Zen v15

Advanced Operations Guide

Procedures and References for Advanced Users

ACTIAN™

Advanced Operations Guide
2023 年 5 月

目次

このドキュメントについて

このドキュメントでは、上級ユーザーのための操作の説明と技術情報を提供します。

ここに提示される内容には、お使いのプラットフォームや製品には適用されないものも含まれています。たとえば、ゲートウェイ設定に関するトピックは Zen サーバー エンジンには適用されません。このような相違点は区別できるよう明記しています。

このドキュメントの読者

このドキュメントは上級ユーザー向けに提供されています。上級ユーザーは Zen ベース アプリケーションの基盤にあるオペレーティング システムについて深く理解していることが前提とされます。上級ユーザーは、オペレーティング システムの構成をスムーズに行うことができ、多くの場合データベース エンジンを構成する管理者権限を持っている必要があります。上級ユーザーとは、以下のような方が該当します。

- 1 つ以上の Zen ベース アプリケーションがインストールされているネットワークの管理者
- Zen ベース アプリケーションの付加価値再販業者
- Zen ベース アプリケーションの開発者

表記上の規則

特段の記述がない限り、コマンド構文、コード、およびコード例では、以下の表記が使用されます。

大文字と小文字の区別	通常、コマンドと予約語は、大文字で表記されます。本書で別途記述がない限り、これらの項目は大文字、小文字、あるいはその両方を使って入力できます。たとえば、`MYPROG`、`myprog`、または `Myprog` は同等です。
太字	太字で表示される単語には次のようなものがあります。メニュー名、ダイアログ ボックス名、コマンド、オプション、ボタン、ステートメントなど。
固定幅フォント	固定幅フォントは、コマンド構文など、ユーザーが入力するテキストに使われます。
[ ]	省略可能な情報には、［*log_name*］のように、角かっこが使用されます。角かっこで囲まれていない情報は必ず指定する必要があります。
|	縦棒は、［*file name* | *@file name*］のように、入力する情報の選択肢を表します。
< >	＜＞ は、`/D=<5|6|7>` のように、必須項目に対する選択肢を表します。
変数	*file name* のように斜体で表されている語は、適切な値に置き換える必要のある変数です。
…	［*parameter...*］のように、情報の後に省略記号が続く場合は、その情報を繰り返し使用できます。
::=	記号 ::= は、ある項目が別の項目用語で定義されていることを意味します。たとえば、a::=b は、項目 *a* が *b* で定義されていることを意味します。

Zen データベース

1

オブジェクト名、名前付きデータベースおよび DSN の詳細

このセクションは、以下のトピックに分かれています。

名前付きデータベース

名前付きデータベース（DBname とも呼ばれます）は論理名付きのデータベースです。ユーザーはデータベースが存在する場所を知らなくても、この名前でそのデータベースを識別することができます。Zen では、すべてのデータベースが名前付きデータベースである必要があります。データベースに名前を付ける際は、その名前を特定の辞書ディレクトリのパスおよび 1 つまたは複数のデータ ファイルのパスに関連付けるようにします。

名前付きデータベースへは、さまざまなアクセス方法で接続されます。たとえば、ODBC アクセスの場合は名前付きデータベースを指すデータ ソース名（DSN）を設定する必要があります。複数の DSN が同じ名前付きデータベースを指すことができます。『*ODBC Guide*』の「ODBC データベース アクセス」を参照してください。その他のアクセス方法の場合、アプリケーション開発者が、それぞれのアクセス方法の API を使用して名前付きデータベースに接続することができます。Zen ドキュメントの開発者向けガイドを参照してください。

メモ 名前付きデータベースを使用する場合、データベース エンジンが存在するコンピューターにログインする際は、管理者レベルまたは Zen_Admin セキュリティ グループのメンバーであるオペレーティング システム ユーザー名を使用する必要があります。

名前付きデータベースを作成する最も簡単な方法は、Zen Control Center を使用することです。『*Zen User's Guide*』の「新規データベースを作成するには」を参照してください。アプリケーション開発者は別のアクセス方法の API を用いて名前付きデータベースを作成することもできます。たとえば、SQL の場合は 「CREATE DATABASE」、DTI の場合は 「PvCreateDatabase()」、ADO.NET の場合は 「Data Access Application Block」を参照してください。

メタデータ

リレーショナル エンジンでは、メタデータでバージョン 1（V1）とバージョン 2（V2）という 2 つのバージョンをサポートします。メタデータ バージョン 2 では、多くの識別子に最大 128 バイトの名前を付けることができ、ビューおよびストアド プロシージャを許可し、メタデータ バージョン 2 固有の DDF（データ辞書ファイル）を持つことができます。

『*ODBC Guide*』の 「SQL 文法のサポート」を参照してください。

識別子とオブジェクト名

識別子は、データベースの名前、またはデータベース内の列、テーブル、プロシージャやその他名前付きオブジェクトの名前です。識別子は、通常の識別子またはデリミター（区切り記号）付きの識別子として指定されます。

通常の識別子

通常の識別子とは、二重引用符で囲まれていない識別子のことです。通常の識別子は大文字または小文字の文字で始まる必要があります。識別子の残りの部分は大文字または小文字の文字、数字、および有効な文字の任意の組み合わせで構成されます。

通常の識別子に予約語を使用することはできません。

通常の識別子は大文字小文字を区別しません。

デリミター付き識別子

デリミター付き識別子とは、二重引用符で囲まれた識別子のことです。デリミター付き識別子は、有効な文字から成る任意の文字列と、それを囲む二重引用符で構成されます。

推奨できる使用法ではありませんが、デリミター付き識別子には予約語も使用できます。たとえば、INSERT は通常の識別子としての使用は許可されませんが、"INSERT" はデリミター付き識別子としては許可されます。識別子がキーワードでもある場合は二重引用符で囲む必要があります。たとえば、SELECT "*password*" FROM my_pword_tbl となります。"*password*" は SET PASSWORD ステートメントのキーワードなので二重引用符で囲みます。

識別子の制限

上に挙げた全般的な制限以外に、各種の識別子に特有の制限を次の表に一覧表示します。

表 1　識別子の種類別の制限

識別子	長さ制限（バイト単位）		無効な文字 [1]	注記
	V1[2]	V2[3]		
列	20	128	¥ / : * ? " < > |	先頭は文字でなければなりません ヌルにすることはできません
データベース	20	20	` ~ ! @ # $ % ^ & * ( ) _ - + = }] { [| ¥ : ; " < , ' > .? /	先頭は文字でなければなりません
関数 (ユーザー定義)	30	128	通常の識別子の場合： ` ~ ! @ # $ % ^ & * ( ) - + = }] { [| ¥ : ; " < , ' > .? /	文字、数字、およびアンダースコア（"_"）が有効です 先頭は文字でなければなりません
			デリミター付き識別子の場合： なし	名前を二重引用符で囲む必要があります
グループ	30	128	¥ / : * ? " < > |（および空白文字）	MASTER にすることはできません
インデックス	20	128	¥ / : * ? " < > |（および空白文字）	Zen Control Center（ZenCC）でインデックスを作成する場合、先頭に UK_ を付けてはいけません ZenCC 外で UK_ で始まるインデックスを作成した場合、そのインデックスは ZenCC で編集できません
キー (外部または主)	20	128	¥ / : * ? " < > |（および空白文字）	先頭は文字でなければなりません 同一テーブル内で、外部キーとインデックスに同じ名前を付けることはできません
パスワード	8	128	; ? " '	先頭を空白 (空白として使用される文字) にすることはできません ヌルにすることはできません [無効な文字] 列に挙げられている文字以外の、あらゆる表示可能な文字が指定できます
プロシージャ (ストアド)	30	128	通常の識別子の場合： ` ~ ! @ # $ % ^ & * ( ) - + = }] { [| ¥ : ; " < , ' > .? /	文字、数字、およびアンダースコア（"_"）が有効です 先頭は文字でなければなりません
			デリミター付き識別子の場合： なし	名前を二重引用符で囲む必要があります
テーブル	20	128	¥ / : * ? " < > |（および空白文字） # ##[4]	無効な文字は、通常とデリミター付きの両方の識別子に適用されます
トリガー	30	128	通常の識別子の場合： ` ~ ! @ # $ % ^ & * ( ) - + = }] { [| ¥ : ; " < , ' > .? /	文字、数字、およびアンダースコア（"_"）が有効です 先頭は文字でなければなりません
			デリミター付き識別子の場合： なし	名前を二重引用符で囲む必要があります

表 1　識別子の種類別の制限

識別子	長さ制限（バイト単位）		無効な文字 [1]	注記
	V1[2]	V2[3]		
ユーザー	30	128	¥ / : * ? " < > | （および空白文字）	MASTER または PUBLIC にすることはできません
ビュー	20	128	通常の識別子の場合： ` ~ ! @ # $ % ^ & * ( ) - + = }] { [| ¥ : ; " < , ' > .? /	文字、数字、およびアンダースコア（"_"）が有効です 先頭は文字でなければなりません
			デリミター付き識別子の場合： なし	名前を二重引用符で囲む必要があります

[1] 特に示されていない限り、無効な文字は通常の識別子とデリミター付き識別子の両方に適用されます。

[2] バージョン 1（V1）メタデータに適用。『*ODBC Guide*』の「SQL 文法のサポート」を参照してください。

[3] バージョン 2（V2）メタデータに適用。『*ODBC Guide*』の「SQL 文法のサポート」を参照してください。

[4] テンポラリ テーブルの名前の先頭は # または ## で始まります。このため、# と ## は永続テーブルの名前の先頭文字としては無効です。『*SQL Engine Reference*』の「CREATE （テンポラリ）TABLE」を参照してください。

ユニークなスコープ

通常、識別子は一定の**スコープ**内でユニークである必要があります。つまり、同じ名前を使用する同一タイプのオブジェクトのインスタンスは同一領域内では使用できません。表 2 は、オブジェクト名がどのような領域または**スコープ**内でユニークである必要があるかを表します。

表 2　共通識別子のユニーク性の適用範囲

このタイプのオブジェクトの名前	この範囲内でユニーク			
	データベース	テーブル	ストアド プロシージャ	その他
データベース				一定のデータベース エンジンによってホストされるすべてのデータベース
テーブル	X			
トリガー、ストアド プロシージャ、ユーザー定義関数	X			
ユーザーまたはグループ	X			
ビュー	X			
制約	X			
列		X		
インデックス		X		外部キーと同じ名前を持つことはできません
キー（外部）		X		インデックスと同じ名前を持つことはできません
カーソル			X	

デフォルトのデータベースと現在のデータベース

既存のアプリケーションで、Btrieve ファイルを作成したり開いたりする際にデータベース名を指定していないアプリケーションをサポートするために、Zen では、トランザクショナル データベース エンジンごとのデフォルト データベースという概念が維持されています。**デフォルト データベース**は、"DefaultDB" という名前であらかじめ定義されているデータベースです。アプリケーション コードを変更しないで新しいセキュリティ モデルを使うようにするには、Btrieve データ ディレクトリをデフォルト データベースと関連付けてから、それらのディレクトリにあるデータ ファイルへのアクセスを制御するよう、デフォルト データベースでユーザーおよび権限を設定します。

また、データベース エンジンは、クライアント接続ごとの**現在のデータベース**という概念も理解しています。Btrieve の Login（78）、Create（14）、または Open（0）オペレーションでデータベース名が指定されていない場合、トランザクショナル エンジンは、そのオペレーションは現在のデータベースに関連するものであると見なします。現在のデータベースとは、各クライアントで、一番最近 Login（78）オペレーション（明示的ログイン）が発生したデータベースを指します。クライアント コンピューターが明示的なログイン操作を要求していない場合は、一番最近 Create（14）または Open（0）オペレーション（暗黙ログイン）が発生したデータベースが現在のデータベースとなります。明示的にも暗黙的にもログインが発生していない場合は、前の段落で説明したデフォルト データベースが現在のデータベースとなります。クライアントが明示または暗黙のログインを実行した場合、あるいは最後のファイル ハンドルを閉じることによって DefaultDB が現在のデータベースとなった場合には常に、現在のデータベースが変わる可能性があることに注意してください。各クライアントの現在のデータベースは、ほかのクライアントの動作とは関係ありません。

既存のアプリケーションに新しいセキュリティ モデルを構成する最も簡単な方法は、すべての Btrieve データ ディレクトリをデフォルト データベースと関連付け、このデータベース内で PUBLIC グループの権限を設定することです。PUBLIC グループは、データベースのセキュリティを有効にしたとき、Master ユーザーと共に自動的に作成されます。「MicroKernel エンジン セキュリティのクイック スタート」を参照してください。

ファイル構造

すべての Zen データベースは共通のデータ形式を使用します。このファイル形式の共有により、同一データのアクセスにトランザクショナル、リレーショナルなどの異なるアクセス方法を使用することができます。すべてのアクセス方法が機能するために使用するシステムは MicroKernel エンジンと呼ばれます。

各 Zen データベース テーブルは個別のファイルで、デフォルトでは .mkd という拡張子が付きます。ただし、開発者は独自のファイル名拡張子を指定することができます。MicroKernel ファイルはデータとインデックスの両方を持ち、さまざまなタイプのページで構成されます。MicroKernel ファイルは共通のデータ形式でデータを格納します。

各 Zen データベースには、拡張子 .ddf のデータ辞書ファイル一式も含まれます。DDF ファイルにはデータベース スキーマが含まれます。メタデータ バージョン 1 とメタデータ バージョン 2 の DDF は異なるファイル名を使用します。『*SQL Engine Reference*』の「システム テーブル」を参照してください。

MicroKernel エンジンは、キー フィールドは別として、データのスキーマには関わりません。ただし、参照整合性の規定または SQL 経由のアクセスではスキーマの知識が必要です。

Zen データベースの名前とロケーションは dbnames.cfg という名前のバイナリ ファイルに格納されます。Zen ファイルのデフォルトの保存場所については、『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

Zen データベースに関連するすべてのファイルはオペレーティング システムから表示させることができます。表 3 は関連するファイルを要約したものです。

表 3　Zen データベースに関連するファイル

種類	説明
データベース名の構成	dbnames.cfg ファイル。Zen データベースの名前とロケーションを含むバイナリ ファイルです。
データ（共通データ形式）	ファイル名は、リレーショナル データベースの場合デフォルトで、**テーブル名** .mkd です。データベース テーブルごとに対応する MicroKernel ファイルがあります。トランザクショナル データ ファイルでは、各ファイル名はアプリケーションが指定します。
データ辞書	拡張子が DDF のファイル。『*SQL Engine Reference*』の「システム テーブル」を参照してください。

ファイル サイズ

サイズの制限はファイル バージョンやページ サイズ、および 1 ページあたりのレコード数によって異なるため、次の表にまとめました。

ファイル バージョン 13.0

データ ファイルの最大サイズは 64 TB です。単一ファイルのサイズが 256 GB を超える場合は、13.0 以上のファイル形式を使用する必要があります。

次の表は、ファイルのレコードに圧縮が設定されていないことを前提にしています。レコード圧縮を使用する場合は、1 ページあたりに格納されるレコードがさらに増えることを考慮してください。『*Zen Programmer's Guide*』の「ページ サイズの選択」および「ファイル サイズの予測」を参照してください。

表 4　ファイル バージョン 13.0 のファイル サイズおよびページ サイズの比較

最大ページ数（100 万単位）	各ページ サイズ（バイト）に対するファイル サイズ（TB）		
	4096	8192	16384
4096	16 TB	32 TB	64 TB

ファイル バージョン 9.5

データ ファイルの最大サイズは 256 GB です。単一ファイルのサイズが 128 GB を超える場合は、9.5 以上のファイル形式を使用する必要があります。

次の表は、ファイルのレコードに圧縮が設定されていないことを前提にしています。レコード圧縮を使用する場合は、1 ページあたりに格納されるレコードがさらに増えることを考慮してください。『*Zen Programmer's Guide*』の「ページ サイズの選択」および「ファイル サイズの予測」を参照してください。

表 5　ファイル バージョン 9.5 のファイル サイズおよびページ サイズの比較

1 ページあたりのレコード数	最大ページ数（100 万単位）	各ページ サイズ（バイト）に対するファイル サイズ（GB）				
		1024	2048	4096	8192	16384
1 - 15	256	256 GB	256 GB	256 GB	256 GB	256 GB
16 - 31	128	128 GB	256 GB	256 GB	256 GB	256 GB
32 - 63	64	64 GB	128 GB	256 GB	256 GB	256 GB

表 5　ファイル バージョン 9.5 のファイル サイズおよびページ サイズの比較

1 ページあたりのレコード数	最大ページ数（100 万単位）	各ページ サイズ（バイト）に対するファイル サイズ（GB）				
		1024	2048	4096	8192	16384
64 - 127	32	32 GB	64 GB	128 GB	256 GB	256 GB
128 - 255	16	16 GB	32 GB	64 GB	128 GB	256 GB
256 - 511	8	N/A[1]	16 GB	32 GB	64 GB	128 GB
512 - 1023	4	N/A[1]	N/A[1]	16 GB	32 GB	64 GB
1024 - 2047	2	N/A[1]	N/A[1]	N/A[1]	16 GB	32 GB
2048 - 4095	1	N/A[1]	N/A[1]	N/A[1]	N/A[1]	16 GB

[1] N/A は「適用外」を意味します。

ファイル バージョン 9.0 以下

データ ファイルの最大サイズは 128 GB です。単一ファイルのサイズが 64 GB を超える場合は、9.0 以上のファイル形式を使用する必要があります。

次の表は、ファイルのレコードに圧縮が設定されていないことを前提にしています。レコード圧縮を使用する場合は、1 ページあたりに格納されるレコードがさらに増えることを考慮してください。『*Zen Programmer's Guide*』の「ページ サイズの選択」および「ファイル サイズの予測」を参照してください。

表 6　ファイル バージョン 9.0 以下のファイル サイズおよびページ サイズの比較

ファイル バージョン	1 ページあたりのレコード数	最大ページ数（100 万単位）	各ページ サイズ（バイト）に対するファイル サイズ（GB）								
			512	1024	1536	2048	2560	3072	3584	4096	8192
9.0	1 - 15	256	128	128	128	128	128	128	128	128	128
9.0	16 - 31	128	64	128	128	128	128	128	128	128	128
9.0	32 - 63	64	32	64	96	128	128	128	128	128	128
9.0	64 - 127	32	16	32	48	64	80	96	112	128	128
9.0	128 - 255	16	N/A[1]	16	24	32	40	48	56	64	128
8	任意	16	8	16	24	32	40	48	56	64	N/A[1]
7	任意	16	8	16	24	32	40	48	56	64	N/A[1]
6	任意	16	4	4	4	4	4	4	4	4	N/A[1]
5	任意	16	4	4	4	4	4	4	4	4	N/A[1]

[1] N/A は「適用外」を意味します。

ファイルのセグメント化

デフォルトでは、データ ファイルは、オペレーティング システムのファイル セグメントである 2 GB の限界を超えるごとに自動的に分割されます。[セグメント サイズを 2 GB に制限] 設定プロパティを使用すると、ファイルを 2 GB のセグメントに分割するか、セグメント化しない 1 つのファイルとするかを指定することができます。よ

り大きいサイズの非セグメント化ファイルを使用する利点は、より効率的なディスク I/O です。つまり、パフォーマンスの向上が期待できます。13.0 形式のファイルはセグメント化をサポートしないので注意してください。

この設定オプションはデータベース エンジンのパフォーマンス チューニング用プロパティの 1 つです。「ZenCC でエンジンのプロパティを設定するには」、および「セグメント サイズを 2 GB に制限」を参照してください。

このプロパティはデフォルトでオンに設定されており、以前のリリースと同様 2 GB の限界でファイルはセグメント化されます。このプロパティをオフに設定すると、ファイルは 2 GB の限界を超えて増大します。設定プロパティの追加情報については、「ファイル バージョンの自動アップグレード」も参照してください。

セグメントされていないファイルは、お使いのオペレーティング システムによって指定されているファイル サイズの制限を受けます。たとえば、FAT32 ファイル システムで［「セグメント サイズを 2 GB に制限」］の設定をオフにしてサイズの大きなファイルを作成すると、倍の 4 GB ファイルに拡張されます。以前作成したファイルが既にセグメント化されている場合、そのセグメントはファイル上でそのまま残ります。

ファイル バージョンの自動アップグレード

設定プロパティの［作成ファイルのバージョン］に 9.0 以上が設定されている場合、バージョン 8.x ファイルはそのファイル サイズの限界（64 GB）に達すると自動的にバージョン 9.0 ファイルに変換されます。次の表に、この動作をまとめて示します。

