブは、SSL バージョン 2 のセッションがセッション有効期限切れになるまで活動なしに待機する長さ (秒単位) を指定するときに使用します。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

### 形式

SSLV2Timeout *seconds*

ここで、*seconds* は 0 と 100 の間の値を表します。

### デフォルト

SSLV2Timeout 100

## SSLV3Timeout - SSLV3 セッションが有効期限切れになるまでの待ち時間を指定する

このディレクティブは、SSL バージョン 3 のセッションがセッション有効期限切れになるまで活動なしに待機する長さ (秒単位) を指定するときに使用します。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

### 形式

SSLV3Timeout *seconds*

ここで、*seconds* は 1 秒と 86400 秒 (秒数による 1 日) の間の値です。

### デフォルト

SSLV3Timeout 100

## SuffixCaseSense - 接尾部定義で大/小文字の区別を行うかどうかを指定する

このディレクティブは、ファイル接尾部を AddClient、AddCharSet、AddType、AddEncoding、および、AddLanguage ディレクティブの接尾部パターンと比較する場合に、サーバーに大文字小文字を区別させたいかどうかを指定するときに使用します。デフォルトでは、サーバーは大文字小文字を区別しません。

### 形式

SuffixCaseSense {on | Off}

### デフォルト

SuffixCaseSense Off

## SupportVaryHeader - HTTP Vary ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる

このディレクティブを使用し、Caching Proxy によって、HTTP Vary ヘッダーを基にしたリソースの複数のバリエーション (URI) をキャッシュに入れることができるようになります。

SupportVaryHeader ディレクティブが使用可能になると、プロキシは URI を基にしたキャッシュ ID と、クライアント要求の選択済みヘッダー値を形成します。

選択済みヘッダーの名前は、サーバーからの事前応答で送信された Vary ヘッダーで指定されます。サーバーがリソースに対する選択済みヘッダー名のセットを変更する場合、リソースに対する以前のキャッシュ・オブジェクトはすべて、プロキシのキャッシュから除去されます。

このディレクティブは、RegisterCacheIdTransformer ディレクティブ (291 ページの『RegisterCacheIdTransformer - Cookie ヘッダーを基にしたリソースの、複数のバリエーションをキャッシュに入れる』) と一緒に使用できます。

両方のディレクティブを使用すると、プロキシは、サーバーおよびクライアントの要求ヘッダーから、Vary ヘッダーを基にした内部キャッシュ ID 変換プログラムを作成します。このようにして、プロキシは、異なる要求と応答のペアに対して (要求された URI が同じであっても) 固有のキャッシュ ID を生成することができます。

同じ URI のキャッシュ・オブジェクトは、それぞれ固有のデフォルト存続時間を持っており、それは要求/応答、または他の構成設定値の「Expire」および「Cache-Control」ヘッダーに依存します。Dynacache プラグインが使用される場合、同じ URI に関連づけられた複数のプレゼンテーションはすべて、プロキシのキャッシュ内で共に無効になります。

## 形式

```
SupportVaryHeader {on | off}
```

## 例

この例として、以下のディレクティブが `ibmproxy.conf` で使用可能にされ、構成されています。次のとおりです。

```
SupportVaryHeader on
RegisterCacheIdTransformer Cookie UserGroup
```

クライアントの Guest は、次のようにしてプロキシ・サーバーにアクセスします。

```
URI [<code>] http://www.dot.com/group.jpg [</code>]
```

そして要求/応答は、次のようになります。

```
GET /group.jpg HTTP/1.1
Host: www.dot.com
Cookie: UserGroup=Guest
Accept-Language: en_US
```

```
HTTP/1.1 200
Server: my-server
Vary: Accept-Language
.....
```

次に、クライアントの Admin は、同じ URI でプロキシ・サーバーにアクセスします。

```
http://www.dot.com/group.jpg
```

そして要求/応答は、次のようになります。



```
GET /group.jpg HTTP/1.1
Host: www.dot.com
Cookie: UserGroup=Admin
Accept-Language: fr_FR
```

```
HTTP/1.1 200
Server: my-server
Vary: Accept-Language
.....
```