［作成ファイルのバージョン］設定プロパティの設定	［セグメント サイズを 2 GB に制限］設定プロパティの設定	ファイル バージョンの自動アップグレードが起こるファイル サイズ
9（デフォルト）	オン（デフォルト、オプションがチェックされている）	64 GB（バージョン 8.x ファイルの最大サイズ）
9（デフォルト）	オフ（オプションがチェックされていない）	2 GB（バージョン 8.x ファイルのセグメントのサイズ）

たとえば、バージョン 8.x ファイルでサイズが 5 GB の場合は、既に 2 GB 単位のセグメント化が行われています。このファイルは既にセグメント化されているので、そのセグメントはファイルに存在し続けます。このようなファイルはその後もセグメント化が続行され、自動アップグレードが起こるサイズ 64 GB までそのサイズを増大させることができます。この動作は、ファイルが既にセグメント化されているため、設定プロパティがオンまたはオフのどちらであっても同じです。ファイルのサイズが 64 GB を超えると、バージョン 9.0 ファイルの最大許容サイズ 128 GB に達するまでセグメント化が続けられます。

バージョン 8.x ファイルでサイズが 1.5 GB の場合は、そのサイズを 2 GB まで増大させることができます。設定プロパティがオフの場合、このファイル サイズが 2 GB になった時点で自動アップグレードが起こります。このファイルは非セグメント化ファイルとして、そのサイズをバージョン 9.0 ファイルの最大許容サイズ 128 GB まで増大させることができます。設定プロパティをオンに設定すると、2 GB 単位のファイルのセグメント化が続行され、そのサイズを 64 GB のサイズまで増大させることができます。バージョン 8.x ファイル用の最大許容サイズ 64 GB になった時点で、バージョン 9.0 への自動アップグレードが起こります。ファイルのサイズが 64 GB を超えると、バージョン 9.0 ファイルの最大許容サイズ 128 GB に達するまでセグメント化が続けられます。

［作成ファイルのバージョン］オプションは、データベース エンジンのファイル互換性用プロパティの 1 つです。「ZenCC でエンジンのプロパティを設定するには」を参照してください。

メモ ファイル バージョンの自動アップグレードが動作するのは、8.x ファイル形式から 9.0 ファイル形式というパターンのみです。これ以外のファイル バージョンの組み合わせの場合、この自動アップグレードは動作しません。たとえば、8.x ファイル形式から 9.5 ファイル形式、あるいは 7.x ファイル形式から 9.0 ファイル形式という組み合わせではアップグレードは起こりません。

アクセス方法

Zen データベースのデータにアクセスする 2 つの主な方法は、トランザクショナルおよびリレーショナル アクセスです。

トランザクショナルでは、アプリケーション プログラムはデータ レコード内を物理的と論理的のどちらの順序に従ってでも自由に移動することができます。トランザクショナル API を使用することで、アプリケーション プログラムは直接制御を備え、開発者はデータの基本構造の知識に基づいてデータ アクセスを最適化することができます。Btrieve は、トランザクショナル データベース エンジンの 1 つです。

リレーショナルとは、データがテーブル、行、列の集まりとして表されるアクセス方法です。リレーショナル モデルは、開発者を基となるデータ構造から切り離し、データを単純な表形式で表します。ODBC はリレーショナル アクセス方法の 1 例です。

単一のアプリケーションが両方のタイプのアクセスを含むこともあります。たとえば、データの追加と変更にはトランザクショナル アクセスを使用し、データの照会およびレポート作成にはリレーショナル アクセスを使用することができます。

アプリケーション プログラムが使用するアクセス方法を知っておく必要があります。これはインストールされる Zen によって異なります。アクセス方法によって設定が異なります。特定のアクセス方法を最適化するために設定をカスタマイズする必要があります。

利用するアプリケーションが使用するアクセス方法を知っていれば、トラブルシューティングも容易になります。たとえば、アプリケーション プログラムが ODBC 経由でリレーショナル アクセスを使用している場合、データベース管理システムではなく ODBC レベルの問題を解決する必要がある可能性があります。

設定のカスタマイズに関する作業とリファレンスについては、「設定リファレンス」を参照してください。

クライアント / サーバー通信

MicroKernel エンジンは、ローカルおよびクライアント / サーバーの 2 種類の処理モードをサポートします。ローカル モードでデータベースにアクセスするアプリケーションは、ローカルにあるエンジンにアクセスします。ローカルのエンジンは、ローカルまたはネットワークのハード ディスクの I/O を実行するワークステーションのオペレーティング システムに要求を出します。

クライアント / サーバー モードでは、共有ファイル サーバー上で実行されるサーバー MicroKernel エンジンを使用します。アプリケーション プログラムがクライアント / サーバー モードでデータベース エンジンにアクセスしているときは、リクエスターがリモート エンジンに接続します。このリクエスターは、オペレーティング システムがサポートするネットワーク プロトコルを使用して、トランザクション レベルのリクエストおよびデータ レコードを、アプリケーション プログラムとサーバー エンジン間で受け渡しします。ファイル I/O 機能は、クライアント / サーバー モードのサーバー エンジンによって完全に処理され、ワークステーションには共有データ ファイルのためのオペレーティング システムのハンドルは割り当てられません。データベース操作は、各ワークステーションに代わってサーバー ベースのエンジンによって実行されます。

処理モードは、アプリケーション プログラムそのものではなく、ワークステーションの設定によって決定されることに注意してください。つまり、アプリケーションはローカルとクライアント / サーバー、どちらのデータベース エンジンにもアクセスできます。アプリケーション プログラムは、ローカル モードからクライアント / サーバー モードに切り替える際に再コンパイルする必要はありません。

ワークグループ エンジンおよびサーバー エンジンはどちらのモードでも動作します。データベース エンジンと同一マシン上のアプリケーションがエンジンにアクセスする場合、ローカル モードで動作します。データベース エンジンと異なるマシン上のアプリケーションがエンジンにアクセスする場合、クライアント / サーバー モードで動作します。

クライアント / サーバーの設定は、Zen のワークグループおよびサーバー バージョン用にカスタマイズできます。スタンドアロンでもクライアント / サーバーでも、その設定を容易にするための構成の設定プロパティが Zen Control Center（ZenCC）に含まれています。

クライアント / サーバー通信およびデータベース エンジンの設定に関する作業とリファレンスについては、「設定リファレンス」を参照してください。

データベース コード ページ

エンコードは文字セットを表す標準規格です。文字データは、コンピューターがデジタル処理できる標準形式に変換する、つまりエンコードする必要があります。エンコードは、Zen データベース サーバーと Zen クライアント アプリケーションとの間で規定する必要があります。互換性のあるエンコードを使用すれば、サーバーとクライアントでデータが正しく変換されます。

エンコードのサポートは、データベース コード ページとクライアント エンコードに分割されています。この 2 種類のエンコードは、別個のものですが相互に関係しています。説明を簡単にするために、データベース コード ページとクライアント エンコードを一緒に説明します。『*Getting Started with Zen*』の「クライアント用のネットワーク通信の設定」を参照してください。

ODBC DSN 作成オプション

『*ODBC Guide*』の 「DSN のセットアップおよび接続文字列」を参照してください。このトピックでは ODBC 接続文字列についても説明しています。

idshosts ファイルの使用

一般的には、アプリケーションは独自のファイルの場所情報を指定します。別の方法として、テキスト ファイル idshosts の情報に基づいてファイル場所のマッピングを指定することができます。

idshosts ファイルは Zen（IDS）の一部でした。IDS はコア製品から取り除かれましたが、idshosts ファイルの設定は依然として可能です。

作成したアプリケーションでは idshosts によるマッピング機能を使用しないという場合は、[**IDS の使用**] 設定をオフにします。アプリケーションが既に idshosts を使用している場合、あるいはこの代替方法を使用してファイルの場所をマップしたいと考えている場合は、[**IDS の使用**] をオンにします。「IDS の使用」を参照してください。

idshosts ファイルを使用する場合は、ファイルにアクセスして内容を読み取る時間が必要となるため、パフォーマンスが低下することに注意してください。

idshosts ファイルは、Windows、Linux、または macOS クライアント リクエスターでのみ使用できます。クライアントは Windows、Linux、または macOS 上の Zen サーバーと通信できます。

メモ [**IDS の使用**] をオンに設定するには、Zen 8.5 以降が必要です。リクエスターはデータベース URI を使用して IDS 情報を表します。データベース URI は PSQL 8.5 で追加されました。『*Zen Programmer's Guide*』の「データベース URI」を参照してください。

[**IDS の使用**] をオンに設定した場合、[**リモート MicroKernel エンジンの使用**] もオンに設定する必要があります。[**リモート MicroKernel エンジンの使用**] はデフォルトでオンです。

「IDS の使用」および「リモート MicroKernel エンジンの使用」を参照してください。

idshosts エントリの形式

idshosts ファイル内のエントリの形式については、ファイル自体のコメントを参照してください。コメントにはマッピングの例も提供されています。デフォルトで、Windows プラットフォームの場合には、idshosts ファイルはデータベース クライアントのインストール ディレクトリ下の ¥bin ディレクトリにインストールされます。Linux、macOS、および Raspbian の場合には、idshosts ファイルはデータベース クライアントのインストール ディレクトリ下の /etc ディレクトリにインストールされます（例：/user/local/actianzen/etc）。

データベース管理の概念

2

データベース管理について

Zen は、MicroKernel Database エンジン（MKDE）を中心に構成された総合的なデータベース管理システムです。Zen により、インストールおよび管理が容易になり、高度な信頼性とパフォーマンスが実現します。Zen は事実上メンテナンスなしで何か月も何年も実行できますが、そのユニークな機能を理解し、役に立つタスクの実行方法を学ぶことにより、そのすべてを自分のものにすることができます。このマニュアルでは、Zen エンジンと関連するデータベースのチューニング、構成および管理の方法を説明します。

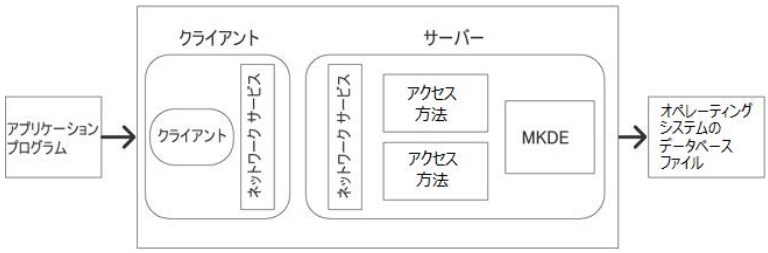
- 「設定」
- 「データベース セキュリティ」
- 「データの保存と復元」
- 「トラブルシューティング」
- 「有用なユーティリティ」

設定

データベース エンジンのサーバーとクライアントそれぞれについて別個に構成することができます。この設定により、ビジネスの要件に基づいて、エンジンのパフォーマンスを最適化することができます。

下図は、オペレーティング システムでのアプリケーション プログラムからデータベース ファイルへのフローを示しています。データベース エンジンはアプリケーション プログラムとデータ ファイルの間に存在します。

図 1 データベース エンジン設定の概念図



サーバーに設定可能な項目は以下のとおりです。

- アクセス
- 通信プロトコル
- 以前のバージョンの MicroKernel データベース エンジン（MKDE）との互換性
- データの整合性
- デバッグ
- ディレクトリ
- メモリの使用
- パフォーマンス

クライアントに設定可能な項目は以下のとおりです。

- アクセス
- 通信プロトコル
- パフォーマンス
- セキュリティ
- アプリケーションの特性

これらの構成は ZenCC を使用して行います。設定に関する作業とリファレンスについては、「設定リファレンス」を参照してください。

データベース セキュリティ

Zen データベースへのアクセスはいくつかの方法で保護されています。管理レベルのセキュリティはオペレーティング システムを介して設定されます。オペレーティング システムのネイティブなセキュリティ メカニズムを使用して、Zen データベースを誰が管理できるかを制御することができます。

Zen はユーザーおよびグループ レベルの関連するセキュリティも提供します。誰がどのような資格でデータにアクセスできるかを制御することができます。たとえば、Zen データベース内の各テーブルで、ユーザーまたはグループがテーブルに対し、SELECT、UPDATE、INSERT、DELETE、ALTER TABLE を実行できるかどうかを指定することができます。

データベース全体に対しパスワードを設定してセキュリティを確立することができます。この時点で、データベースにアクセスする権限を与えられたユーザーのみが Master という名前のデフォルト ユーザーとなります。その後、ユーザーやグループを追加することができます。

セキュリティは ZenCC 内で設定することができます。また、GRANT および REVOKE の 2 つの SQL ステートメントもサポートされています。これら 2 つの SQL ステートメントを使用すれば、テーブル レベルと列レベルの両方でセキュリティを設定することもできます。

GRANT 構文はトランザクショナル オーナー ネームと統合され、リレーショナル アクセスの場合にもオーナーネームを使用することができます。

セキュリティ、オーナー ネーム、ユーザーおよびグループに関する詳細は、「Zen セキュリティ」を参照してください。

データの保存と復元

データのバックアップはデータベースの保護とトラブル復旧を確実にするための重要な手順です。Zen データベースをバックアップおよび復元するにはいくつかの方法があります。

Zen データベースにアクセスするアプリケーションを停止することができる仕事の場合、オペレーティング システムやサードパーティ製ソフトウェアを使ってデータベース ファイルをバックアップおよび復元することができます。

アーカイブ ログはオペレーティング システムのユーティリティを補完するのに使用できるもう 1 つのバックアップ方法です。アーカイブ ログを使用すると、最後のバックアップ以降のすべてのデータベース操作のログを保存することができます。システム障害が発生した場合、バックアップからデータ ファイルを復元し、ログ ファイルから変更をロールフォワードして、データベースをシステム障害発生前の状態に戻すことができます。

Continuous **オペレーション**を使用すると、データベース エンジンが実行中でユーザーが接続中でも、データベース ファイルのバックアップを行うことができます。Continuous オペレーションの開始後、データベース エンジンはアクティブなデータ ファイルを閉じ、すべての変更をテンポラリ データ ファイル（**デルタ** ファイルと呼びます）に格納します。バックアップが完了したら、Continuous オペレーションを解除します。データベース エンジンはデルタ ファイルを読み取り、元のデータ ファイルにすべての変更を適用します。

データベースのバックアップおよび復元に関する追加情報については、「ログ、バックアップおよび復元」を参照してください。

トラブルシューティング

『*Zen User's Guide*』および『*Getting Started with Zen*』には、トラブルシューティングに関する情報が含まれています。『*Getting Started with Zen*』には、Zen 製品のインストールに関するトラブルシューティングの情報が記載されています。『User's Guide』には一般的なトラブルシューティングの情報に加えて、よく寄せられる質問のリストが記載されています。

有用なユーティリティ

Zen には、データベースの制御および管理に役立つようデザインされたさまざまなユーティリティが付属しています。主なユーティリティの一覧については、『*Zen User's Guide*』を参照してください。一部のユーティリティは、カスタム インストールではインストールされないことに注意してください。

Zen コンポーネントのアーキテクチャ

3

アーキテクチャ機能の説明

以下のトピックでは、インストールや重要なアプリケーションの実行で問題が起こらない環境を提供するために設計された機能について説明します。

- 「Zen データベース管理システム」
- 「リレーショナル アーキテクチャの概要」
- 「エラー コード」
- 「Auto Reconnect」

Zen データベース管理システム

Zen データベース管理システムは、以下の 2 つの方法でデータ処理アプリケーションをサポートしています。

- MicroKernel エンジン。Btrieve API を介して、データベース トランザクションへのアクセスを提供します。
- リレーショナル エンジン。ODBC を介して、SQL リレーショナル インターフェイスを提供します。

リレーショナル エンジンの SQL インターフェイスは、Btrieve API を使用してデータ ファイルにアクセスします。

このトピックでは、これら 2 つのエンジンの共有機能に加え、通常の運用環境に基づいて、これらの設定および動作における相違点について説明します。

共通アドレス空間

Zen は最適化されたメモリ アーキテクチャを使用し、トランザクショナル アクセスとリレーショナル アクセスの両方に高いパフォーマンスを提供します。MicroKernel エンジンおよびリレーショナル エンジンは、共に同一プロセス アドレス空間でロードかつ動作し、それらの間の通信にかかる CPU 時間を最小限にします。

行レベルのロック

同時に多数の更新や書き込みが行われたり、長時間にわたってトランザクションが開かれたままになったりする複数ユーザー環境では、行レベルのロック（Btrieve v6 データ ファイル形式で導入）によってデータベース エンジンのパフォーマンスが向上します。

トランザクションはページ全体ではなく直接影響する行のみをロックします。クライアントはあるページのレコードの変更を、他のクライアントが同一ページの別のレコードを変更するのと同時に行うことができます。2 番目のアプリケーションは、最初のアプリケーションが現在ロックしているのとまったく同じレコードを変更しようとしたときにのみ、待つことが必要になります。行レベルのロックは複数ユーザー環境で総体的な待ち時間を減少させ、パフォーマンスを向上させます。

行レベルのロックはデータ ページに実装され、部分的にはキー ページにも実装されます。可変ページには適用されません。キー ページの一部分の変更によって、キー エントリが別のページに移動することがあります。これはたとえば、キー ページが分割または結合されるときに起こります。これらの変更では、トランザクションが完了するまで完全なページ ロックが継続します。

MicroKernel エンジン

すべてのインストールにおいて、MicroKernel エンジンはエンジンと同じシステム上で実行されるローカル アプリケーションおよびデータ ファイルに対し、Btrieve API サポートを提供します。Zen クライアント / サーバー環境では、MicroKernel エンジンはローカル アプリケーションとリモート アプリケーションの両方をサポートします。ワークグループ環境では、MicroKernel エンジンはゲートウェイ構成を使用してリモート アプリケーションに接続し、リモート ワークグループ エンジンから送信された要求を処理できます。

Windows のクライアント / サーバー環境では、MicroKernel エンジンはデフォルトで、Windows サービスとして実行するようにインストールされます。ワークグループ環境では、エンジンをアプリケーションまたはサービスのどちらで実行するようにインストールするかを設定できます。アプリケーションとしてインストールされた場合は、エンジンが実行されていることを表すトレイ アイコンが表示されます。サーバー エンジンの場合、またはワークグループ エンジンがサービスとしてインストールされた場合には、トレイ アイコンは表示されません。「サーバーとワークグループの技術的な相違」も参照してください。

Zen の Btrieve API および ODBC API は、分散データベース アプリケーションをサポートすると同時に、それらのアプリケーションからローカルまたはリモート データベース エンジンへの接続の詳細を隠します。このアーキテクチャを使用すると、アプリケーションは、そのローカルにあるデータにアクセスできると同時に、リモート コンピューターのデータにもアクセスできます。さらに、SQL データベースは、ローカルの MicroKernel エンジンによって処理されるデータ辞書ファイル（DDF）、およびリモートの MicroKernel エンジンによって処理されるデータ ファイル（テーブル）を作成することにより、分散化が可能です。ローカルの MicroKernel エンジン以外のエンジンにも処理されるこのような SQL データベースは、「**混合アクセス データベース**」と呼ばれます。

混合アクセス データベースには、以下の制約があります。

- 参照整合性（RI）、バウンド データベース、トリガ、分散されたトランザクション アトミシティ（2 段階のコミットが必要）の機能は使用できません。
- DDF へのアクセスには、リレーショナル エンジンおよび MicroKernel エンジンが、同一コンピュータ上で動作している必要があります。
- 参照整合性の関係にかかわるテーブル、定義されているトリガーがあるテーブル、またはバインドされた名前付きデータベースにあるテーブルのデータ ファイルは、リモートの MicroKernel エンジンで開くことができません。
- ファイルを開く際、リレーショナル エンジンは、リクエストを処理する MicroKernel エンジンのバージョンを確認しません。バージョン 6.30 以降の MicroKernel エンジンの API サポート（たとえば、共有ロックなど）を必要とするオペレーションが、バージョン 6.30 より前のエンジンに対して発行された場合は、エラー コードが返されます。DDF を開く際、または DDF ファイルやデータ ファイルをバインドしようとする際に、リレーショナル エンジンでは、ローカルの MicroKernel エンジンがリクエストを処理していることを確認します。

非同期 I/O

Windows サーバーでは、MicroKernel エンジンはパフォーマンスを向上させるために、非同期 I/O を使用してページをディスクに書き込んでいます。エンジンは、Windows システム キャッシュまたは独自のキャッシュにページを書き込みます。次に、Windows はページがディスク上にあることを通知し、MicroKernel が効率的に書き込み操作を実行できるようにします。

MicroKernel エンジンでたくさんの並行オペレーションが同時に行われる場合、特にデータ セットがストライプ ディスク ドライバーのセット上にある場合には、読み取りパフォーマンスも向上しています。各読み取りは、ページが利用可能になるまでワーカ スレッドを待機させます。非同期 I/O では、オペレーティング システムは複数の読み取り要求の作業をプールして、読み取り操作をより効率的に行います。

リレーショナル エンジン

Zen のリレーショナル エンジンは、Zen アプリケーションに対し ODBC サポートを提供します。ODBC クライアントのプラットフォームには、Windows プラットフォームがあります。リレーショナル エンジンへのリモート ODBC アプリケーション アクセスには、Zen の ODBC クライアントがインストールされている必要があります。これは、ネットワーク上でクライアント側の ODBC 呼び出しを ODBC 通信サーバーに転送する専用 ODBC ドライバーです。

リレーショナル エンジンには以下のような機能が含まれています。

- アトミック ステートメント
- 双方向カーソル（ODBC カーソル ライブラリを使用）
- 外部結合への対応
- 更新可能なビュー
- ODBC のデータ型への対応
- テーブル内の複数の可変長列

ODBC 通信サーバーは、以下の機能を実行します。

- ODBC クライアント用ネットワーク通信のサポート
- サーバー側 ODBC ドライバー マネージャーへの ODBC 呼び出しの転送（呼び出しをエンジンに転送）

SQL および ODBC の詳細については、『*SQL Engine Reference*』の「SQL の概要」および『*ODBC Guide*』の「DSN のセットアップおよび接続文字列」を参照してください。

リレーショナル アーキテクチャの概要

下図は、サーバー版 Zen のリレーショナル エンジンのアーキテクチャ コンポーネントを示したものです。SQL 接続マネージャーは、MicroKernel エンジンおよびリレーショナル エンジンと同じプロセス アドレス空間で起動し実行します。

サーバーおよびワークグループの Zen リレーショナル アーキテクチャ

SQL 接続マネージャーは、最大 2000 までの同時接続をサポートしており、ODBC ドライバー マネージャーを使用してリレーショナル エンジン（SRDE : SQL Relational Database Engine）を呼び出します。エンジンは MicroKernel の上に置かれます。

図 2 は、Zen のクライアント / サーバー リレーショナル アーキテクチャを示します。クライアントは、TCP/IP を介してサーバーの SQL 接続マネージャーに接続します。このアーキテクチャは、サーバー エンジンおよびワーク グループ エンジン（ローカル ワークグループ エンジンからリモート ワーク グループ エンジンへの接続にクライアント DSN を使用する場合）に適用されます。

図 2 クライアント / サーバー リレーショナル アーキテクチャ

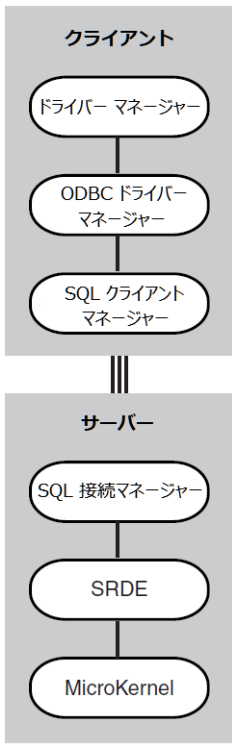
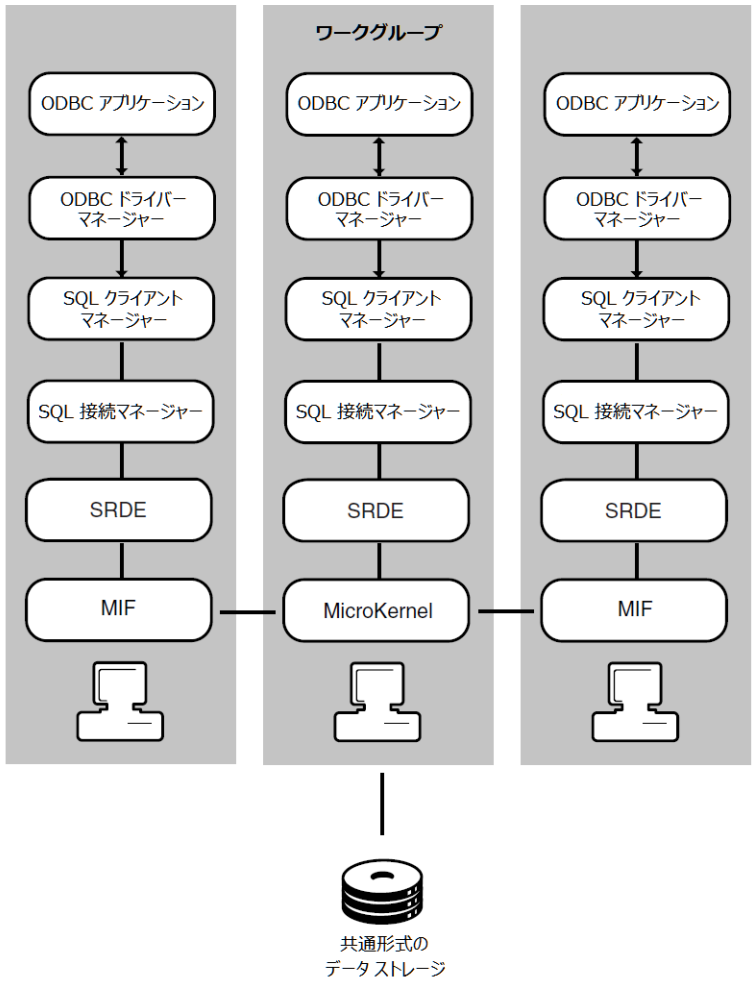


図 3 は、ローカル ワークグループ エンジンからリモート データベースへの接続に DSN を使用する場合のワークグループ リレーショナル アーキテクチャを示します。リモート ワークグループ エンジンはリモート データへのゲートウェイとなっていることを想定しています。

図 3 ワークグループ リレーショナル アーキテクチャ



エラー コード

Zen の多くの最上位コンポーネントは、その下の階層のコンポーネントからのエラー コードがそのまま渡されるため、呼び出し元アプリケーションまたはログ ファイルでエラーの実際の原因を明確に特定できるようになりました。エラー コードがさまざまな状況に該当する可能性がある場合は、Zen イベント ログ内の具体的な情報により、そのエラーの根本的な原因が特定されます。「メッセージ ログの見直し」を参照してください。

Auto Reconnect

Zen 自動再接続（Auto Reconnect）機能を使用すると、クライアント / サーバーまたはワークグループ アプリケーションは、一時的なネットワークの中断時にも現在のデータベース オペレーションを中止しません。Zen がネットワークの中断を検出すると、設定可能な指定時間ごとに自動的に再接続を試行します。この機能はクライアントのコンテキストも維持するので、通信が再確立されたとき、ネットワークの中断が起こったときとまったく同じようにデータベース アクセスが継続されます。

この機能は、アプリケーションのコンテキストを維持し、ネットワーク通信が中断されたときにクライアントまたはサーバーがその時点でデータを送信しようとしていたかどうかにかかわらず、再接続を試行します。

ネットワークが中断したとき、一定の待ち時間の経過ごとに再接続が試行されます。すべての接続で、0.5、1、2、4、8 秒ごとに継続的に再接続が試行され、その後、自動再接続タイムアウトの値に達するまで 8 秒ごとに試行されます。最大待ち時間に達しても接続に成功しない場合、現在のオペレーションは失敗し、クライアントの接続はリセットされます。最大の待ち時間は 45 秒から 65,635 秒の間に設定できます。

この機能はデフォルトで使用不可になっています。この機能を動作させるには、クライアントとサーバーの両方の設定で、「自動再接続の有効化（Windows のみ）」を選択する必要があります。待ち時間の値は、サーバー設定の「自動再接続タイムアウト」を使って行うことができます。

備考

この機能は、Btrieve、ODBC、および DTI 接続でサポートされています。

Btrieve 通信サーバーは、トランザクション ログ ディレクトリに .par または .sar ファイルを書き込むことがあります。これらは、サーバーがクライアントに送信しようとした最新のアイテムのコンテキストを含むテンポラリ ファイルです。再接続が行われると、クライアントはデータの再送信を再度要求します。サーバーはこれらのファイルを読んで適切なデータを取得します。これらのファイルは、データが読み取られたあと、接続が最終的に終了するときにサーバーが削除します。

設定リファレンス

4

Zen での設定方法およびプロパティ設定

以下のトピックでは、データベース サーバー エンジンやクライアントを設定するさまざまな方法について説明します。

- 「設定の概要」
- 「ZenCC での設定」
- 「bcfg を使用した設定」
- 「サービス設定プロパティ」
- 「全プラットフォームにおけるサーバー設定プロパティ」
- 「Windows クライアント設定プロパティ」
- 「Linux、macOS、および Raspbian クライアント設定プロパティ」
- 「Reporting Engine 設定プロパティ」

設定の概要

Zen はそのデータベース エンジンとクライアントの設定プロパティを使用して構成されます。設定プロパティは Zen Control Center（ZenCC）またはコマンド プロンプトで設定できます。

ZenCC では、設定はエンジンまたはクライアントのプロパティです。「ZenCC でエンジンのプロパティを設定するには」および「ZenCC でローカル クライアントのプロパティを設定するには」を参照してください。

コンポーネントの設定は任意です。設定を行わない場合は、各コンポーネントにはデフォルトの構成が読み込まれます。よい結果を得るためには、クライアントとエンジン コンポーネントで同じバージョンの ZenCC を使用してください。

設定を使用できるのは、以下の理由からです。

- システムまたは Zen アプリケーションが、設定の変更を必要とするため。お使いのアプリケーションのマニュアルで、推奨される値を確認してください。複数のアプリケーションを同時に起動する場合は、それぞれに推奨されている値の合計を使用します。複数のアプリケーションを順次起動する場合は、推奨されている最も高い値を使用します。
- 設定を最適化して Zen が必要以上にメモリを使用せずにサービスを提供できるようにするため

設定自体の説明は、「設定リファレンス」にあります。

設定の変更が有効になっていることを確認する

エンジンの設定によっては、設定内容の変更後にデータベース エンジンの再起動が必要になります。エンジンを再起動することにより、設定が有効になります。この章の各設定の説明では、データベース エンジンの再起動が必要かどうかを明示しています。また、設定の変更によりエンジンの再起動が必要な場合は、ZenCC によりメッセージが表示されます。

CLI ツールも、入力ファイルを使用しないでコマンド ラインから設定を変更したことを知らせます。入力ファイルを使用した場合は、常にエンジンの再起動が要求されます。「bcfg を使用した設定」を参照してください。

コマンド ラインからサーバー データベース エンジンの停止および起動を行う場合は、『*Zen User's Guide*』の以下のトピックを参照してください。

- 「Windows サーバー上での Enterprise Server エンジンの起動と停止」
- 「Linux、macOS および Raspbian 上でのデータベース エンジンの起動と停止」

ワークグループ エンジンの停止および起動を行う場合は、『*Zen User's Guide*』の 「Windows 上での Workgroup エンジンの起動と停止」を参照してください。

さらに、クライアント プロパティを変更すると、クライアントを再起動する必要がある場合があります。クライアントの再ロードは、Zen に依存するすべてのアプリケーションを終了して再起動するだけです。

別のマシンに接続する

ローカル クライアントのコンポーネントだけでなく、ローカル エンジンおよびリモート エンジンの構成も行えます。ただし、エンジンはそれぞれ個別に構成する必要があります。「ZenCC での設定」および 「bcfg を使用した設定」を参照してください。

ZenCC を使ってリモート マシンに接続しているときは、エンジン コンポーネントのみを表示および変更できます。クライアントのコンポーネント（ワークグループ エンジン、ワークステーション エンジン、クライアント マシンなど）は、マシンごとのローカルでしか設定できません。

ZenCC での設定

ZenCC で、設定とはエンジンまたはクライアントのプロパティです。登録済みのすべてのエンジンが Zen エクスプローラーにノードとして表示され、それらのノードを選択することでエンジンのプロパティを設定できます。ただし、クライアントはローカル クライアントしか表示されません。リモート クライアントを設定するには、クライアント マシン上の Zen エクスプローラーでそのクイアントを見つけます。

リモート サーバーを構成するには、サーバーにアクセス可能なクライアント上、またはネットワークにアクセス可能なサーバー上で ZenCC を開き、エンジン ノードの下でそのリモート サーバーを探すか登録して、プロパティを設定します。この方法は、ZenCC がサポートされていない Windows Nano Server や Windows IoT Core、および Raspbian などのオペレーティング システムで役立ちます。

以下の手順では、エンジンとクライアントのプロパティにアクセスする方法を示します。

▶▶ ZenCC でエンジンのプロパティを設定するには

1 Zen エクスプローラーで**エンジン** ノードを展開します。

2 設定を指定するデータベース エンジンを右クリックします。

3 ［**プロパティ**］を選択します。

4 プロパティのカテゴリをクリックして設定のグループを表示します。

選択した設定に関するヘルプ トピックを開くには、F1 キーを押します。設定の全般的な説明は、「全プラットフォームにおけるサーバー設定プロパティ」にあります。

▶▶ ZenCC でローカル クライアントのプロパティを設定するには

1 Zen エクスプローラーで、［**ローカル クライアント**］ノードを展開します。

2 ［MicroKernel **ルーター**］を右クリックします。

3 ［**プロパティ**］を選択します。

4 ツリー内で目的のオプション カテゴリをクリックし、そのカテゴリに含まれるオプションの設定内容を表示します。

選択した設定に関するヘルプ トピックを開くには、F1 キーを押します。プロパティ設定の全般的な説明については、以下のトピックを参照してください。

- 「Windows クライアント設定プロパティ」
- 「Linux、macOS、および Raspbian クライアント設定プロパティ」
- 「Reporting Engine 設定プロパティ」

Windows システムではプロパティをコマンド ラインから設定することもできます。通常、この方法は Linux、macOS、および Raspbian システムでも使用されます。「bcfg を使用した設定」を参照してください。

bcfg を使用した設定

bcfg コマンド ライン ツールは、ZenCC のプロパティ ダイアログと同様の設定機能を提供します。このツールは、Zen でサポートされる Windows、Linux、macOS、および Raspbian プラットフォームで実行します。この実行可能プログラムは、bcfg.exe（Windows の場合）および bcfg（Unix ベースのシステムの場合）です。Windows システムの場合、bcfg は Zen インストールの bin ディレクトリにインストールされます。Unix システムの場合は、/usr/local/actianzen/bin にあります。

以下のトピックでは、bcfg の使用方法について説明します。

コマンド構文

bcfg は、構成の設定を以下のように管理するためのコマンドを提供します。

- 単一の設定の値を読み込む

  ```
  bcfg ID [-S server] [-U username] [-P password] [-JSON]
  ```

- 単一の設定の値を変更する

  ```
  bcfg ID value [-S server] [-U username] [-P password] [-JSON]
  ```

- 現在のすべての設定の出力ファイル（編集して入力用に使用できる）を書き込む

  ```
  bcfg -O outputfile [-S server] [-U username] [-P password] [-E] [-F] [-JSON]
  ```

- 新しい設定の入力ファイルを読み込み、それらを適用する（テキスト入力のみ）

  ```
  bcfg -I inputfile [-S server] [-U username] [-P password] [-E] [-F]
  ```

- キーワードで設定を検索する

  ```
  bcfg -H <keyword | "keywords with spaces"> [-S server] [-U username] [-P
  password] [-JSON]
  ```

オプション

-E	*inputfile* の読み取り時、あるいは *outputfile* への書き込み時のエラーを無視します。
-F	確認を求めずに *outputfile* を上書きします。このフラグを使用しない場合、既存の *outputfile* と stdin の欠如（リモートの PowerShell セッションでの検出など）の組み合わせが原因で、bcfg は失敗し、" 対話型入力でエラーが検出されました。" というメッセージが表示されます。
-H	*keyword* 用のヘルプ検索オプション。*Keyword* を含む設定を検索するためのツールで、ID および設定名を返すか、あるいは " 該当するものが見つかりません " を返します。このオプション一覧の次の項目も参照してください。
-I	ツールの入力をファイルから読み込むのに使用します。ファイルの名前は *inputfile* パラメーターで付けられます。
ID	設定を示す、2 桁または 3 桁の整数。設定のいくつかは、その設定を有効にするためにデータベース エンジンの再起動を必要とします。再起動が必要な場合は、bcfg が再起動を促します。「新しい設定を適用した後のエンジンの再起動」を参照してください。

inputfile	指定したサーバーに関する 1 つまたは複数の設定のレコードと、各設定に割り当てる値を含んでいるテキスト ファイル。-I オプションで使用されます。入力ファイルを作成する便利な方法は、最初に出力ファイルを作成することです。その後、設定値を必要に応じて編集し、その編集したものを入力ファイルとして使用できます。編集が可能な設定の種類については、「入力ファイルの編集」を参照してください。
-JSON	出力（エラー メッセージなど）にテキストではなく JSON 形式を使用するための任意のパラメーター。
keyword	" クライアント保持の資格情報の容認 " または " サポート プロトコル " など、設定の名前。 *Keyword* では、大文字と小文字は区別されません。スペースを含む複数のキーワードを使用する場合は、その文字列を二重引用符で囲みます。 部分的なキーワードを基にヘルプを表示できます。たとえば、「`-H クライアント`」と指定した場合は、設定名の一部に " **クライアント** " という語を含んでいるすべての設定が返されます。Linux 版の場合は設定名が英語なので " クライアント " の部分を "client" に置き換えてください。「`-H a`」と指定した場合は、設定名に "a" を含んでいるすべての設定が返されます。
-O	ツールの出力をファイルへ書き込むために使用します。ファイルの名前は *outputfile* パラメーターで付けられます。
outputfile	指定されたサーバーの 1 つまたは複数の設定が書き込まれるテキストまたは JSON ファイル。-O オプションで使用されます。
-P	*server* へのアクセスにパスワードが必要な場合に必須。*password* パラメーターで提供されます。
password	*server* へ接続する *username* と一緒に使用するパスワード。-P オプションで使用されます。「username」を参照してください。「Zen セキュリティ」も参照してください。
-S	設定がリモート サーバー（ローカル サーバー以外のサーバー）に適用される場合に必須。*server* パラメーターで提供されます。
server	データベース エンジンを含んでいるリモート サーバーの名前、または IP アドレス。-S オプションで使用されます。
-U	*server* へのアクセスにユーザー名が必要な場合に必須。*username* パラメーターで提供されます。
username	*server* に接続するユーザー名。「Zen セキュリティ」も参照してください。-U オプションで使用されます。 *server* がローカル マシンの場合、以下の条件を満たしていれば *username* と *password* は必要ありません。 • 管理者として、あるいは Zen_Admin グループのメンバーとしてローカル マシンにログインしている。 • ローカル マシンがターミナル サービスを実行していない。
value	設定に割り当てる値。有効な値は、設定の「オプション」または「範囲」に示されています。 *value* を省略した場合、ツールは現在の設定を返します。 *value* を記述した場合、ツールはその値に対する設定を変更します。 「シナリオの例：コマンド プロンプトから単一の設定を構成する」を参照してください。

シナリオの例：コマンド プロンプトから単一の設定を構成する

ネットワークが停止した場合に、クライアントがサーバーに再接続するかどうかに関係する設定をオンにしたいとします。しかし、その設定の完全な名前がわかりません。以下の手順の例では、コマンド プロンプトで設定を見つける方法と、その値の設定方法を示します。

1 コマンド プロンプトで「`bcfg -H 再接続`」と入力し、Enter キーを押します。Linux 版の場合は " 再接続 " の部分を "reconnect" に置き換えてください。

ツールによって、文字列 " 再接続 " を含んでいるすべての設定がレポートされます。Linux の場合は英語でレポートされます。

```
ID                  設定名
-----------------------------
29                  自動再接続の有効化
148                 自動再接続の有効化
149                 自動再接続タイムアウト
```

29 と 148 の 2 つが目的の設定であるようですが、必要な設定は一体どちらなのでしょうか？

2 「bcfg 29」と入力して、Enter キーを押します。

ツールは、設定 ID 29 について次のようにレポートします。

```
=====================
自動再接続の有効化
=====================
ID: 29
値 : Off
オプション : On    Off
デフォルト値 : Off
説明 : < クライアント設定 > ネットワーク停止の場合に、クライアントがサーバーへの再接続を試みるかどうかを指定します。再接続されたクライアントは、エラーに遭遇しなかったかのように処理を続行します。
```

説明によると、この設定はクライアントに適用されるものであり、現在 Off に設定されているということです。

3 「bcfg 29 on」と入力して、Enter キーを押します。

ツールは、システム設定 29 が更新されたことを知らせます。

4 設定が On になっていることを確認したい場合は、「bcfg 29」と入力して Enter キーを押します。

ツールは、設定 ID 29 について、値が On に設定されていることをレポートします。

```
=====================
自動再接続の有効化
=====================
ID: 29
値 : On
オプション : On    Off
デフォルト値 : Off
説明 : < クライアント設定 > ネットワーク停止の場合に、クライアントがサーバーへの再接続を試みるかどうかを指定します。再接続されたクライアントは、エラーに遭遇しなかったかのように処理を続行します。
```