結果として、応答がキャッシュ可能である場合、プロキシ・サーバーは、2 つの異なるキャッシュ ID を生成します。

1. CacheID(URI, "Guest", "en\_US")
2. CacheID(URI, "Admin", "fr\_FR")

プロキシ・サーバーは、キャッシュにあるサーバーからの応答の 2 つの異なるバリエーションを保管しています。その後、クライアントのいずれかが、言語プリファレンスとユーザー・グループ値を組み合わせて、リソース (.../group.jpg) を要求する時、プロキシ・サーバーは、キャッシュからリソースの適切なバリエーションを検索して、それをサービス供給します。

## デフォルト

SupportVaryHeader off

## TLSV1Enable - トランスポート層セキュア・プロトコルを使用可能にする

このディレクティブは、SSL 接続で TLS バージョン 1 プロトコルを使用可能にするときに使用します。このディレクティブが on に設定されたら、SSL 接続は最初に TLS プロトコルを検査し、次に SSLv3 プロトコル、最後に SSLv2 プロトコルを検査します。

注: このディレクティブは、Internet Explorer やその他のブラウザで機能しますが、Netscape では機能しません。(Netscape は、Caching Proxy 用の使用が推奨されるブラウザではありません。)

## 形式

TLSV1Enable {on | off}

## 例

TLSV1Enable on

## 構成ファイルの初期設定値

なし

## Transmogriifier - データ操作ステップをカスタマイズする

このディレクティブは、データ操作ステップ中にサーバーが呼び出すカスタマイズ済みアプリケーション関数を指定するときに使用します。このコードは、3 つのアプリケーション関数を提供します。

- データ処理の前に初期化を実行する *open* 関数
- データ処理を行う *write* 関数

- クリーンアップ活動を実行する *close* 関数
- 発生した問題を提供する *error* 関数

サーバーの各インスタンスごとに複数の Transmogrifier をアクティブにすることができます。

## 形式

Transmogrifier */path/file:function\_name:function\_name:function\_name*  
*/path/file*

拡張子を含む、コンパイル済みプログラムの完全修飾ファイル名を指定します。

*function\_name*

プログラム内でアプリケーション機能に付与した名前を指定します。

open、write、および close という機能の名前を提供する必要があります。

## 例

Transmogrifier /ics/bin/icsext05.so:open\_data:write\_data:close\_data

## デフォルト

なし

## TransmogrifiedWarning - 警告メッセージをクライアントへ送信する

このディレクティブは、データに関する以下の内容を通知するメッセージをクライアントへ送信するために使用します。

## 形式

transmogrifiedwaning {yes|no}

## デフォルト

Yes

## TransparentProxy - Linux で透過プロキシーを使用可能にする

これは、フォワード・プロキシー構成にのみ適用されます。

**Linux** システムの場合にのみ、このディレクティブを使用すると、サーバーが透過プロキシー・サーバーとして稼働できるかどうかを指定できます。

TransparentProxy ディレクティブを on に設定すると、BindSpecific ディレクティブは無視されて、デフォルトで off となります。ほとんどの HTTP トラフィックがポート 80 を流れるため、ポート 80 を構成ポートの 1 つにすることをお勧めします。

## 形式

TransparentProxy {on | off}  
 Port 80

## デフォルト

TransparentProxy off

IPCHAIN ファイアウォールを使用する場合、このディレクティブを使用可能にするだけで、透過プロキシを正常に構成できます。IPTABLES ファイアウォールを使用する場合は、IPTABLES ファイアウォール・ルールを手動で追加する必要があります。

IPTABLES ファイアウォールを使用していて、TransparentProxy ディレクティブを使用可能に設定した場合、**プロキシ・サーバーを開始する前に**、次のコマンドを実行して、ファイアウォール・ルールを IPTABLES に追加してください。

```
iptables -t nat -A PREROUTING -i your-network-interface -p tcp --dport 80 -j  
REDIRECT --to-port ibmproxy-listening-port
```