入力ファイルの編集

設定は、コマンド ラインから一度に 1 つずつ設定するのではなく、入力ファイルに 1 つ以上を指定して設定することができます。現在のリリースで、この目的に対応するのはテキストのみです。

入力ファイルを作成する便利な方法は、出力ファイルを編集し、それを入力ファイルとして使用することです。入力ファイルには、すべての設定を含める、または変更が必要な設定のみを含めることができます。入力ファイルには、少なくとも 1 つの設定に対する完全なレコードが 1 つは含まれていなければなりません。出力ファイルを基に入力ファイルを作成し、設定を削除する際には、残す設定は完全なレコードの状態であるようにしてください。

ID、値、オプション、または範囲については、その対応する行で設定値を変更できます。上記以外の行へ変更を加えようとすると、エラーになる、設定の変更に失敗する、またはその他予期しない動作が発生する可能性があります。

完全なレコードには、ID と値のペアが少なくとも 1 つ含まれます。見出しから説明までのすべての行を含めることをお勧めします。たとえば、このトピックでのシナリオ例の最後のステップでは「自動再接続の有効化」の設定が完了したので、それを入力ファイルで使用できます。

新しい設定を適用した後のエンジンの再起動

入力ファイルを使用した場合、そのファイルの設定内容にかかわらず、bcfg は設定を有効にするためにデータベース エンジンを再起動するよう促します。

コマンド ラインから構成を行った場合は、bcfg はその設定が再起動を要求する場合にのみ、データベース エンジンの再起動を促します。

エンジンを再起動する手順については、「設定の変更が有効になっていることを確認する」を参照してください。

トラブルシューティング

このツールの実行に問題がある場合は、次の表に示すトラブルシューティングのご提案を参考にしてください。

トラブルシューティングする状態	説明
次のエラーが返された。 データベース エンジンに接続できません。ターゲット マシンがアクセス可能で、かつエンジンが実行されていることを確認してください。	このエラーは、ローカル サーバーを設定しようとした場合に発生します。 ローカル サーバーを設定するには、zen-data グループのメンバーであるか root ユーザーである必要があります。 『*Getting Started with Zen*』の 「Linux、macOS、Raspbian での Zen のアカウント管理」も参照してください。
次のエラーが返された。 対話型入力でエラーが検出されました。ファイルを上書きせずに終了します。	コマンドを実行するときに -F オプションを追加して、プロンプトを出力させない、stdin から読み取りを行わない、および常にファイルを上書きするようにしてください。

サービス設定プロパティ

Windows サーバー環境では、Zen サーバーはサービスとして起動します。サービスはインストール処理の一部として読み込まれ、完全インストールであれば常に使用可能な状態に設定されます。

ZenCC 内でサービスの起動ポリシーを構成することができます。

▶▶ ZenCC でサービスの起動ポリシーを設定するには

1 ZenCC で、**サービス** ノードを展開します。

2 Actian Zen Server Engine を右クリックします。

3 ［**プロパティ**］をクリックします。

4 **スタートアップの種類**として、起動ポリシーを選択します。

ポリシー	説明
手動	サービスは、オペレーティング システムの起動時に自動的に開始されません。オペレーティング システムの起動後または再起動後に、サービスを手動で開始する必要があります。
自動	サービスは、オペレーティング システムの起動時または再起動時に自動的に開始されます。
無効	サービスは、起動ポリシーが［手動］または［自動］に再設定されるまで機能しなくなります。

5 ［OK］をクリックします。

全プラットフォームにおけるサーバー設定プロパティ

各 Zen データベース エンジンには、独自のサーバー設定があります。このトピックでは、各エンジンのオプションについて説明します。オプションはエンジンごとに独立しており、ほかのエンジンと依存関係はありません。

Zen サーバーは、Windows、Linux、macOS、および Raspbian プラットフォームで、ZenCC またはコマンド ライン ツールの bcfg を使用して構成することができます。ZenCC では、データベース エンジン ノードを右クリックしてプロパティ ウィンドウを開きます。コマンド ラインについては、「bcfg を使用した設定」を参照してください。

次の表は、サーバー設定プロパティとその設定項目の一覧をアルファベット順で示します。各設定の詳しい説明がリンク先のセクションにあります。

表 7　サーバー設定プロパティ

設定オプション	設定項目名
「アクセス」	「リモート リクエストの受付」
	「キャッシュ エンジン接続の許可」
	「クライアント保持の資格情報の容認」
	「Authentication（Linux ベースのエンジンのみ）」
	「Configuration File（Linux、macOS、および Raspbian エンジンのみ）」
	「クライアント資格情報の入力要求」
	「記憶域サーバー」
	「ワイヤ暗号化」
	「ワイヤ暗号化レベル」
「通信プロトコル」	「自動再接続タイムアウト」
	「自動再接続の有効化（Windows のみ）」
	「リッスン IP アドレス」
	「サポート プロトコル」
	「TCP/IP マルチホーム」
	「TCP/IP ポート」
「ファイル互換性」	「ファイル サイズの制限」
	「作成ファイルのバージョン」
	「システム データ」

表 7　サーバー設定プロパティ

設定オプション	設定項目名
「データ整合性」	「選択ファイルのアーカイブ ロギング」
	「起動時間制限」
	「オペレーション バンドル制限」
	「トランザクション一貫性保持」
	「トランザクション ログ」
	「ウェイト ロック タイムアウト」
「デバッグ」	「データ バッファーのバイト数」
	「キー バッファーのバイト数」
	「トレースするオペレーションの選択」
	「トレース ファイルのロケーション」
	「トレース オペレーションの実行」
「ディレクトリ」	「一時ファイルの場所」
	「トランザクション ログのディレクトリ」
	「作業ディレクトリ」
	「DBNames 設定ファイルのディレクトリ」
	「TEMPDB ディレクトリ（Client Reporting Engine のみ）」
「情報」	サーバー名（表示のみ） エンジンのバージョン（表示のみ） エンジンのタイプ（表示のみ）
「メモリの使用」	「起動時のリソース割当」
	「非アクティブ時、最小の状態に戻す」
	「最小の状態に戻す待ち時間」
	「ソート バッファー サイズ」
	「システム キャッシュ」

表 7　サーバー設定プロパティ

設定オプション	設定項目名
「パフォーマンス チューニング」	「自動最適化」
	「キャッシュ割当サイズ」
	「通信スレッド数」
	「ファイルを閉じるまでの待ち時間」
	「ファイルの拡張係数」
	「インデックス バランスの実行」
	「セグメント サイズを 2 GB に制限」
	「トランザクション ログ バッファー サイズ」
	「MicroKernel の最大メモリ使用量」
	「I/O スレッド数」
	「トランザクション ログ サイズ」

アクセス

サーバーを右クリックし、[プロパティ] > [アクセス] を選択すると、次の設定が表示されます。

- 「リモート リクエストの受付」
- 「キャッシュ エンジン接続の許可」
- 「クライアント保持の資格情報の容認」
- 「Authentication（Linux ベースのエンジンのみ）」
- 「Configuration File（Linux、macOS、および Raspbian エンジンのみ）」
- 「クライアント資格情報の入力要求」
- 「ワイヤ暗号化」
- 「ワイヤ暗号化レベル」

リモート リクエストの受付

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	Yes

この設定は、通信マネージャーが、リモート サーバーやクライアント ワークステーションからのリクエストを受け付けるかどうかを指定します。このオプションを**オン**にすると、通信マネージャーは、ネットワークでの存在を通知します。

キャッシュ エンジン接続の許可

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	Yes

クライアントがキャッシュ エンジンを使用してサーバーに接続することをサーバーがサポートするかどうかを指定します。**オフ**に設定してもクライアントはサーバーに接続はできますが、キャッシュ エンジンを使用することはできません。

クライアント保持の資格情報の容認

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	No

この設定を**オン**にした場合、データベース エンジンはクライアントに保管されているユーザー資格情報を受け付けます。保管方法および場所は、クライアントのオペレーティング システムによって異なります。

- Windows クライアント：資格情報は Windows レジストリに格納されています。［**クライアント資格情報の入力要求**］が**オン**に設定されている場合は、ポップアップ ダイアログで［**ユーザー名とパスワードを保存**］チェック ボックスをオンにすることによって、資格情報を保存できるようになります。代わりの方法として、pvnetpass コマンド ライン ツールを使用すると、保管されている資格情報を管理することができます。
- Linux、macOS、および Raspbian クライアント：資格情報は、pvnetpass によって Zen レジストリに格納されます。

この設定を**オフ**にした場合、データベース エンジンはクライアントに対し、資格情報が必要なデータベース操作の対象から保管されている資格情報は強制的に除外します。このような資格情報は、アプリケーションから、またはログイン ダイアログで提供する必要があります。この設定がオフであっても、ログイン ダイアログを使用して指定されれば、ログイン ダイアログはクライアント保持の資格情報を書き込もうとします。ただし、これは受け入れられません。

クライアント保持の資格情報が認められている場合は、資格情報を知らなくても、誰でもその特定クライアントのコンピューターに向かい、保管されている資格情報を使ってデータベースにログインすることができます。この動作は、個々のユーザーの厳密な認証は重要でない環境、たとえば、すべてのユーザーが同レベルのアクセス権限を持っている、物理的にセキュリティ保護されているような環境では便利です。一方、権限のない人が存在したり、権限のあるユーザーがアクセス権限のレベルを変更しているような環境では、この設定は**オフ**にしておく必要があります。

「クライアント資格情報の入力要求」も参照してください。

ログイン動作のまとめ

表 8　ログイン設定動作のまとめ

クライアント資格情報の入力要求	クライアント保持の資格情報の容認	動作
オフ	オフ	Zen クライアントは、ユーザーに入力を求めることも保管されている資格情報を使用することもしないため、クライアント アプリケーションが Btrieve オペレーションで資格情報を提供する必要があります。
オフ	オン	資格情報がクライアント アプリケーションの Btrieve オペレーションで提供されない場合、クライアントは、ログイン ダイアログまたは pvnetpass によって保管されている資格情報を利用できる場合は、それを使用します。どちらの方法でも資格情報が提供されない場合は、接続に失敗します。ログイン ダイアログは表示されません。
オン	オフ	資格情報がクライアント アプリケーションの Btrieve オペレーションで提供されない場合、クライアントはユーザーにログイン ダイアログを表示し、Linux、macOS、または Raspbian クライアントはアクセス許可エラーを示すステータス コードを返します。ログイン ダイアログまたは pvnetpass によって保管された資格情報は使用しません。
オン	オン	資格情報がクライアント アプリケーションの Btrieve オペレーションで提供されない場合は、保管されている資格情報を使用します。利用できる資格情報が保管されていない場合、クライアントはユーザーにログイン ダイアログを表示し、Linux、macOS、または Raspbian クライアントはアクセス許可エラーを示すステータス コードを返します。

Authentication（Linux ベースのエンジンのみ）

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
単一選択	3 つのオプションがあります。次の解説を参照。	Emulate Workgroup Engine（ワークグループ エンジンのエミュレート）	なし	Yes

以下のオプションは、サーバー エンジンへのアクセスに使用する認証のタイプを設定します。

- Emulate Workgroup Engine（ワークグループ エンジンのエミュレート）。Samba を使用してシステムのユーザー アクセスを認証する場合は、この値を使用します。オペレーティング システムによるセキュリティを回避し、レジストリに RTSS パスワードを保存したくない場合は、[Emulate Workgroup Engine] を使用します。
- Proprietary Authentication（using btpasswd）（**専用認証（btpasswd の使用）**）。Linux では Samba、macOS では SMB を認証に使用せず、ユーザーがサーバーにアカウントを持っていない場合は、この値を使用します。このオプションを使用すると、Linux、macOS、または Raspbian システムへの接続用パスワード ファイルを別々に保持できます。

  サーバーで BTPASSWD の認証を使用する場合、クライアントからこのサーバーに接続するユーザー名とパスワードを設定する必要があります。Zen Control Center を使用するか、またはコマンド プロンプトで pvnetpass を使用します。『*Zen User's Guide*』の「グループ、ユーザー、およびセキュリティ」および「pvnetpass」の両トピックを参照してください。

  Proprietary Authentication は、サーバーにより高いセキュリティ強度が必要で、サーバーで使用されるユーザー認証方式と異なるユーザー名とパスワードが必要な場合に使用します。

- Standard Linux Authentication（標準 Linux 認証）。認証に Samba を使用せず、ユーザーが Linux、macOS、または Raspbian システムにアカウントを持っている場合、この値を使用します。

  Linux の標準認証は PAM と共に使用されます。PAM は、Linux、macOS、または Raspbian サーバーで既存のユーザー名とパスワードを使用したい場合に使用します。pvnetpass を使用してクライアントからユーザー名とパスワードを指定することができます。PAM は非常に柔軟で、Linux、macOS、および Raspbian 用のカスタム モジュールを多数提供しています。詳細については、PAM に関する Web サイトを確認してください。

  Zen のインストールで PAM が検出された場合、PAM を使用できるようインストールの設定を整えます。Zen のインストール後に PAM をインストールし、PAM による標準認証を使用する場合は、Zen を再インストールする必要があります。これは、PAM のインストールでファイルのコピー、各種設定ファイルの作成、権限の設定、およびリンクの作成を行っているためです。Zen を再インストールして PAM を検出し、その PAM 設定を正しく完了させる必要があります。

  Zen をアンインストールし、再度インストールし直します。これを行う手順については、『*Getting Started with Zen*』の 「Linux、macOS および Raspbian への Zen のインストール」を参照してください。

PVPIPE$ を使用した Samba と認証（Linux のみ）

上記の 3 種類の認証方法に加え、使用可能な場合は、Samba を使用することもできます。「Configuration File（Linux、macOS、および Raspbian エンジンのみ）」で説明されているように PVPIPE$ が共有されている場合、Zen エンジンは FIFO を `$ACTIANZEN_ROOT/etc/pipe/mkde.pip` に作成します。PVPIPE$ は Linux でのみサポートされます。

メモ 末尾のドル記号（$）は、この共有が非表示になることを意味します。Zen クライアント コンポーネントが、¥¥< サーバー >¥PVPIPE$¥mkde.pip（大文字小文字区別なし）として自動的にこのパイプへのアクセスを処理します。明示的な処理を実行したり、このパイプにアクセスするようにアプリケーションを変更したりする必要はありません。唯一の例外は、Samba または Zen の設定のトラブルシューティングを行う場合です。

クライアントがリモート エンジンに接続し、エンジンがバージョン ブロックで "Unix" と返した場合は、まずレジストリ（RTSS 設定）で認証情報を確認します。ユーザー名とパスワードがレジストリ内で見つからなかった場合、クライアントは上記のパイプに接続し、サーバーからクライアント認証情報を受け取り、検証します。

認証されるためには、共有に接続し、パイプを読み取ることができる必要があります。これは、エンジンを使用できるユーザーと使用できないユーザーを指定する 1 つの方法です。これを行う最も簡単な方法は、smb.conf 設定ファイル内の「valid users」値を設定することです。クライアントが認証されなかった場合、ステータス 3119 が返されます。

注意 PVPIPE$ へのクライアントの読み取りアクセスを許可すると、そのクライアントはリモートでエンジンへのアクセスが許可されます。

クライアントが認証されることを確認するには、コマンド プロンプトで「¥¥< サーバー >¥pvpipe$¥mkde.pip」と入力します。多数の疑問符（印刷不可能なシンボル）といくつかの印刷可能な文字が表示され、警告音が鳴ります。表示されなかった場合は、Samba の設定ファイルを調べて、このパイプを読み取る権限があるかどうかを確認します。権限があるのにエラー 94 または 3119 が発生する場合は、Zen Control Center でエンジンの設定プロパティを使用して、または pvnetpass を使用して、RTSS 設定を確認します。

Samba を介して共有されるファイルへのアクセスの詳細については、Samba のマニュアルを参照してください。

Configuration File（Linux、macOS、および Raspbian エンジンのみ）

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	/etc/smb.conf	なし	Yes

この設定は、ローカル ファイル システムを Windows クライアントへエクスポートする場合に使用される smb.conf ファイルの場所を示します。エンジンは、リモート システムの UNC パスを正しいデータベース ファイルへのローカル呼び出しに変換するときに、このファイルを必要とします。サード パーティ製の Samba パッケージではなくネイティブの SMB ファイル共有が使用されている macOS システムや Raspbian システムでは、この設定は適用されないことに留意してください。

デフォルト値は **/etc/smb.conf** です。Samba の設定ファイルを別の場所にインストールした場合は、正しいパス名を入力します。

Zen は smb.conf 設定ファイルの有無について、次に示す場所をこの順序で調べます。

- /etc/samba/smb.conf
- /etc/smb.conf
- /usr/local/samba/lib/smb.conf
- /usr/local/lib/smb.conf
- /lib/smb.conf
- /etc/samba.d/smb.conf
- /opt/samba/lib/smb.conf
- /usr/share/samba/smb.conf
- /usr/local/share/samba/smb.conf
- /home/samba/lib/smb.conf
- /opt/local/etc/samba3/smb.conf

最初に見つかった smb.conf が使用されます。smb.conf が見つからない場合は、Zen はシステム ログ ファイルにエントリを記録し、どの Samba 共有も有効になりません。

Linux の場合、PVPIPE$ の FIFO 共有を使用したいが、smb.conf ファイルにまだ共有が存在しない場合には、次のようにファイルに設定する必要があります。zen-svc ユーザーと zen-data グループは Zen サーバーのインストールにより作成されます。

```
[PVPIPE¥$]
   comment = Zen pipes
   path = /usr/local/actianzen/etc/pipe
   # only members of group zen-data will have access
   valid users = @zen-data
   # Absolutely necessary - prevents caching
   oplocks = no
   level2 oplocks = no
   read only = yes
   browseable = no
```

zen-svc というユーザー名のユーザーがこの共有にアクセスできるようにするには、**smbpasswd** コマンドを特定のパラメーターと一緒に使用する必要があります。-n オプションに関する情報は、Samba のドキュメントをお読みください。アクセスを有効にする準備が整ったら、Zen Enterprise Server または Cloud Server がインストールされているシステムで以下のことを行います。

1 root としてログインします。

2 次のコマンドを実行して、ユーザー zen-svc をローカルの smbpasswd ファイルへ追加します。

```
smbpasswd -a -n zen-svc
```

3 smbd を再起動してこの変更を有効にします。

クライアント資格情報の入力要求

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	No

この設定は、ユーザー認証を必要とするデータベース操作中に利用できる資格情報がない場合、Windows Zen クライアントがユーザーにログイン資格情報の入力を求めるかどうかを決定します。

この設定を**オン**にした場合、ほかの認証資格情報がないときには、ユーザーにログイン ダイアログを表示するよう、エンジンが Windows クライアントに要求します。この設定は、混合またはデータベース セキュリティが有効になっている場合にのみ適用されます。ただし、いかなる状況下でも Linux、macOS、または Raspbian クライアントには適用されません。有効な資格情報が別の手段、たとえば、明示的な Btrieve Login（78）オペレーションやクライアントに保管されている資格情報によって提供される場合は、ログイン ダイアログは表示されません。

ユーザー資格情報を必要とする操作で、エンジンにデータベース コンテキストが指定されていない場合、エンジンは、ユーザーは現在のデータベースへのログインを試みていると見なします。

この設定を**オフ**にして、新しいセキュリティ モデルのいずれかを使用している場合は、ユーザー資格情報をプログラムから提供する（クライアントに保管されている資格情報を提供するか、Btrieve の Login（78）、Open（0）、または Create（14）オペレーションによって提供する）必要があります。そうしないと、ログインの試行は認証エラーで失敗します。

関連項目

「クライアント保持の資格情報の容認」

記憶域サーバー

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	空白	なし	Yes

このプロパティは、Client Reporting Engine がサポートしている Zen サーバーの名前を表すものです。これは Client Reporting Engine がインストールされているシステムで設定されます。Windows では、文字列の大文字小文字は区別されません。この設定は、Zen で Client Reporting Engine にのみ提供されます。

ワイヤ暗号化

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
単一選択	しない 必要な場合 常時	必要な場合	なし	Yes

このプロパティは、指定されたクライアントまたはサーバーがネットワーク通信で暗号化を使用するかどうかを指定します。デフォルト値は "**必要な場合**" です。これは、クライアントまたはサーバーは、通信ストリームの相手側が暗号化を要求している場合にのみ暗号化を使用するという意味です。たとえば、サーバー A は［**ワイヤ暗号化**］値を "**常時**" に設定しており、サーバー B は "**しない**" に設定しているとします。クライアントはこの

値を " **必要な場合** " に設定しています。この場合、クライアントはサーバー A と通信する場合には暗号化を使用しますが、サーバー B と通信する場合には暗号化を使用しません。

次の表は、クライアント値とサーバー値の各組み合わせを基に動作をまとめています。

表 9　クライアント / サーバーのワイヤ暗号化設定の組み合わせの結果

クライアント設定	サーバー設定：" しない "	サーバー設定：" 常時 "	サーバー設定：" 必要な場合 "
しない	暗号化を使用しません。	ステータス コード 5001	暗号化を使用しません。
常時	ステータス コード 5000	暗号化を使用します。レベルは、クライアントとサーバー間で最も高い［**ワイヤ暗号化レベル**］設定によって決定されます。	暗号化を使用します。レベルは、クライアントの［**ワイヤ暗号化レベル**］設定によって決定されます。
必要な場合	暗号化を使用しません。	暗号化を使用します。レベルは、サーバーの［**ワイヤ暗号化レベル**］設定によって決定されます。	暗号化を使用しません。

ワイヤ暗号化レベル

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
単一選択	低 中 高	中	なし	Yes

この設定では、暗号化された通信で使用する暗号化キーの長さを指定します。次の表は、使用できるレベルを示しています。

表 10　暗号化レベル値の意味

値	説明
低	40 ビット暗号化キーを使用します
中	56 ビット暗号化キーを使用します
高	128 ビット暗号化キーを使用します

128 ビット長のキーを使用する暗号化は、一般に「高度」暗号化と認められています。ほかの設定は、そのレベルに応じて保護力は低下しますが、パフォーマンスは向上します。ある程度のレベルの暗号化は必要であるが、より良いパフォーマンスを得るために、抑止レベルが低下することを受け入れるつもりである場合に適しています。

クライアントとサーバーの両方が暗号化を要求し、一方が他方よりも強度の高い暗号化レベルを指定した場合、2 つのエンティティは強度の高いレベルを使って通信します。

通信プロトコル

［通信プロトコル］には次の設定が含まれています。

- 「自動再接続タイムアウト」
- 「自動再接続の有効化（Windows のみ）」
- 「リッスン IP アドレス」
- 「サポート プロトコル」

- 「TCP/IP マルチホーム」
- 「TCP/IP ポート」

自動再接続タイムアウト

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	45 - 65535	180	秒	Yes

この設定は、接続不能となる前にクライアントがサーバーにどれだけの時間接続を試行するかを指定します。自動再接続が有効になっているクライアントが、最初に自動再接続が有効なサーバーに接続したとき、サーバーはこの値をクライアントに通知し、両方のコンポーネントがネットワーク中断イベントでどれだけの時間再接続を試行するかがわかるようにします。

自動再接続の有効化（Windows のみ）

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

この設定は、ネットワークの停止時にクライアントが自動的に再接続を試行することをサーバーにサポートさせるかどうかを指定します。**オン**に設定すると、自動再接続が有効になります。

自動再接続は、この設定をクライアント側の設定でも有効にしておかない限り、そのクライアント接続では有効になりません。

接続不能となる前にクライアントがサーバーにどれだけの時間再接続を試行するかを指定します。前述の**自動再接続タイムアウト**を参照してください。

リッスン IP アドレス

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	有効な IP アドレス（単独または複数）。複数のアドレスの場合は各アドレス間をカンマで区切ります。	0.0.0.0	なし	Yes

このオプションは、［TCP/IP マルチホーム］が**オフ**の場合に MicroKernel が受信待ちする IP アドレス（1 つまたは複数）を指定します。このオプションは、［TCP/IP マルチホーム］が**オン**の場合は無視されます。

複数の IP アドレスを指定できますが、その場合は各アドレス間をカンマで区切る必要があります。このアドレス文字列は IPv4 アドレスと IPv6 アドレスの組み合わせも可能です。Zen でサポートされる IPv6 アドレス形式が使用できます。『*Getting Started with Zen*』の「ドライブ ベースの形式」を参照してください。

サポート プロトコル

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
複数選択	1 つ以上	TCP/IP	なし	Yes

この設定では、データベース エンジンがクライアント接続の受信待ちをするプロトコルを指定します。2 つ以上のプロトコルを選択した場合、エンジンはそれらすべてで受信待ちしながらセッションを開始します。最初に接

続したプロトコルが、そのセッションの間使用されます。サーバーとクライアントの両方で有効なプロトコルが最低 1 つないと、通信を行うことができません。

メモ デフォルトは TCP/IP です。現在、このオプションのみがサポートされています。

Linux、macOS、および Raspbian

Linux、macOS、および Raspbian 上の Zen でサポートされるプロトコルは TCP/IP のみです。したがって、サポート プロトコルの設定は、これらの環境では使用できません。

TCP/IP マルチホーム

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	Yes

このオプションは、データベース エンジンがすべてのネットワーク インターフェイスでクライアント接続の受信待ちをするかどうかを指定します。**オン**に設定した場合、データベース エンジンはすべてのネットワーク インターフェイスで受信待ちをし、［リッスン IP アドレス］オプションの IP アドレスは無視されます。この設定を**オフ**にする場合は、クライアント通信に使用するデータベース エンジンへのアドレスを［リッスン IP アドレス］に指定する必要があります。

TCP/IP ポート

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	256 ～ 65535	1583	なし	Yes

これは、リレーショナル エンジンで使用されるポート番号を設定します。

このポート番号は、このサーバーを指定するクライアント DSN に定義されたものと同じ番号である必要があります。クライアント DSN でポート番号を変更する方法については、『*ODBC Guide*』の「詳細な接続属性」を参照してください。

ポートに関する詳細については、『*Getting Started with Zen*』の「デフォルトの通信ポートの変更」を参照してください。

ファイル互換性

［ファイル互換性］には次の設定が含まれています。

- 「ファイル サイズの制限」
- 「作成ファイルのバージョン」
- 「システム データ」

ファイル サイズの制限

このチェック ボックスをオンにすると、バージョン 13.0 以降のファイルがバージョン 9.5 のファイルの最大サイズ 256 GB を超えないようにすることができます。

作成ファイルのバージョン

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
単一選択	6.*x* - 8.*x*、9.0、9.5、13.0	9.5	なし	No

この設定はファイルを新規作成するときに使用する形式を指定します。ただし、既に作成済みのファイルには影響しません。ファイルのバージョンが 6.*x* 以降であれば、ファイル形式を変更することなく読み込みと書き込みができます。このため、8.*x* ファイルは 8.*x* ファイル形式のままで、7.*x* ファイルは 7.*x* 形式のままで、そして 6.*x* ファイルは 6.*x* 形式ままで状態が保持されます。5.*x* ファイル以前は例外です。これらのファイルは読み込むことはできますが、最初にリビルドして新しいバージョンの形式に変換しない限り書き込みはできません。

6.*x*、7.*x*、または 8.*x* は、旧バージョンの MicroKernel と互換性を保つ必要がある場合にのみ選択します。

メモ 辞書ファイル（DDF）は 6.*x* またはそれ以降のファイル形式で作成する必要があります。**データベースの新規作成**ウィザードは［作成ファイルのバージョン］設定を使用します。データ ファイルは、サポートされている以前のファイル形式のいずれでもかまいません。DDF だけは 6.*x* またはそれ以降のファイル形式を使用する必要があります。

システム データ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
単一選択	次の解説を参照。	必要な場合	なし	No

システム データは各レコード内の隠されている重複のないキーを参照します。MicroKernel はトランザクション一貫性保持を確実にするのに行を一意に識別できることが必要なため、ファイルは重複のないキーを定義するかまたはシステム データを含める必要があります。デフォルト値は［**必要な場合**］です。使用可能な値は、次のとおりです。

- ［**なし**］。デフォルトでは、システム データはファイル作成時に追加されません。Create オペレーションを使用するアプリケーション開発者はこの設定を無効にして変更することができます。
- ［**必要な場合**］。ファイルが、重複のないキーを持たない場合、ファイル作成時にそのファイルにシステム データが追加されます。
- ［**常時**］。ファイルが、重複のないキーを持つかどうかに関係なく、システム データが、ファイル作成時に常に追加されます。

メモ システム データ設定の変更は既存のファイルには影響しません。この設定は新規ファイルがどのように作成されるかにのみ影響します。

［システム データの作成］を有効にしないで作成され、重複のないキーを持たないファイルにトランザクション一貫性保持を適用したい場合は、［システム データの作成］を［**常時**］または［**必要な場合**］に設定した後、ファイルを再作成する必要があります。

リレーショナル エンジンは常に、システム データを含むファイルを作成します。この情報は、SQL、JDBC、また Btrieve API 以外のあらゆる方法で作成されたファイルに当てはまります。

この設定は、ログ用に Zen 標準システム データを導入します。現在、システム データ v2 は作成しません。

インデックスはユーザーによって削除されることがあるため、ファイルが重複のないキーを持っている場合でも、システム データを追加したい場合があります。

データ整合性

［データ整合性］には次の設定が含まれています。

- 「選択ファイルのアーカイブ ロギング」
- 「起動時間制限」
- 「オペレーション バンドル制限」
- 「トランザクション一貫性保持」
- 「トランザクション ログ」
- 「ウェイト ロック タイムアウト」

選択ファイルのアーカイブ ロギング

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

この設定は、ファイルのバックアップを容易にするアーカイブ ログを MicroKernel に実行させるかどうかを指定します。システム障害が発生した場合、アーカイブ ログ ファイルおよび butil -rollfwd コマンドを使用して、最後にバックアップを取ったときからシステム障害が発生するまでの間にファイルに加えられた変更を修復できます。

MicroKernel にアーカイブ ログを実行させるには、アーカイブ ログを実行するファイルのあるボリュームにアーカイブ ログ設定ファイルを作成し、エントリを追加することによりファイルを指定する必要があります。アーカイブ ログの詳細については、「アーカイブ ログおよび Continuous オペレーションについて」を参照してください。

起動時間制限

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	1 - 1800000	10000	ミリ秒	No

この設定は、システム トランザクションを起動させるタイム リミットを指定します。MicroKernel は、［**オペレーション バンドル制限**］か［起動時間制限］のいずれかで設定された値に達した場合、またはキャッシュの再利用が必要な場合に、システム トランザクションを起動します。

オペレーション バンドル制限

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	1 ～ 65535	65535	なし	No

このオプションは、システム トランザクションを起動するのに必要なオペレーション（1 つの任意のファイルで実行）の最大数を指定します。MicroKernel は、［オペレーション バンドル制限］か［**起動時間制限**］のいずれかで設定された値に達した場合、またはキャッシュの再利用が必要な場合に、システム トランザクションを起動します。

MicroKernel Database エンジンでは、各ユーザー トランザクション（Begin Transaction から End Transaction または Abort Transaction まで）が 1 つのオペレーションとして扱われます。たとえば、Begin Transaction オペレーション

と End Transaction オペレーションの間に 100 の Btrieve オペレーションがある場合、Begin Transaction オペレーションと End Transaction オペレーションを含む 102 の Btrieve オペレーションが、1 つのオペレーションとして扱われます。

トランザクション一貫性保持

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

トランザクション一貫性保持は、システム障害時に正常終了していたトランザクションがデータ ファイルにコミットされることを保証する点以外は**トランザクション ログ**と同じです。

トランザクション ログとトランザクション一貫性保持の詳細については、「トランザクション ログおよびトランザクション一貫性保持」を参照してください。

メモ ［トランザクション一貫性保持］をオンにしている場合、一部のファイルで、トランザクションの一貫性が保持されないことがあります。ファイルは、少なくとも 1 つの重複のないキーを含んでいるか、ファイル作成時に［**システム データ**］設定が " 常時 " または " 必要な場合 " になっている必要があります。そうしないと、ファイルに対する変更はトランザクション ログに書き込まれません。トランザクション一貫性保持およびシステム データの詳細については、『*Zen Programmer's Guide*』を参照してください。

システム データ設定の変更は既存のファイルには影響しないため、重複のないキーを持たず、［システム データの作成］を有効にしないで作成したファイルは再作成する必要があります。これらのファイルを再作成する前に［システム データの作成］を必ず有効にしてください。

注意 ゲートウェイ ロケーター ファイルにより、同一ファイル サーバー上の異なるディレクトリにある複数のファイルを異なるエンジンが管理することができます。データベースが異なるディレクトリにある複数のファイルを含んでいる場合は、データベース内のすべてのデータ ファイルを同一データベース エンジンで管理する必要があります。同一データベース内のファイルを複数のデータベースで管理する場合、データベースの整合性およびトランザクション アトミシティは保証されません。起こり得る問題を回避する方法については、「リダイレクト ロケーター ファイルの作成」を参照してください。

関連する設定

詳細については、「**トランザクション ログ**」にある同様の「関連する設定」を参照してください。

トランザクション ログ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	Yes

この設定は、データ ファイルに影響するすべてのオペレーションをログに記録することにより、MicroKernel がトランザクションのアトミシティを確実にするかどうかを制御します。

関連する［**トランザクション一貫性保持**］の設定がオンになっている場合、ロギングは自動的に行われ、［**トランザクション ログ**］設定は無視されます。

トランザクション ログとトランザクション一貫性保持の詳細については、「トランザクション ログおよびトランザクション一貫性保持」を参照してください。

メモ ［トランザクション ログ］をオンにしている場合、一部のファイルで、トランザクションの一貫性が保持されないことがあります。ファイルは、少なくとも 1 つの重複のないキーを含んでいるか、ファイル作成時に［**システム データ**］設定が " 常時 " または " 必要な場合 " になっている必要があります。そうしないと、ファイルに対する変更はトランザクション ログに書き込まれません。トランザクション一貫性保持およびシステム データの詳細については、『*Zen Programmer's Guide*』を参照してください。

システム データ設定の変更は既存のファイルには影響しないため、重複のないキーを持たず、［システム データの作成］を有効にしないで作成したファイルは再作成する必要があります。これらのファイルを再作成する前に［システム データの作成］を必ず有効にしてください。

注意 使用するデータベースがデータ ファイル間のトランザクション アトミシティを必要としない場合を除いては、トランザクション ログの設定をオフにしないでください。トランザクション ログがオフに設定されている場合、複数ファイルのデータベースのデータの整合性は保証されません。

［トランザクション ログ］の設定をオフにすることを、使用するアプリケーションのベンダーがサポートしていない場合は、オフにしないでください。

関連する設定

サーバー設定の［**トランザクション一貫性保持**］はトランザクション ログに似ていますが、より高レベルのデータ安全性を提供し、パフォーマンスは低レベルとなります。サーバーの設定にある［**トランザクション ログ バッファー サイズ**］および［**トランザクション ログ サイズ**］は**トランザクション ログ**と関連しています。［**トランザクション ログ バッファー サイズ**］を使用すると、トランザクション一貫性保持とパフォーマンスとのバランスを構成することができます。ログ バッファーが大きいほどディスクに書き込まれる回数が少ないため、パフォーマンスが向上します。ただし、ログ バッファー内のデータベースの変更はシステム障害が起こった場合保守されません。

［**トランザクション ログ サイズ**］は、新しいセグメントを開始する前に各ログ ファイル セグメントがどれだけのサイズになることができるかを制御します。

Btrieve または SQL トランザクションが使用されない場合は、これらすべての設定は無視されることに注意してください。

ウェイト ロック タイムアウト

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 - 2147483647	30000	ミリ秒	Yes

レコード ロックの競合が発生した場合、データベース エンジンとそのクライアントは同等の再試行メカニズムを使用します。エンジンは、ウェイト ロック タイムアウト時間内に要求されたレコードのロックを取得できなかった場合には、該当するステータス コードと共に制御をアプリケーションに返します。

ウェイト ロック タイムアウトの利点

ウェイト ロック タイムアウト設定は、ロックの競合が発生した場合に次のような利点をもたらします。

- データベース エンジンは、ロックの解除を待つ間にもリクエストの処理を続行できる

- 複数のスレッドがロックされたリソースを待っている場合、スレッド キューの能力が向上する
- ネットワーク トラフィックが軽減することによって、ネットワークのパフォーマンスが向上する

ウェイト ロック タイムアウトを適用する状況

ウェイト ロック タイムアウトが適用されるのは、次の 2 種類のアプリケーションのみです。

- リレーショナル エンジンを使用するアプリケーション。
- 再試行する必要がない 1 つの変更操作を実行する Btrieve アプリケーション。このようなアプリケーションは、1 秒またはウェイト ロック タイムアウト値のうち、いずれか短い方の時間内にロック エラーを受け取ります。

通常、ウェイト ロック タイムアウトは、Windows、Linux、macOS、または Raspbian 上の Zen クライアントを介して MicroKernel エンジンを使用する Btrieve アプリケーションには適用されません。その代りに、このようなアプリケーションは次のいずれかの操作を行います。

- " ノーウェイト " ロック バイアス (200, 400) 付きの読み取り操作、または " 書き込みノー ウェイト " ロック バイアス（500）が適用されている書き込み操作の場合、ページまたはロックの競合エラーを直ちに受け取ります。
- " 書き込みノー ウェイト " ロック バイアス（500）が適用されて**いない**非トランザクショナル書き込み操作の場合、1 秒またはウェイト ロック タイムアウト値のうち、いずれか短い方の時間内にページまたはロックの競合エラーを直ちに受け取ります。
- 関連する操作が、" ウェイト " ロック バイアス（100, 300）付きの読み取り操作の場合は、無限に待機します。
- 関連する操作が、トランザクション内の書き込み操作であり、かつ、" 書き込みノー ウェイト " ロック バイアス（500）がその操作またはトランザクションに**適用されていない**場合は、無限に待機します。

ページまたはロックの競合エラーを受け取った際、アプリケーションは、再試行、待機、または他のオプションのどの方法でその競合を処理するか判断することができます。

ページ ロックへの対処

MicroKernel エンジン API は、レコード ロック状況を処理するための制御を提供します。ここでは、制御のメカニズムを簡単に説明します。

- 明示的レコード ロック - 読み取り操作にロック バイアス（100、200、300 または 400）を加えると、レコードの読み取り時にウェイトするかどうかを指定できます。これらのバイアスは Begin Transaction オペレーションに適用することもできます。
- トランザクション中の暗黙ページ ロック - ほとんどのページ ロックの競合は行レベルのロックであるために回避されますが、ロック バイアス 500 を加算すれば、トランザクションがページ ロックの発生によってウェイト状態になることを避けられます。
- 排他ファイル ロック - 明示的ファイル ロックを避けるために並行トランザクションを使用します。ファイルを排他的に開けなかった場合、リクエストは常に直ちに返されます。

トランザクション ロックの詳細については、『*Zen Programmer's Guide*』のほかに、『*Btrieve API Guide*』に記載されているさまざまなオペレーションのロック バイアス手順を参照してください。

デバッグ

［デバッグ］には次の設定が含まれています。

- 「データ バッファーのバイト数」
- 「キー バッファーのバイト数」
- 「トレースするオペレーションの選択」
- 「トレース ファイルのロケーション」
- 「トレース オペレーションの実行」

データ バッファーのバイト数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 - 65535	128	バイト	No

この設定は、MicroKernel がトレース ファイルに書き込むデータ バッファーのサイズを指定します。この設定を使用するには、[トレース オペレーションの実行] 設定を**オン**にしておく必要があります。指定するサイズは、必要とするトレースのレベル（データ全体のバッファーの内容を確認する必要があるのか、またはあるレコードを特定できるだけのバッファー内容があれば十分なのかなど）に左右されます。

キー バッファーのバイト数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 - 255	128	バイト	No

この設定は、MicroKernel がトレース ファイルに書き込むキー バッファーのサイズを指定します。この設定を使用するには、[トレース オペレーションの実行] 設定を**オン**にしておく必要があります。指定するサイズは、必要とするトレースのレベル（キー全体のバッファー内容を確認する必要があるのか、またはあるキーを特定できるだけのバッファー内容があれば十分なのかなど）に左右されます。

トレースするオペレーションの選択

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
複数選択	次 の 解 説 を参照。	すべて	なし	No

トレースされる Btrieve API オペレーション コードにはチェックマークが付けられています（**選択されています**）。リストからトレースするオペレーションを選択します。

- Abort Transaction（21）
- Begin Transaction (19)
- Clear Owner (30)
- Close (1)
- Create（14）
- Create Index (31)
- Delete (4)
- Drop Index (32)
- End Transaction (20)
- Extend(16)
- Find Percent (45)
- Get By Percent (44)
- Get Position (22)
- Get Previous (7)
- Get Previous Extended (37)
- Insert (2)
- Insert Extended (40)
- Open (0)
- Reset (28)
- Set Directory (17)
- Set Owner (29)
- Stat (15)
- Step First (33)
- Step Last (34)