ファイアウォールとプロキシ・サーバーが同じボックス上にあると想定すると、このルールでは、IPTABLES ファイアウォールに対し、ポート 80 を宛先とするすべての TCP トラフィックを、ローカル・プロキシ listen ポートにリダイレクトするよう指示します。このルールを IPTABLES 構成に追加することもできます。これにより、システム再始動時に、このルールが自動的に読み込まれます。

**透過プロキシの開始後に**、Caching Proxy サーバーを停止したい場合は、ルートとして以下のコマンドを発行することも必要です。

```
ibmproxy -unload
```

Linux システムでは、このコマンドでリダイレクト・ファイアウォール・ルールを除去します。サーバーを停止した後に、このコマンドを出さないでいると、ユーザーのマシンに宛先指定されたものでない要求を受信してしまうことになります。

## UpdateProxy - キャッシュ宛先を指定する

このディレクティブは、キャッシュ・エージェントがどのプロキシ・サーバーを更新するかを指定するときに使用します。これは、キャッシュ・エージェントが稼働しているローカル・プロキシ・サーバー以外のプロキシ・サーバーをキャッシュ・エージェントが更新する必要がある場合に必要とされます。オプションで、ポートを指定することができます。

**注:** Linux および UNIX プラットフォームでは、キャッシュ・エージェントを使用する場合にこのディレクティブが必要です。プロキシに使用しているマシンが 1 つだけの場合は、ホスト名を指定してください。

キャッシュ・エージェントは別のサーバー上のキャッシュを更新できますが、そのマシンからのキャッシュ・アクセス・ログを検索することはできません。したがって、UpdateProxy ディレクティブがローカル・ホスト以外のホストを指定している場合、LoadTopCached ディレクティブは無視されます。

## 形式

UpdateProxy *fully\_qualified\_host\_name\_of\_proxy\_server*

## 例

```
UpdateProxy proxy15.ibm.com:1080
```

## デフォルト

なし

## UserId - デフォルトのユーザー ID を指定する

このディレクティブは、サーバーがファイルにアクセスする前に変更する先のユーザー名または番号を指定するときに使用します。

このディレクティブを変更した場合には、サーバーを手動で停止してから再始動しなければ、変更が有効になりません。再始動しただけでは、サーバーは変更を認識しません。（15 ページの『第 5 章 Caching Proxy の開始および停止』を参照してください。）

**注:** サーバーのユーザー ID、グループ ID、あるいはログ・ディレクトリー・パスに対するデフォルトを変更する場合は、新規ディレクトリーを作成し、その許可および所有権を更新します。サーバーが情報をユーザー定義のログ・ディレクトリーに書き込むことができるようにするには、そのディレクトリーの許可を 755 として設定し、ユーザー定義のサーバー・ユーザー ID を所有者として設定します。例えば、サーバーのユーザー ID をデフォルトから jdoe に変換し、デフォルト・ログ・ディレクトリーを server\_root/account に変更すると、server\_root/account ディレクトリーの許可は、755 になり、jdoe によって所有されます。

## 形式

UserId {ID\_name | number}

## デフォルト

**AIX, Linux, Solaris:** UserId nobody

**HP-UX:** UserId www

## V2CipherSpecs - SSL バージョン 2 についてサポートされる暗号仕様をリストする

このディレクティブは、SSL バージョン 2 に使用可能な暗号仕様をリストします。

**注:** SSL ディレクティブは、SUSE Linux ではサポートされていません。

## 形式

V2CipherSpecs *specification*

以下を任意に組み合わせたものが許容値となります。いずれも 2 回使用することはできません。

- 1 — RC4 US
- 2 — RC4 Export
- 3 — RC2 US
- 4 — RC2 Export
- 6 — DES 56-bit
- 7 — Triple DES US

- NULL — デフォルトの暗号仕様が使用される

### 例

- 米国の場合: V2CipherSpecs '137624'
- 米国以外の場合: V2 Cipherspecs '246'

### デフォルト

なし (デフォルトでは SSL は使用不可)

## V3CipherSpecs - SSL バージョン 3 についてサポートされる暗号仕様をリストする

このディレクティブは、SSL バージョン 3 について使用可能な暗号仕様をリストします。

注: SSL ディレクティブは、SUSE Linux ではサポートされていません。

FIPSEnable ディレクティブが「on」に設定されている場合、V3CipherSpecs ディレクティブは無視されます。詳しくは、234 ページの『FIPSEnable - SSLV3 および TLS 用の Federal Information Processing Standard (FIPS) 承認済み暗号を使用可能にする』を参照してください。