- Abort Transaction（21）
- Get Direct/Chunk (23)
- Get Directory (18)
- Get Equal（5）
- Get First（12）
- Get Greater (8)
- Get Greater or Equal (9)
- Get Last（13）
- Get Less or Equal (11)
- Get Less Than (10)
- Get Next (6)
- Get Next Extended (36)
- Get Position (22)
- Step Next (24)
- Step Next Extended (38)
- Step Previous
- Stop (25)
- Step Previous Extended (39)
- Unlock (27)
- Update (3)
- Update Chunk (53)
- Version (26)

トレース ファイルのロケーション

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	*file_path*¥Zen¥bin¥mkde.tra	なし	No

この設定は、MicroKernel がトレース情報を書き込むトレース ファイルを指定します。ファイル名には、ドライブまたはボリュームの指定およびパスが含まれていること、もしくは UNC パスを使用していることが必要です。トレース ファイルをデフォルトのロケーションに配置しない場合は、別のパスやファイル名を入力します。

Zen ファイルのデフォルトの保存場所については、『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

メモ ODBC トレースと MicroKernel トレースに同じトレース ファイル名を使用しないでください。

トレース オペレーションの実行

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	No

この設定は、各 Btrieve API 呼び出しをトレースし、その結果をファイルに保存することを可能にするトレース機能を有効にするかどうかを指定します。開発者はトレース機能を使用して、アプリケーションのデバッグを行えます。MicroKernel は、強制書き込みモードを使用してトレース ファイルに書き込みを行うため、MicroKernel が正常にアンロードされなかった場合でも、データはファイルに書き込まれます。MicroKernel のパフォーマンスは、リクエストを受け取る頻度によって大幅に影響を受けることがあります。このオプションを有効にする場合は、［トレース ファイルのロケーション］を指定してください。

メモ トレースを開始または終了するためにエンジンを再起動する必要はありません。トレースのオン / オフは実行中に行ってエンジンに直接変更を適用することができます。PCC から、変更を有効にするためにエンジンを再起動するようにメッセージが表示されても、この設定については無視してかまいません。

ディレクトリ

［ディレクトリ］には次の設定が含まれています。

- 「一時ファイルの場所」
- 「トランザクション ログのディレクトリ」
- 「作業ディレクトリ」
- 「DBNames 設定ファイルのディレクトリ」
- 「TEMPDB ディレクトリ（Client Reporting Engine のみ）」

一時ファイルの場所

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	環境による	なし	No

このプロパティは、Zen がクエリの処理中に一時ファイルを書き込むことができるディレクトリを設定します。詳細については、『*SQL Engine Reference*』の「テンポラリ ファイル」を参照してください。

トランザクション ログのディレクトリ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	環境による	なし	Yes

この設定は、MicroKernel がトランザクション ログを保存するために使用するロケーションを指定します。これは有効なパスであり、ドライブまたはボリュームの指定か UNC パスが含まれている必要があります。デフォルトはオペレーティング システムによって異なります。

エンジンは、「トランザクション一貫性保持」または「トランザクション ログ」の設定がオンでない場合、この設定を無視します。

注意 複数のデータベース エンジンで同一のディレクトリを使用しないでください。たとえば、2 つ以上のエンジンのトランザクション ログ ディレクトリとしてリモート サーバーのディレクトリを設定すると便利だと思われるかもしれません。しかし、エンジンは、ログのロール フォワードを実行する必要が生じた場合、どのトランザクション ログ セグメントがどのエンジンに属しているかを判断できません。

データベース エンジンを頻繁に使用すると見込まれる場合は、データ ファイルが存在するのとは物理的に別のボリュームにトランザクション ログが保持されるようにシステムを構成する必要があります。高負荷の下では、単一ドライブで I/O 帯域幅を競う代わりに、ログの書き込みとデータ ファイルの書き込みを別のドライブに分ける方が一般的にパフォーマンスがよくなります。トランザクション ログの詳細については、「トランザクション ログおよびトランザクション一貫性保持」を参照してください。

作業ディレクトリ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	データ ファイルと同一ディレクトリ	なし	Yes

この設定は、多数のインデックスを作成するようなオペレーションで、テンポラリ ファイルを保存するために使用される MicroKernel 作業ディレクトリのロケーションを指定します。ディスク容量に制限がある場合は、このオプションを使用して十分な空き容量があるボリュームに作業ディレクトリを指定できます。

デフォルト値はありませんが、作業ディレクトリを指定しない場合、データ ファイルのロケーションがデフォルトになります。作業ディレクトリを指定するには、[**作業ディレクトリ**] フィールドにパスを入力します。パスには、ドライブまたはボリュームの指定か UNC パスが含まれている必要があります。

DBNames 設定ファイルのディレクトリ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	環境による	なし	Yes

この値は、データベース名設定ファイルの別のロケーションへのパスを設定します。

Zen サーバー エンジンの場合、ディレクトリ パスではなくローカル ファイルのパスです。ワークグループ エンジンの場合、ワークグループ MicroKernel にアクセス可能なリモート パスを指定できます。デフォルト値はオペレーティング システムによって異なります。

- Windows プラットフォーム：*application_data_directory*¥
- Linux、macOS、または Raspbian サーバー：/usr/local/actianzen/etc

設定ファイルをデフォルトのロケーションに配置しない場合は、有効なパスを入力します。

TEMPDB ディレクトリ（Client Reporting Engine のみ）

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	適用外	環境による	なし	Yes

このプロパティでは、Client Reporting Engine が、記憶域サーバーからデータをキャッシュするための一時的なデータベースを作成する場所を設定します。これは有効なパスであり、ドライブまたはボリュームの指定か UNC パスが含まれている必要があります。デフォルトはオペレーティング システムによって異なります。この設定は、Client Reporting Engine でのみ提供されます。

情報

情報には、次のような表示のみの項目が示されます。

表示のみの項目	説明
サーバー名	データベース エンジンが実行されているマシンの名前。
エンジンのバージョン	データベース エンジンのリリース バージョン。
エンジンの種類	データベース エンジンの製品カテゴリ。

メモリの使用

［メモリの使用］には次の設定が含まれています。

- 「起動時のリソース割当」
- 「非アクティブ時、最小の状態に戻す」
- 「最小の状態に戻す待ち時間」
- 「ソート バッファー サイズ」
- 「システム キャッシュ」

起動時のリソース割当

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

この設定は、MicroKernel の起動時に、スレッドやメモリ バッファーを含むリソースを割り当てるように MicroKernel に指示します。起動時のリソース割り当てには L1 キャッシュに加えてバックグランド スレッドも含まれます。必要に応じて、Zen コンポーネントにより、自動的にリソースが割り当てられます。このため、ほとんどの場合、この設定はオフ（デフォルト）にすることができます。

この設定がオフの場合、最初のオペレーションが要求されるまでリソースは割り当てられません。お使いのシステムが多数のユーザーをサポートする場合は、この設定をオンにする方が適切です。

この設定を初めてオンにしたときに、Windows の動作方法のため、メモリの割り当てに顕著な違いが見られないかもしれません。MicroKernel がその L1 キャッシュを割り当てる場合、Windows オペレーティング システムはメモリのページを確保しますが、それらを MicroKernel にコミットしません。その後、MicroKernel が実際にキャッシュ メモリにアクセスすると、Windows は物理ページをコミットするので、Zen コンポーネントのメモリ使用量が増加します。

Windows タスク マネージャーで仮想メモリ サイズを見ると、L1 キャッシュがアクセスされたときにメモリの値が変化することがわかります。バックグランド スレッドがアクセスされたときにスレッド数が変わることもわかります。

非アクティブ時、最小の状態に戻す

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

この設定により、アクティブなクライアントが存在しない状態で一定の時間が過ぎると、MicroKernel は、大部分のメモリ容量およびスレッド リソースをシステムに解放し、最小の状態に戻ります。その時間は、［**最小の状態に戻す待ち時間**］の値で設定します。ほかのクライアントがアクティブになった場合は、MicroKernel により、リソースが再度割り当てられます。

最小の状態に戻す待ち時間

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 - 2147483647	300000	ミリ秒	Yes

この値は、MicroKernel が最小の状態（MicroKernel 起動時の初期状態）に戻る前に非アクティブ状態でどれだけ待機するかを設定します。この状態に戻ると、MicroKernel はメモリとスレッドのリソースをシステムに解放します。MicroKernel を最小の状態に戻したくない場合もあります。たとえば、繰り返し MicroKernel を使用するバッチ ファイルを使用中の場合などです。ほかのクライアントがアクティブになった場合は、MicroKernel により、リソースが再度割り当てられます。

この設定は、[**非アクティブ時、最小の状態に戻す**] に**オフ**（デフォルト）が設定されている場合は無視されます。

ソート バッファー サイズ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 - メモリによって制限される	0	バイト	Yes

この設定は、ランタイムでインデックスを作成中に MicroKernel がソートのために動的に割り当てる、または解放するメモリの最大容量（キロバイト単位）を指定します。

ソート用に指定されたサイズ以上のメモリが必要な場合、または利用可能な処理メモリの 60 % 以上を占める場合は、MicroKernel により、テンポラリ ファイルが作成されます。処理に必要な利用可能メモリの量は、動的な値で、システムの設定や負荷によって変化します。0 キロバイトを指定すると、MicroKernel は利用可能なメモリのうち最大 60 % まで、必要なだけのメモリを割り当てます。

システム キャッシュ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ（Server） オン（Workgroup）	なし	Yes

このオプションは、MicroKernel が、MicroKernel 自体の**キャッシュ割当サイズ**に加え、設定プロパティで指定したシステム キャッシュを使用するかどうかを指定します。

L2 キャッシュを使用している場合は、[**システム キャッシュ**] をオフに設定する必要があります。「MicroKernel の最大メモリ使用量」の設定を確認してください。[MicroKernel の最大メモリ使用量] にゼロより大きい値が設定されている場合は、L2 キャッシュを使用しています。

L2 キャッシュを使用していない場合は [**システム キャッシュ**] をオンにすると、パフォーマンスが向上します。MicroKernel は書き出すページを構成したりグループ化するのにシステム キャッシュを使用します。システム キャッシュがより効果的にページをディスクに書き出せるようにフラッシュを十分に遅らせます。ただし、サーバーにキャッシュ付きのディスク アレイがある場合は [**システム キャッシュ**] の設定を**オフ**にすることにより、より良いパフォーマンスを得られることがあります。

Windows サーバーの場合のみ、Windows パフォーマンス モニター ツールのページング ファイルおよびプロセス オブジェクトを使用して Windows のシステム キャッシュが効率的に使用されているかをチェックできます。zenengnsvc.exe インスタンスの場合は、ページ ファイル オブジェクトの [% 使用] および [% 最大使用] をモニターし、プロセス オブジェクトの [Page Fault/sec] および [Page File Bytes] をモニターします。

パフォーマンス チューニング

［パフォーマンス チューニング］には次の設定が含まれています。

- 「自動最適化」
- 「キャッシュ割当サイズ」
- 「通信スレッド数」
- 「ファイルを閉じるまでの待ち時間」
- 「ファイルの拡張係数」
- 「インデックス バランスの実行」
- 「セグメント サイズを 2 GB に制限」
- 「トランザクション ログ バッファー サイズ」
- 「MicroKernel の最大メモリ使用量」
- 「I/O スレッド数」
- 「トランザクション ログ サイズ」

自動最適化

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	No

この設定は、自動最適化をオンまたはオフにします。オンの場合には、この機能は次のように動作します。

- 設定がオンになった 1 時間後から、エンジンは最適化対象のファイルの検出を開始します。
- エンジンはまず、前回エンジンが再起動した以降にアクティブになったファイルを識別します。
- エンジンは次に、残りのファイルについて、以下の条件のいずれかを満たしているか調べます。

分析	以上
断片化	15%
未使用	15%
順序不同	5%

  これらの条件は、「ウォッチ リスト」トピックで定義されたものと同じです。自動最適化の利点として、ファイルの選択や［ウォッチ リスト］への追加を手動で行う必要がなく、また、その後ファイルの確認や最適化も手動で行う必要がありません。
- これらの条件のいずれかを満たす最初のファイルから、最適化が直ちに開始されます。
- エンジンは 1 度に 1 つのファイルを最適化します。1 つのファイルの処理が終わると、次のファイルの最適化を始めるまでに 1 分間待機します。
- 効率化のため、次の条件を満たすファイルは対象から外れます。
  - 24 時間以内に最適化されている
  - サイズが 10 MB 未満

  このようなファイルを最適化する場合には、［ウォッチ リスト］にファイルを追加して手動で監視するようにするか、または dbdefrag コマンド ライン ツールを使用します。
- 自動最適化の条件を満たすすべてのファイルの最適化が終了すると、エンジンは次の回のファイル検出を始めるまでに 60 分間待機します。

- zen.log のメッセージには、ファイルが自動で最適化されたか手動で最適化されたかが記されます。

メモ アクティブなファイルが閉じられ、キャッシュからそのページが削除された場合、エンジンはそのファイルを自動最適化の候補と見なさなくなります。次にファイルが開かれたとき、エンジンは再度それを検出します。

データの最適化の詳細については、「データ ファイルの断片化の監視」を参照してください。

キャッシュ割当サイズ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	1 MB からメモリによって制限される量	最初の起動時に計算されるサーバー値。ほかのエディションの場合は 64 MB。	MB	Yes

このプロパティでは、MicroKernel によって割り当てられるレベル 1 キャッシュのサイズを指定します。MicroKernel では、データ ファイルへのアクセスにこのキャッシュが使用されます。

一般的に言うと、キャッシュ割当サイズの値がシステム上の物理メモリの 40 % より小さく、設定プロパティの［MicroKernel の最大メモリ使用量］が 40% より大きい値に設定されている場合に、全体のパフォーマンスは最高になります。最適な設定は、データ ファイルのサイズ、システム上で実行されるほかのアプリケーションの数、および物理メモリの量によって変わります。

サーバー エンジン用の設定

この初期設定には物理メモリの 20% です。データベース エンジンは、初めて起動されたときにこの値を計算し、レジストリに書き込みます。その後は、エンジンが起動するたびにレジストリから値を読み取ります。エンジンがこの値を再計算することはありません。値を手動で変更するとレジストリが更新されます。

ワークグループ、クライアント キャッシュ、およびクライアント レポート エンジン用の設定

サーバー版以外の Zen のエディションの場合、キャッシュの割当サイズのデフォルト値は 64 MB に設定されます。サーバーの場合と同様、値を手動で変更するとレジストリが更新されます。

メモ PSQL v10 より前の Zen クライアントを使用する場合、キャッシュ割当サイズの値はバイト単位で指定する必要があります。最小の 64 KB（65,536 バイト）が指定されます。

キャッシュ割当サイズを使用したパフォーマンスの最適化

パフォーマンスを最適化するには、キャッシュ割当サイズにより大きな値を設定します。この値は、エンジンが使用しているファイルの合計サイズ以下にしてください。それより大きい値を設定してもメリットはありません。また、ほかのタスクのためにシステムが必要とするメモリを無駄に使用することになります。利用可能なメモリすべて取ってしまうと、特にシステムがほかのアプリケーションを実行している場合は、望ましくない動作を引き起こす可能性があります。

システムへメモリを追加したり、システムからメモリを取り除いたりする場合は、新たなメモリ容量を考慮してこのプロパティを設定し直す必要があります。

通信スレッド数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	**プロセッサのコア数** ～ 256	**プロセッサのコア数**。この**プロセッサのコア数**は、データベース エンジンが起動しているマシンのプロセッサ数です。	なし	Yes

この設定は、リモート クライアントからのリクエストを処理するために MicroKernel が最初に作成するスレッドの数を指定します。通信スレッドは、リクエストを行うクライアント プロセスに代わって、実際に Btrieve オペレーションを実行する要素です。このように、通信スレッドはワーカ スレッドによく似ています。通信スレッドは許容される最大範囲まで、必要に応じて動的に増加します。最大値は、256 です。

通信スレッドの設定は、ある特定の状況下における改善に役立ちます。たとえば、1 つのファイルに対して多くのクライアントが操作（主に書き込み）を行うような場合、より低い値を設定すればスケーラビリティが向上するはずです。スレッド数の値をより低くすると、システム リソースでコンテキスト スイッチを行わないようにします。このほか、大量のワーカー スレッド間でスラッシングによって速度低下が発生する状況も、この通信スレッドの設定によって改善する可能性があります。ワーカー スレッドは、既存の全スレッドがレコードまたはファイルのロックで待機している場合にのみ動的に作成されます。

ファイルを閉じるまでの待ち時間

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0-5000	50	ミリ秒	No

この設定では、ファイルのクライアント接続を閉じた後に、エンジンがそのファイルを開いたままにしておく時間を制御できます。ファイルを繰り返し開いているとエンジンのパフォーマンスが低下するので、ファイルを開いたままにしておく時間を増やすことでパフォーマンスを向上させます。この遅延は 5 秒まで指定できます。値が 0 の場合、この設定はオフになり、最後のユーザーが Btrieve Close オペレーションを実行した直後にファイルが閉じます。

butil -close を使用すると、エンジンは指定された（開いている）ファイルを確認し、遅延時間がまだ残っていても直ちにファイルを閉じます。詳細については、「Close」を参照してください。

ファイルの拡張係数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0-100	15	パーセント	Yes

この値は、バージョン 8.x 以降の形式のデータ ファイル内に保持する空きページの概算パーセントを指定します。この設定は、旧バージョンの形式のファイルには適用されません。MicroKernel では、この値を使用して、ファイルを拡張するか空きページを最初に使用するかを決定します。データベース エンジンは複数の連続したファイル ページをディスク上に書き込むことができます。ファイル内の空きページ数が増加するにつれて、ディスク書き込みのパフォーマンスを向上させることが狙いです。ただし、ディスク書き込みのパフォーマンスはファイルのサイズとの引き換え（トレードオフ）となるので、ファイル内の空きページ数を多くし過ぎると全体的なパフォーマンスは低下する恐れがあります。

データ ファイル内に一定の連続した空きページを保持するためには、データベース エンジンが定期的にそのデータ ファイルを拡張する必要があります。この設定がファイルのサイズに影響があることを覚えておいてください。たとえば、空きページのないファイルで作業するときに、［ファイルの拡張係数］の値に 50% を設定した場合、そのファイルのサイズは最終的に 2 倍になります。［ファイルの拡張係数］の値に 75% を設定した場合、そのファイルのサイズは 4 倍になります。90% を指定した場合、そのサイズは 10 倍近くになります。

完全に使用しないページのみが空きページとみなされることに注意してください。ほとんど使用しないページでも空きページとしてはみなされないので、15% の空きページといった場合、ファイルの 15% が使用されないという意味ではありません。

ファイルが更新される度合いによって、空きページの実際のパーセンテージはいつでも、はるかに小さくなる可能性があります。

この設定は、バージョン 8.x より前の形式のファイルには適用できません。旧バージョンの形式のファイルにも空きページがありますが、そのファイルの動作状況によって空きページのパーセンテージは異なります。

インデックス バランスの実行

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	Yes

この設定は、MicroKernel が、インデックス バランスを実行するかどうかを制御します。インデックス バランスにより、読み取りオペレーションのパフォーマンスは向上します。しかし、このオプションを有効にすると、時間がさらにかかり、挿入、更新、削除オペレーション時にさらにディスクの I/O が必要な場合があります。インデックス バランスの詳細については、『*Zen Programmer's Guide*』を参照してください。

セグメント サイズを 2 GB に制限

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	Yes

この設定は、データ ファイルのサイズがオペレーティング システムのファイル セグメントの上限である 2GB を超えるごとに、ファイルを自動的に分割するかどうかを指定します。**オン**に設定した場合、データ ファイルは 2GB のセグメントに分割されます。**オフ**に設定した場合、データ ファイルはセグメント化されず単一のファイルのままです。より大きいサイズの非セグメント化ファイルを使用する利点は、より効率的なディスク I/O です。つまり、パフォーマンスの向上が期待できます。

セグメント化されていないファイルは、お使いのオペレーティング システムによって指定されているファイル サイズの制限を受けます。以前作成したファイルが既にセグメント化されている場合、そのセグメントはファイル上でそのまま残ります。

「ファイル バージョンの自動アップグレード」の関連情報も参照してください。

メモ この設定は、13.0 形式のファイルには影響しません。この形式のファイルはセグメント化されません。

トランザクション ログ バッファー サイズ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	262144 (0.25 MB) ー メモリによって制限されるサイズ	次の値のうち、小さい方の値： • MB 単位の物理メモリ / 256 • 33549312（32 MB）	バイト	Yes

この設定は、MicroKernel によって使用されるトランザクション ログ バッファーおよびアーカイブ ログ バッファーのサイズを指定します。ログ バッファー サイズを増やすと、MicroKernel がログ情報をディスクに書き込む頻度が低下するため、パフォーマンスが向上します。