### 形式

V3CipherSpecs *specification*

許容値として、以下の値があります。

- 00 — NULL NULL
- 01 — NULL MD5
- 02 — NULL SHA
- 03 — RC4 MD5 Export
- 04 — RC4 MD5 US
- 05 — RC4 SHA US
- 06 — RC2 MD5 Export
- 09 — DES SHA Export
- 0A — Triple DS SHA US
- 62 — 56-bit DES CBC SHA
- 64 — 56-bit RC4 SHA
- NULL — デフォルトの暗号仕様が使用される。

### 例

- 米国の場合: V3CipherSpecs '0A09060564620403020100'
- 米国以外の場合: V3Cipherspecs '0906646203020100'

### デフォルト

なし (デフォルトでは SSL は使用不可)

## WebMasterEMail - 選ばれたサーバー報告書を受け取るための電子メール・アドレスを設定する

このディレクティブは、SSL 証明書の有効期限切れ 30 日前の通知など、選ばれた Caching Proxy 報告書を受け取るための電子メール・アドレスを設定するために使用します。Linux および UNIX システムでは、sendmail プロセスが実行されていなければなりません。Windows システムの場合、sendmail プロセスは Caching Proxy 内に構築されるため、外部メール・サーバーは必要ありませんが、他に

『WebMasterSocksServer (Windows のみ) - sendmail ルーチン用に Socks サーバーを設定する』および 300 ページの『SMTPServer (Windows のみ) - sendmail ルーチン用に SMTP サーバーを設定する』で説明する 2 つのディレクティブを設定しなければなりません。

注: この電子メール・アドレスは、無名 FTP パスワードとしても使用することができます。

### 形式

WebMasterEMail *webmastermailaddress*

### 例

WebMasterEMail webmaster@computer.com

### デフォルト

WebMasterEMail webmaster

## WebMasterSocksServer (Windows のみ) - sendmail ルーチン用に Socks サーバーを設定する

このディレクティブは、Caching Proxy for Windows 内の内部 sendmail ルーチンによって使用される Socks サーバーを設定するために使用します。

『WebMasterEMail - 選ばれたサーバー報告書を受け取るための電子メール・アドレスを設定する』および 300 ページの『SMTPServer (Windows のみ) - sendmail ルーチン用に SMTP サーバーを設定する』で説明する 2 つのディレクティブも、このルーチンのために設定しなければなりません。

### 形式

WebMasterSocksServer *IP address or hostname of socks server*

### 例

WebMasterSocksServer socks.mybox.com

### デフォルト

なし

## Welcome - ウェルカム・ファイルの名前を指定する

このディレクティブは、特定のファイル名を含んでいない要求に応答するためにサーバーが探索するウェルカム・ファイルの名前を指定するときに使用します。このディレクティブを構成ファイル内に複数回指定して、ウェルカム・ファイルのリストを作成することができます。

ファイル名またはディレクトリー名を含まない要求の場合、サーバーは、常にファイル・ルート・ディレクトリーを見て、Welcome ディレクティブに指定された名前と一致するファイルを探します。一致するファイルが見つかったら、そのファイルが要求側に戻されます。

ディレクトリー名を含むが、ファイル名は含まない要求の場合、AlwaysWelcome ディレクティブは、サーバーがディレクトリーで戻すべきウェルカム・ファイルを探すかどうかを指定します。デフォルトにより、AlwaysWelcome の値は On に設定されます。つまり、サーバーは必ず要求されたディレクトリーを見て、Welcome ディレクティブに指定された名前に一致するファイルを探すということです。一致するファイルが見つかったら、そのファイルが要求側に戻されます。

サーバーが、ディレクトリー内のファイルと Welcome ディレクティブに指定されたファイル名との間で一致するものを複数見つけた場合は、Welcome ディレクティブの順序によって、どのファイルが戻されるかが決まります。サーバーは、構成ファイルの一番上に最も近い Welcome ディレクティブを使用します。

## 形式

Welcome *file\_name* [*server\_IP\_address* | *host\_name*]

*file\_name*

ウェルカム・ファイルとして定義したいファイルの名前を指定します。

[*server\_IP\_address* | *host\_name*]

複数の IP アドレスまたは仮想ホストを使用している場合は、このパラメーターを使用して、IP アドレスまたはホスト名を指定してください。サーバーは、この IP アドレスで受け取った要求、またはこのホストに対する要求にのみ、このディレクティブを使用します。IP アドレスの場合、これはサーバーのネットワーク接続のアドレスであり、要求を出しているクライアントのアドレスではありません。

IP アドレス (例えば、240.146.167.72) を指定するか、またはホスト名 (例えば、hostA.bcd.com) を指定することができます。

このパラメーターはオプションです。このパラメーターを使用しないと、サーバーは、要求が入ってきた IP アドレスや URL のホスト名とは無関係に、このディレクティブをすべての要求に使用します。

ワイルドカード文字をサーバーの IP アドレスとして指定することはできません。

## 例

- 以下の例では、2 つのウェルカム・ページを定義して、AlwaysWelcome ディレクティブがデフォルト値の On に設定されているものと想定しています。ファイル名を含まない要求の場合に、サーバーは、要求に指定されたディレクトリー (要求がファイル名またはディレクトリーを指定していない場合は、ファイル・ルート・ディレクトリー) から、ウェルカム・ファイルに戻そうとします。サーバーは、まず letsgo.html という名前のファイルを検索します。その名前のファイルがディレクトリーにない場合には、サーバーは Welcome.html という名前のファイルを探します。

```
Welcome letsgo.html
Welcome Welcome.html
```



- 以下の例において、サーバーは、要求が入ってきたネットワーク接続の IP アドレスに基づいて、別のウェルカム・ファイルを探します。0.67.106.79 に入ってくる要求の場合には、サーバーは CustomerA.html という名前のウェルカム・ファイルを探します。0.83.100.45 に入ってくる要求の場合には、サーバーは CustomerB.html という名前のウェルカム・ファイルを探します。別の IP アドレスに送信された要求に対しては、デフォルトのアドレスが参照されます。

```
Welcome CustomerA.html 0.67.106.79
Welcome CustomerB.html 0.83.100.45
```

- 以下の例において、サーバーは、URL のホスト名に基づいて別のウェルカム・ファイルを探します。hostA に送信された要求の場合、サーバーは CustomerA.html という名前のウェルカム・ファイルを探します。hostB に入ってくる要求の場合には、サーバーは CustomerB.html という名前のウェルカム・ファイルを探します。別のホストに対する要求が入ってくると、サーバーはデフォルトのホスト名を検索します。

```
Welcome CustomerA.html hostA.bcd.com
Welcome CustomerB.html hostB.bcd.com
```

## デフォルト

以下のデフォルトは、デフォルト構成で使用される順になっています。

```
Welcome Welcome.html
Welcome welcome.html
Welcome index.html
Welcome Frntpage.html
```



---

## 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032  
東京都港区六本木 3-2-31  
IBM World Trade Asia Corporation  
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation  
ATTN: Software Licensing  
11 Stanwix Street  
Pittsburgh, PA 15222-9183  
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用するのですが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者をお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

---

## 商標

以下は、IBM Corporation の商標です。

- AIX®
- IBM
- Netfinity®

- RS/6000<sup>®</sup>
- SecureWay<sup>®</sup>
- Tivoli
- ViaVoice<sup>®</sup>
- WebSphere

Java<sup>™</sup> およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft<sup>®</sup>、Windows、Windows NT<sup>®</sup>、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Intel<sup>™</sup>、Intel Inside (ロゴ)、MMX<sup>™</sup> および Pentium<sup>®</sup> は、Intel Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。









Printed in Japan

GD88-6861-00



日本アイ・ビー・エム株式会社  
〒106-8711 東京都港区六本木3-2-12

Spine information:



WebSphere Application  
Server

Caching Proxy 管理ガイド

バージョン 6.1

GD88-6861-00