メモ ［**トランザクション ログ バッファー サイズ**］に［トランザクション ログ サイズ］より大きい値を設定すると、MicroKernel は［**トランザクション ログ サイズ**］の値を［**トランザクション ログ バッファー サイズ**］に指定した値に自動的に変更します。

MicroKernel の最大メモリ使用量

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0-100	60	パーセント	No

このプロパティは、第 2（L2）キャッシュのサイズを動的に制御することにより、すべてのデータ キャッシュ（L1 + L2）と MicroKernel エンジンが必要とする内部メモリ使用量が、総物理メモリの指定された割合を超えないようにします。エンジンは、指定した割合が必要でないか使用できない場合は、これより少ないメモリを使用します。トランザクションで大量のデータの挿入、更新、および削除操作を行うような、特定のシナリオを処理する必要がある場合には、この設定で指定された制限を超えて内部処理用の追加メモリを使用することができます。このディスカッションにリレーショナル エンジン用のメモリは含まれないことに留意してください。

値にゼロを入力すると、動的キャッシュはオフになります。この場合、使用できるキャッシュは L1 のみで、［キャッシュ割当サイズ］で指定したサイズになります。また、この場合、使用されるメモリの最大量は制限されず、必要に応じて、MicroKernel 用の追加メモリの割り当てと解放が行われるようになります。

データベース サーバー専用のマシンを使用している場合、［MicroKernel の最大メモリ使用量］を最大値に設定するか、オペレーティング システムが占有しないメモリの割合を指定します。データベース サーバーでほかのアプリケーションを実行し、これらすべてのパフォーマンスのバランスをとる必要がある場合は、低い値を設定して、データベース キャッシュが利用可能なメモリを使用するときに、ほかのアプリケーションと競合しないようにする必要があります。

メモ ［キャッシュ割当サイズ］に［MicroKernel の最大メモリ使用量］の値よりも大きな物理メモリ量を設定した場合は、［キャッシュ割当サイズ］の値が優先されます。たとえば、物理メモリが 1 GB のマシンで、［キャッシュ割当サイズ］に 600 MB、［MicroKernel の最大メモリ使用量］に 40% を設定したとします。L1 キャッシュには 600 MB のメモリが割り当てられます。［キャッシュ割当サイズ］の値の方が大きいので、その値が優先され、L1 キャッシュに割り当てられるメモリ量は［MicroKernel の最大メモリ使用量］に指定された値よりも大きくなります。

次の方程式を使用して MicroKernel **最大メモリ使用量**の近似値を割り出します。

(**L1 キャッシュサイズ** + **内部割り当て** + **L2 キャッシュ サイズ** / **物理メモリのサイズ**) * 100

各項目の説明は次のとおりです。

L1 キャッシュ サイズ：「キャッシュ割当サイズ」

内部割り当て：L1 キャッシュの約 25% のサイズ

L2 キャッシュ サイズ：システムのメモリ ロードに基づいて拡大および縮小するメモリ容量

物理メモリサイズ：マシンにインストールされているメモリ容量

パフォーマンス チューニングの詳細については、「パフォーマンス チューニング」を参照してください。

I/O スレッド数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	1 ～ 1024	32	なし	Yes

この設定は、MicroKernel が起動するバックグラウンドの I/O スレッドの数を指定します。これらのスレッドは、MicroKernel のキャッシュからディスク上のファイルへすべてのページをアトミックかつ調和のとれた方法で書き出す責任を果たします。また、最初にファイルを開いてファイル コントロール レコードの読み取りも行います。その他のほとんどの読み取りはローカル ワーカ スレッドおよび通信スレッドが行います。MicroKernel が、データ ファイルを更新、またはデータ ファイルに書き込みを行うと、各ファイルは、指定された I/O スレッドにシーケンシャルに割り当てられます。最後のスレッドに達すると、すべてのデータ ファイルがバックグラウンド スレッドに割り当てられるまで、MicroKernel は処理をやり直します。MicroKernel では、必要に応じて追加の I/O スレッドを作成しないため、必要と思われる I/O スレッドの数を予想して最大数を指定します。

最適なパフォーマンスを実現するには、[最大オープン ファイル数] に基づいた値を指定します。Monitor ユーティリティは開いているファイル数の現在値とピーク値を表示します。データベースに平均 256 個のファイルがある場合、デフォルトの 32 I/O スレッドでは各スレッドが 8 ファイルを受け持つことになります。実際に即した方法として、I/O スレッドごとに、およそ 8 ファイルを持つようにします。たとえば、オープン ファイル数の平均が 400 の場合、およそ 50 個の I/O スレッドを使用します。64 より大きい値を指定するとパフォーマンスが損なわれることがありますが、これはシステムの能力によります。

メモ この設定は、マシンの特性、OS の設定およびデータベース エンジンの作業負荷に影響されるため、適切な I/O スレッド数を正確に計算することはできません。

トランザクション ログ サイズ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	65536 - ディスク容量によって制限される	「トランザクション ログ バッファー サイズ」の 2 倍	バイト	Yes

この設定は、トランザクション ログ セグメントの最大サイズを指定します。ログ ファイルが制限されたサイズに達すると、MicroKernel は古いログ セグメント ファイルを閉じ、新しいファイルの使用を開始します。トランザクション ログ セグメントのサイズを制限することにより、MicroKernel が一時的に使用するディスク容量を少なくできるため、これを行いたいと思われるでしょう。しかし、サイズを制限すると、ログ セグメントを頻繁に閉じて作成することが必要になるため、MicroKernel の処理量が増え、パフォーマンスが低下する可能性があります。

メモ このオプションに [**トランザクション ログ バッファー サイズ**] で指定した値よりも小さい値を設定すると、データベース エンジンは [**トランザクション ログ サイズ**] の設定を自動的に [**トランザクション ログ バッファー サイズ**] の値に合わせます。

Windows クライアント設定プロパティ

Zen のすべてのエディションをクライアントとして構成できます。クライアント プロパティは（クライアントして動作するサーバーも含め）クライアントごとに個別に設定する必要があります。Windows プラットフォームで、Zen Control Center またはコマンド ライン ツールの bcfg を使用してクライアントを構成することができます。ZenCC では、Zen エクスプローラーでデータベース エンジン ノードを右クリックしてプロパティ ウィンドウを開きます。bcfg については、「bcfg を使用した設定」を参照してください。

次の表は、サーバー設定プロパティとその設定項目の一覧をアルファベット順で示します。各設定の詳しい説明がリンク先のセクションにあります。

表 11　クライアント設定プロパティ

設定オプション	プロパティ名
「アクセス」	「ゲートウェイ一貫性保持」
	「ロード再試行回数」
	「IDS の使用」
	「ローカル MicroKernel エンジンの使用」
	「リモート MicroKernel エンジンの使用」
	「ワイヤ暗号化」
	「ワイヤ暗号化レベル」
「アプリケーションの特性」	「スペースを含むファイル / ディレクトリ」
	「キー長の検証」
	「スプラッシュ スクリーン」
「キャッシュ エンジン」	「起動時のリソース割当」
	「非アクティブ時、最小の状態に戻す」
	「キャッシュ割当サイズ」
	「MicroKernel の最大メモリ使用量」
	「最小の状態に戻す待ち時間」
「キャッシュ エンジンのデバッグ」	[キャッシュ エンジンのデバッグ] で使用できる設定は、サーバー構成にある類似した設定と同様の機能をクライアント キャッシュに対して実行します。「デバッグ」の関連するサーバー設定を参照してください。
「通信プロトコル」	「自動再接続の有効化」
	「サポート プロトコル」
	「接続タイムアウト」
「パフォーマンス チューニング」	「キャッシュ エンジンの使用」
「セキュリティ」	「ランタイム サーバー サポート」

アクセス

［アクセス］には次の設定が含まれています。

- 「ゲートウェイ一貫性保持」
- 「ロード再試行回数」
- 「IDS の使用」
- 「ローカル MicroKernel エンジンの使用」
- 「リモート MicroKernel エンジンの使用」

ゲートウェイ一貫性保持

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

このオプションは、MicroKernel ルーターが、Zen が動作していないコンピューターのリストをレジストリに保存する必要があるかどうかを指定します。レジストリに保存すると、ゲートウェイ エンジンの検索に必要な時間を短縮できます。ワークグループに新しいエンジンを追加するときは、このオプションを**オフ**に設定する必要があります。

ロード再試行回数

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	0 ～ 65536	5	なし	適用外

この設定は、MicroKernel ルーターが、ターゲット エンジンに対して接続を再試行する回数を指定します。

IDS の使用

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定は主に、Zen（IDS）を使用するレガシー アプリケーションで使用されます。IDS 機能はコア製品に統合されたため、IDS を単独のインストール済みコンポーネントとして使用することはできなくなりました。IDS の統合は、クライアント / サーバー環境の再構成が必要となります。

一般的には、アプリケーションは独自のファイルの場所情報を指定します。別の方法として、IDS は、テキスト ファイル idshosts の情報に基づいてファイル場所のマッピングを指定します。

アプリケーションが idshosts によるマッピング機能を使用しない場合は、パフォーマンスを向上させるために［**IDS の使用**］を**オフ**にします。

アプリケーションが既に idshosts を使用している場合、あるいはこの代替方法を使用してファイルの場所をマップしたいと考えている場合は、［**IDS の使用**］を**オン**にします。「idshosts ファイルの使用」を参照してください。

メモ ［IDS の使用］を**オン**に設定した場合や、レガシー アプリケーションがファイルの場所情報を PIDS URL の形式で渡す場合には、PSQL 8.5 以降が必要です。リクエスターはデータベース URI を使用して IDS 情報を表します。データベース URI は PSQL 8.5 で追加されました。『*Zen Programmer's Guide*』の「データベース URI」を参照してください。

ローカル MicroKernel エンジンの使用

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	適用外

この設定は、ローカル アプリケーションがローカル エンジンに接続を試みるかどうかを指定します。オフに設定すると、ローカル エンジンへの接続は試行されません。

リモート MicroKernel エンジンの使用

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	適用外

この設定は、MicroKernel ルーターが、リモート サーバーで動作しているエンジンへのアクセスを許可するかどうかを指定します。この設定を**オン**にし、［ローカル MicroKernel エンジンの使用］の設定を**オン**にした場合、まずリモート サーバーへの接続が試行されます。

ワイヤ暗号化

「ワイヤ暗号化」を参照してください。

メモ クライアント側のワイヤ暗号化設定は Zen JDBC および ADO.NET アクセス方法では使用されません。クライアント側のワイヤ暗号化設定の場合、暗号化は接続文字列を使用して指定できます。『*JDBC Driver Guide*』の「接続文字列の概要」、および『*Data Providers for ADO.NET Guide*』の「基本的な接続文字列の定義」を参照してください。

ワイヤ暗号化レベル

「ワイヤ暗号化レベル」を参照してください。

メモ クライアント側のワイヤ暗号化設定は Zen JDBC および ADO.NET アクセス方法では使用されません。クライアント側のワイヤ暗号化設定の場合、暗号化は接続文字列を使用して指定できます。『*JDBC Driver Guide*』の「接続文字列の概要」、および『*Data Providers for ADO.NET Guide*』の「基本的な接続文字列の定義」を参照してください。

アプリケーションの特性

［アプリケーションの特性］には次の設定が含まれています。

スペースを含むファイル / ディレクトリ

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	適用外

このオプションは、MicroKernel エンジン オペレーションのファイル名でスペースを使用できるように MicroKernel エンジンに指示します。

スプラッシュ スクリーン

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定は、MicroKernel エンジンのスプラッシュ スクリーンを表示させるかどうかを制御します。スプラッシュ スクリーンは、最初にクライアント リクエスターがロードされるときに表示されます。

キー長の検証

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オン	なし	適用外

このオプションを従来の Btrieve アプリケーションで使用すると、クライアント リクエスターに渡されたインデックスのキー長がキーに対して十分なサイズであるかどうかが、リクエスターで検証されなくなります。このオプションをオフに設定することで、アプリケーションはステータス 21 のエラーを回避できるようになります。

注意 ただし、オフに設定した場合、このオプションはメモリの上書きを防ぐための Zen リクエスターによるチェックを無効にします。メモリの上書きにより生じる状況で、望ましくない状況はいくつかありますが、中でも一般保護エラー（GPF）が発生する可能性があります。

キャッシュ エンジン

これらの設定は、キャッシュ エンジンが実行されている場合にのみ適用されます。ワークグループ エンジンはキャッシュ エンジンを兼ねています。ただし、データベース サーバー エンジンが実行されている場合にはキャッシュ エンジンは**使用されない**ことに注意してください。

［キャッシュ エンジン］には次の設定が含まれています。

- 「起動時のリソース割当」
- 「非アクティブ時、最小の状態に戻す」
- 「キャッシュ割当サイズ」
- 「MicroKernel の最大メモリ使用量」
- 「最小の状態に戻す待ち時間」

起動時のリソース割当

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定は、キャッシュ エンジンの起動時に、スレッドやメモリ バッファーを含むリソースを割り当てるようにキャッシュ エンジンに指示します。

このオプションをオフにすると、最初のオペレーションが要求されるまでリソースが割り当てられません。必要に応じて、Zen コンポーネントにより、自動的にリソースが割り当てられます。したがって、多くの場合、このリソース割り当てを自分で行う必要はありません。

非アクティブ時、最小の状態に戻す

この設定は、ワークグループ エンジンが実行されている場合にのみ表示されます。

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定により、アクティブなクライアントが存在しない状態で一定の時間が過ぎると、キャッシュ エンジンは大部分のメモリ容量およびスレッド リソースをシステムに解放し、最小の状態に戻ります。その時間は、［**最小の状態に戻す待ち時間**］の値で設定します。ほかのクライアントがアクティブになった場合は、キャッシュ エンジンにより、リソースが再度割り当てられます。

キャッシュ割当サイズ

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Numeric	1 MB からメモリによって制限される量	最初の起動時に動的に初期化される	MB	適用外

この設定は、MicroKernel によって割り当てられるレベル 1 キャッシュのサイズを指定します。MicroKernel では、すべてのデータ ファイルへのアクセスにこのキャッシュが使用されます。

一般的に言うと、［キャッシュ割当サイズ］の値がシステム上の物理メモリの 40% より小さく、設定プロパティの［MicroKernel の最大メモリ使用量］が 40% より大きい値に設定されている場合に、全体のパフォーマンスは最高になります。最適な設定は、データ ファイルのサイズ、システム上で実行されるほかのアプリケーションの数、および物理メモリの量によって変わります。

データベース エンジンは、初めて起動されたときにこの値を設定し、レジストリに書き込みます。この初期値は物理メモリの 20% です。その後は、エンジンが起動するたびにレジストリから値を読み取ります。このプロパティの値を変更すると、レジストリの値も更新されます。システムへメモリを追加したり、システムからメモリを取り除いたりする場合は、新たなメモリ容量を最大限活用するようにこのプロパティを設定し直す必要があります。

メモ PSQL v10 より前の Zen クライアントを使用する場合、キャッシュ割当サイズの値はバイト単位で指定する必要があります。最小の 64 KB（65,536 バイト）が指定されます。

MicroKernel の最大メモリ使用量

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Numeric	0-100	60	パーセント	適用外

この設定は、総物理メモリに対してキャッシュ エンジンが消費できるメモリの割合を指定します。L1、L2 およびその他すべてのメモリに適用される割合がキャッシュ エンジンによって使用されます。データベース エンジンは、メモリが必要でないか使用できない場合は、これより少ないメモリを使用します。

値にゼロ（0）を入力すると、動的キャッシュはオフになります。この場合、使用できるキャッシュは L1 のみで、［**キャッシュ割当サイズ**］で設定されたメモリ量が利用可能です。

パフォーマンス チューニングの詳細については、「パフォーマンス チューニング」を参照してください。

最小の状態に戻す待ち時間

この設定は、ワークグループ エンジンが実行されている場合にのみ表示されます。

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Numeric	0 - 2147483647	300000	ミリ秒	適用外

この設定は、キャッシュ エンジンが最小の状態に戻る前に非アクティブ状態でどれだけ待機するかを指定します。（最小の状態とは、キャッシュ エンジン起動時の初期状態です。）キャッシュ エンジンを最小の状態に戻すと、大部分の容量のメモリとスレッド リソースがシステムに解放されます。キャッシュ エンジンを最小の状態に戻したくない場合もあります。たとえば、繰り返しキャッシュ エンジンを使用するバッチ ファイルを使用中の場合などです。ほかのクライアントがアクティブになった場合は、キャッシュ エンジンにより、リソースが再度割り当てられます。

この設定は、［**非アクティブ時、最小の状態に戻す**］に**オフ**（デフォルト）が設定されている場合は無視されます。

キャッシュ エンジンのデバッグ

これらの設定は、キャッシュ エンジンが実行されている場合にのみ適用されます。ワークグループ エンジンはキャッシュ エンジンを兼ねています。ただし、データベース エンジンが実行されている場合にはキャッシュ エンジンは**使用されない**ことに注意してください。

［**キャッシュ エンジンのデバッグ**］で使用できる設定は、［**サーバー | デバッグ**］にある類似した設定がメインのデータベース エンジンに対して実行するのと同様の機能を、クライアント キャッシュに対して実行します。各設定の詳細については、関連するサーバー設定を参照してください。

- 「データ バッファーのバイト数」
- 「キー バッファーのバイト数」
- 「トレースするオペレーションの選択」
- 「トレース ファイルのロケーション」
- 「トレース オペレーションの実行」

通信プロトコル

［通信プロトコル］には次の設定が含まれています。

- 「自動再接続の有効化」
- 「サポート プロトコル」
- 「接続タイムアウト」

自動再接続の有効化

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定は、ネットワークの停止時にクライアントが自動再接続を試行するかどうかを指定します。**オン**に設定すると、自動再接続が有効になります。

自動再接続は、この設定をサーバーの設定でも有効にしておかない限り、有効になりません。

サポート プロトコル

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
複数選択	1 つ以上	TCP/IP	なし	Yes

この設定では、データベース エンジンがクライアント接続の受信待ちをするプロトコルを指定します。2 つ以上のプロトコルを選択した場合、エンジンはそれらすべてで受信待ちしながらセッションを開始します。最初に接続したプロトコルが、そのセッションの間使用されます。サーバーとクライアントの両方で有効なプロトコルが最低 1 つないと、通信を行うことができません。

メモ デフォルトは TCP/IP です。現在、このオプションのみがサポートされています。

Linux、macOS、および Raspbian

Linux、macOS、および Raspbian 上の Zen でサポートされるプロトコルは TCP/IP のみです。したがって、サポート プロトコルの設定は、これらの環境では使用できません。

接続タイムアウト

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Numeric	1 ～ 2147483647	15	秒	適用外

この設定の値は、クライアントがリモート データベース エンジンを検索または接続するまで待つ時間を指定します。この値の設定が小さ過ぎると、接続が完了する前にタイムアウトになるため、サーバーが見つからないという擬似エラーがクライアントに返されます。この値の設定が大き過ぎると、クライアントが接続不可能なサーバーへ接続しようとした場合、エラーが返されるまでに著しく時間がかかる可能性があります。一般的に、ほとんどのネットワークでは 15 秒から 30 秒の間の値が妥当です。「サーバーが見つかりません」というエラーが頻発する場合は、より高い値を設定してください。

この設定は、以前は**通信リクエスターの TCP/IP 接続タイムアウト**と呼ばれていました。

パフォーマンス チューニング

［パフォーマンス チューニング］には次の設定プロパティが含まれています。

- 「キャッシュ エンジンの使用」

キャッシュ エンジンの使用

データ型	範囲	デフォルト	単位	データベース エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

このプロパティではクライアント キャッシュ エンジンを使用するかどうかを設定します。このエンジンは、MicroKernel エンジンのサブセットで、読み取り専用にデータをキャッシュし、別プロセスで実行されます。クライアント キャッシュ エンジンは、**クライアント キャッシュ**や**キャッシュ エンジン**とも呼ばれます。

［**キャッシュ エンジンの使用**］設定がオフの場合、クライアント側に何もキャッシュされません。アプリケーションからの読み取り要求によって、リモート データベース エンジンからデータを直接取得します。

［**キャッシュ エンジンの使用**］設定がオンの場合、クライアント キャッシュ エンジンが、クライアントとリモート データベース エンジンとの間で、データのキャッシュを仲介する役割を果たすことを意味します。アプリケーションが初めて読み取り要求を行ったときに、クライアント キャッシュ エンジンはデータをキャッシュします。その後、同じレコードに対してさらに読み取り要求を行う際は、このキャッシュ データで対応します。その読み取りのために、リモート データベース エンジンにアクセスする必要はありません。

クライアント キャッシュは、ワークグループ エンジンに似ています。デフォルトでは、アプリケーションが最初にデータベースにアクセスしたときにクライアント キャッシュはメモリに自動的にロードされ、そのアプリケーションを終了すると直ちにアンロードされます。

クライアント キャッシュをメモリに保持して、各使用セッションがキャッシュを再配置するパフォーマンス コストを回避したい場合もあるでしょう。クライアント キャッシュをロードしたままにするには、コマンド プロンプトから次を実行してください。

install_path¥Zen¥bin¥zenengnapp

クライアント キャッシュが開始すると、タスク バーの右端の通知領域にアイコンが表示されます。このアイコンでクライアント キャッシュ エンジンを制御できます。クライアント キャッシュを停止するには、アイコンを右クリックし、［**エンジンを停止して終了**］を選択します。

クライアントがサービスとしてインストールされた場合、クライアント キャッシュ エンジンはデフォルトで自動的に開始します。ただし、クライアント キャッシュ サービスが実行中でも、［**キャッシュ エンジンの使用**］がオンになっていなければ、アプリケーションはクライアント キャッシュを実行しません。

メモ Zen は、このクライアント キャッシュと、データベース エンジン キャッシュやほかのクライアント キャッシュとの同期をとります。この動作は完全に自動的かつ透過的に行われます。ただし、データベース エンジン キャッシュでデータベースの変更が発生し、その変更がすべてのクライアント キャッシュに反映されるまでの間には最大 5 秒の遅延が生じます。お使いのシステムで、クライアント キャッシュにそのような古いページが最大 5 秒間存在することが許容されない場合は、クライアント キャッシュを使用しないでください。

以下の操作はクライアント キャッシュには**格納されません**。

- トランザクション内にあるすべての操作
- ロック バイアスのかかった操作
- INSERT、UPDATE、DELETE などの書き込み操作
- Open および Close 操作

「キャッシュ割当サイズ」プロパティのクライアント キャッシュに関する説明も参照してください。

セキュリティ

［セキュリティ］には次の設定が含まれています。

- 「ランタイム サーバー サポート」

ランタイム サーバー サポート

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
String	Yes No ユーザー名 , パスワード	Yes	なし	適用外

この設定は、ランタイム サーバー サポートを制御します。この設定が Yes（有効）の場合は、現在実行しているドライブの現在のユーザー名とパスワードが使用されます。別のユーザー名で RTSS を有効にするには、「**ユーザー名 , パスワード**」を入力します。

SUPERVISOR および ADMIN は、正しいパスワードと一緒に入力しても、有効なユーザー名ではありません。リクエスターが、SUPERVISOR または ADMIN 以外のユーザー名を検出できない場合は、ログインを試行しません。

Linux、macOS、および Raspbian クライアント設定プロパティ

Zen Control Center またはコマンド ライン ツールの bcfg を使用して Zen クライアントを構成することができます。ZenCC では、Zen Control Center で MicroKernel ルーター ノードを右クリックしてプロパティ ウィンドウを開きます。bcfg については、「bcfg を使用した設定」を参照してください。

メモ ZenCC の GUI がサポートされない Raspbian の場合は、別の Zen インストレーションで ZenCC を開き、リモートでクライアントを追加することでその設定プロパティにアクセスできます。PowerShell などのリモート セッションで bcfg を使用することもできます。

設定値での大文字小文字の区別

Linux、macOS、または Raspbian クライアントは設定値の確認または変更を行う場合、大文字小文字を区別しません。たとえば、設定値に `Yes` または `yes` を入力した場合、クライアントはこれらを同一の値であると解釈します。

ローカル設定（[Local]）によって影響を受けるクライアントのパフォーマンス

Linux、macOS、および Raspbian クライアント インターフェイスが初めて呼び出されたときに、そのデフォルトの設定値が Zen レジストリに設定されます。Zen クライアントは、自身のインストールにサーバー エンジンが含まれているかどうかを認識しません。そのため、ローカル設定（[Local]）を有効（"Yes"）にします。これは、クライアントのパフォーマンスに影響を及ぼす可能性があります。

クライアントを使用しているコンピューターにサーバー エンジンがない場合は、ローカル設定（[Local]）を無効 ("No") にします。「Use Local MicroKernel Engine ｜ ローカル MicroKernel エンジンの使用」を参照してください。

埋め込みスペースを含むファイル名

デフォルトで、Linux、macOS、および Raspbian クライアント インターフェイスはスペースを含むファイル名をサポートします。

たとえば、次のような例です。

```
/mymount/usr/gary/file with spaces.mkd
```

スペースを含まないファイル名を使用したい場合は、[スペースを含むファイル / ディレクトリ]（[Embedded Spaces]）設定を変更する必要があります。「Embedded Spaces ｜ スペースを含むファイル / ディレクトリ」を参照してください。

設定リファレンス

次の表は、Linux、macOS、および Raspbian クライアントの設定オプションの一覧をアルファベット順で示します。各設定の詳しい説明がリンク先のセクションにあります。

表 12　クライアントの設定

設定オプション	プロパティ名
「Access ｜アクセス」	「Use Local MicroKernel Engine ｜ローカル MicroKernel エンジンの使用」
	「Use Remote MicroKernel Engine ｜リモート MicroKernel エンジンの使用」
	「Use IDS ｜ IDS の使用」
	「Wire Encryption ｜ワイヤ暗号化」
	「Wire Encryption Level ｜ワイヤ暗号化レベル」
「Communication Protocols ｜通信プロトコル」	「Enable AutoReconnect（自動再接続の有効化）」
「Application Characteristics ｜アプリケーションの特性」	「Embedded Spaces ｜スペースを含むファイル / ディレクトリ」
	「Verify Key Length ｜キー長の検証」

Access ｜アクセス

［Access ｜アクセス］には次の設定が含まれています。

- 「Use Local MicroKernel Engine ｜ローカル MicroKernel エンジンの使用」
- 「Use Remote MicroKernel Engine ｜リモート MicroKernel エンジンの使用」
- 「Use IDS ｜ IDS の使用」
- 「Wire Encryption ｜ワイヤ暗号化」
- 「Wire Encryption Level ｜ワイヤ暗号化レベル」

Use Local MicroKernel Engine ｜ローカル MicroKernel エンジンの使用

「ローカル MicroKernel エンジンの使用」を参照してください。

Use Remote MicroKernel Engine ｜リモート MicroKernel エンジンの使用

「リモート MicroKernel エンジンの使用」を参照してください。

リモート エンジンと UNC パス

UNC パスがクライアントから正しく動作するようにするには、以下のことを行う必要があります。

- アクセスしようとするファイルと同じコンピューターにあるエンジンを実行している必要があります。
- ［**リモート MicroKernel エンジンの使用**］をオンに設定しておきます。

メモ ローカルの Linux、macOS、または Raspbian システムを指す UNC パスを使って送信することはできません。ただし、次のような UNC スタイルのパスは使用することができます。

```
//localhost/usr/local/actianzen/data/samples/sample.btr
```

ファイル サーバーにエンジンが不要の場合（つまり、クライアントのローカル エンジンが欲しい場合）、クライアントのリモート ファイル システムをマウントし、UNC 形式ではなく " ネイティブ形式 " のパスになるようパスを変更することができます。たとえば、次のようなパスが Linux、macOS、および Raspbian でネイティブな形式です。

```
/mnt/myremotedata/sample.btr
```

Use IDS ｜ IDS の使用

「IDS の使用」を参照してください。

Wire Encryption ｜ワイヤ暗号化

「ワイヤ暗号化」を参照してください。

メモ クライアント側のワイヤ暗号化設定は Zen JDBC および ADO.NET アクセス方法では使用されません。クライアント側のワイヤ暗号化設定の場合、暗号化は接続文字列を使用して指定できます。『*JDBC Driver Guide*』の「接続文字列の概要」、および『*Data Provider for .NET Guide*』の「基本的な接続文字列の定義」を参照してください。

Wire Encryption Level ｜ワイヤ暗号化レベル

「ワイヤ暗号化レベル」を参照してください。

メモ クライアント側のワイヤ暗号化設定は Zen JDBC および ADO.NET アクセス方法では使用されません。クライアント側のワイヤ暗号化設定の場合、暗号化は接続文字列を使用して指定できます。『*JDBC Driver Guide*』の「接続文字列の概要」、および『*Data Provider for .NET Guide*』の「基本的な接続文字列の定義」を参照してください。

Communication Protocols ｜通信プロトコル

［Communication protocols ｜通信プロトコル］には次の設定が含まれています。

- 「Enable AutoReconnect（自動再接続の有効化）」

Enable AutoReconnect（自動再接続の有効化）

データ型	範囲	デフォルト	単位	エンジン再起動の必要性
Boolean	オン オフ	オフ	なし	適用外

この設定は、ネットワークの停止時にクライアントが自動再接続を試行するかどうかを指定します。**オン**に設定すると、自動再接続が有効になります。

自動再接続は、この設定をサーバーの設定でも有効にしておかない限り、有効になりません。

メモ Zen Linux、macOS、および Raspbian のクライアントはこの機能をサポートしますが、現在、それらのオペレーティング システム上のサーバーではサポートしません。AutoReconnect（自動再接続）を使用できるのは、それらのクライアントから Windows サーバーに接続する場合のみです。

Application Characteristics ｜アプリケーションの特性

［Application Characteristics ｜アプリケーションの特性］には次の設定が含まれています。

- 「Embedded Spaces ｜スペースを含むファイル / ディレクトリ」
- 「Verify Key Length ｜キー長の検証」

Embedded Spaces ｜スペースを含むファイル / ディレクトリ

「スペースを含むファイル / ディレクトリ」を参照してください。

Verify Key Length ｜キー長の検証

「キー長の検証」を参照してください。

Reporting Engine 設定プロパティ

Zen Control Center またはコマンド ライン ツールの bcfg を使用して Client Reporting Engine を構成することができます。

- ZenCC では、Zen エクスプローラーで Reporting Engine ノードを右クリックしてプロパティ ウィンドウを開きます。
- bcfg については、「bcfg を使用した設定」を参照してください。

パフォーマンス

5

データベース パフォーマンスの分析およびチューニング

以下のセクションでは、データベースのパフォーマンスについて説明します。

- 「パフォーマンスの分析」
- 「パフォーマンス チューニング」
- 「ハイパーバイザー製品でのパフォーマンス」

パフォーマンスの分析

Zen サーバーの Windows 版では、Windows パフォーマンス モニターで使用するパフォーマンス カウンターを提供します。パフォーマンス カウンターはデータベース エンジンの状態や動作を測定します。これによりアプリケーションのパフォーマンスを分析することができます。

「パフォーマンス カウンターの監視」を参照してください。

パフォーマンス チューニング

このセクションでは、データベース初期接続および実行時オペレーションでパフォーマンスを最大にするための一般的なヒントをいくつか提供します。一般的なガイドラインをいくつか示しますが、特定のアプリケーションのパフォーマンスは、以下に挙げる要因だけに限らず、非常に多くの要因によって変わります。

- ネットワークの帯域幅および使用率
- アプリケーションでのコーディング技法
- 使用可能な物理記憶域
- 使用可能なメモリ
- プロセッサの速度
- アプリケーションでの使用パターン（大量の書き込み、大量の読み取り、トランザクションの使用 / 不使用、レコード サイズの大小、複雑または単純なクエリ、ODBC、Btrieve のみ、など）
- CPU サイクルで競合する無関係なアプリケーション
- データベース エンジン設定

ご覧のように、エンジンの設定は一定のアプリケーションの全体のパフォーマンスの中では比較的限定的な役割を果たします。さらに、データベース エンジンは使用パターンに基づいてさまざまなリソースを動的に管理します。データベース エンジン自体が、必要に応じ環境に合わせてチューニングします。以下のセクションで提供するのは、有用なガイドラインにすぎず、特定のレベルのパフォーマンスを保証するものではありません。

パフォーマンスのボトルネックを見極める

Monitor ツールを使用すると、特定のデータベース エンジンの設定オプションに関連するパフォーマンスのボトルネックを見つけることができます。オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から、あるいは Zen Control Center の［**ツール**］メニューから Monitor を起動します。

Monitor の表示と設定オプション

Monitor にある 2 つのタブには、設定オプションに関連するパフォーマンスの値が表示されます。

- リソース使用状況
- MicroKernel 通信統計情報

次の表に示すように、データベース エンジンは複数のサーバー設定オプションを動的に管理します。

表 13　Monitor で表示される、動的に管理されている設定

動的に管理される設定	リソース使用状況タブに表示される値	通信統計情報タブに表示される値
ファイル数	X	
ハンドル数	X	
クライアント数	X	
ワーカ スレッド数	X	
リモート セッション総数		X

表示および動作の解釈

Monitor に表示される情報を活用することができます。Monitor はリソースのタイプごとに 3 つの情報を表示します。たとえば、リモート セッション総数には次のように表示されます。

- **現在値**。実際に動作しているリモート セッションの現在の数。
- **ピーク値**。エンジンが開始されてから実際に使用されたリモート セッションの最大数。
- **最大値**。リモート セッションの使用可能な最大数。

リソースのピーク値と最大値が同じ場合には、そのリソースの最大値を増やすように設定プロパティを設定すれば、データベース エンジンは、必要な場合に特定のリソースの追加のインスタンスを割り当てることができます。

設定プロパティを変更する前に

以下のセクションでは、次のことを前提とします。

1 Zen Control Center（ZenCC）が既に開いていること。

このタスクについての説明が必要な場合は、『*Zen User's Guide*』の「Windows での ZenCC の起動」を参照してください。

2 構成しようとするエンジンが既に登録済みであること（可能な場合）。

このタスクについての説明が必要な場合は、『*Zen User's Guide*』の「リモート サーバー エンジンを登録するには」を参照してください。

3 所定のエンジンを構成するには、オペレーティング システムの適切な権限が必要です。

このタスクについての説明が必要な場合は、『*Zen User's Guide*』の「データベース エンジンの管理者権限の許可」を参照してください。

4 一部のエンジン設定については、設定を変更した後にデータベース エンジンの再起動が必要になります。

初期接続時間を最小限にする

最短接続時間の基礎を成す理論は、3 つの必要条件を中心に展開します。これらの必要条件についての要約は次のようになり、詳細についてはその後で説明します。

- **既知の通信プロトコル。**クライアントまたはサーバーが、その環境でサポートされていないプロトコルを介して接続を試行するために余計な時間を費やすことは望ましくありません。クライアント / サーバーが使用できないプロトコルを削除することにより、ネットワーク コンポーネントがそれらの使用を試行することを防げます。
- **既知のデータベース エンジンのロケーション。**クライアントが存在しないエンジンへの接続を試行しない場合もあります。ワークグループのみのサポート、またはサーバーのみのサポートを適切に指定することにより、クライアントが存在しない接続でタイムアウトになることを防ぐことができます。非共有および共有データベースの両方がある環境では、ゲートウェイの一貫性保持を使用して、データベース エンジンがデータベース エンジンの実行されていないマシンを常に知ることができ、これにより、これらのマシン上のエンジンへの接続を試行せず、別の方法でデータにアクセスします。
- **実行準備の整ったデータベース エンジン。**休止しているエンジンが新しい接続リクエストを受け取ると、リソースの再割り当ておよび実行時の状態に戻るのに時間がかかります。接続スピードがサーバー上のリソース使用量より重要な場合は、サーバー エンジンが使用されていないときにも休止しないようにすることができます。

クライアントのプロパティ

クライアントのプロパティの変更は、そのクライアント マシンで行う必要があります。個々のクライアントでプロパティを変更する必要があります。

▶▶ クライアント側の接続待ち時間を最小限にするには

1 Zen エクスプローラーで、ツリーの**ローカル クライアント** ノードを展開します（ノードの左の展開アイコンをクリックします）。

2 ［MicroKernel ルーター］を右クリックします。

3 ［**プロパティ**］をクリックします。

4 プロパティ ツリーで［**通信プロトコル**］をクリックします。

5 ［**サポート プロトコル**］で、必要なプロトコルにはチェック マークが入って選択されており、**使用されない**プロトコルのチェック マークは**外されている**ようにします。

6 ［**適用**］をクリックします。

これで、クライアントが、使用されていないプロトコルと通信しようとするのを妨ぐことができるようになりました。

7 プロパティ ツリーで［**アクセス**］をクリックします。

8 リモート　サーバーまたはリモート　ワークグループ　エンジンのみを使用している場合は、［**ローカル MicroKernel エンジンの使用**］チェック ボックスの選択を**解除**し、**オフ**に設定されるようにします。

ローカル ワークグループ エンジンのみを使用している場合は、［**リモート MicroKernel エンジンの使用**］チェック ボックスの選択を**解除**し、**オフ**に設定されるようにします。

ワークグループ エンジンを使用することもあり、サーバー エンジンまたはリモート ワークグループ エンジンを使用することもある場合には、両方の設定を選択し、**オン**にしておく必要があります。

このような共有と非共有データの混合環境では、各クライアントで［**ゲートウェイ一貫性保持**］を**オン**に設定しておくことができます。この設定によって、データベース エンジンに接続することができないすべてのマシンの名前のリストを強制的にクライアント ソフトウェアに保持させます。クライアント ソフトウェアが、指定されたコンピューターにはエンジンが存在しないことを判定するには、すべてのネットワーク プロトコルのリクエストがタイムアウトになるまで待ちます。

Zen データベース エンジンを持たないサーバーにデータが格納されていて、［**リモート MicroKernel エンジンの使用**］の設定がオンの場合、クライアントはそのマシンにエンジンがないことを知るために少なくとも 1 度はタイムアウトを待たなければなりません。ゲートウェイ一貫性保持を使用すると、アプリケーションがそのデータに最初にアクセスしたときのみにこのタイムアウトが起きるようにできます。

メモ　ゲートウェイ一貫性保持を使用すると、ネットワーク トポロジを固定します。後に、リモート コンピューターに Zen サーバーまたはワークグループ エンジンをインストールする場合、各クライアントで［ゲートウェイ一貫性保持］をオフにして、データベース エンジンを持たないコンピューターのリストが削除されるようにする必要があります（これにより新しいデータベース エンジンに接続できるようになります）。［ゲートウェイ一貫性保持］に戻ると、リストは空になっています。

9 ［OK］をクリックします。

これで、クライアントが、使用されていないデータベース エンジンに接続しようとするのを防ぐことができます。

サーバーのプロパティ

▶▶ サーバー側の接続待ち時間を最小限にするには

1 エクスプローラーで、ツリーの**エンジン**　ノードを展開します（ノードの左の展開アイコンをクリックします）。

2 設定を指定するデータベース エンジンを右クリックします。

3 ［**プロパティ**］をクリックします。

4 ［**サポート プロトコル**］で、必要なプロトコルにはチェック マークが入って選択されており、**使用されない**プロトコルのチェック マークは**外されている**ようにします。

5 ［**適用**］をクリックします。

これで、サーバーが、使用されていないプロトコルと通信しようとするのを防ぐことができます。

メモ サーバーの設定で選択したプロトコルの少なくとも 1 つが、クライアントの設定で選択されたものと同一であることを確認してください。サーバーとクライアントは同じプロトコルを使用しなければ通信できません。

6 プロパティ ツリーで［**メモリの使用**］をクリックします。

7 ［**起動時のリソース割当**］をクリックして、値を**オン（チェック ボックスを選択）**に設定します。

このオプションは、データベース エンジンが必要なすべてのメモリを割り当てるタイミングを、最初の接続要求が入ってきたときではなく、起動時とします。この値を選択すると、より多くのメモリが必要ですが、クライアントから見ると、エンジンの処理速度が速くなります。

8 ［**非アクティブ時、最小の状態に戻す**］を**オフ**に設定します（チェック ボックスをクリア）。

このオプションは、データベース エンジンが非アクティブなときにも、リソースを解放してオペレーティング システムに戻さないことを指定します。すべてのリソースは割り当てられたままで、次のクライアント接続に備えられます。

9 ［OK］をクリックします。

10 これらの変更を有効にするために、［**はい**］をクリックしてエンジンを再起動します。

ランタイムのスループットを最大にする

最大スループットを得るための原理は、ここに挙げる非常に多くの要因に依存します。最も重要な要因のいくつかを挙げます。

- **十分な物理メモリ。**ホスト システムの RAM が少なすぎる場合、異なるユーザーおよびプロセスが限られたリソースを競い合うと、システムはその時間と CPU サイクルのほとんどをメモリとディスクのスワップに費やすことになります。
- **十分な CPU 能力。**CPU が、データ処理リクエストの流入量について行けなくなると、アプリケーションのパフォーマンスが悪影響を受けます。
- **十分なネットワーク帯域幅。**ネットワークが遅かったり、高いコリジョン率によって悪影響を受けている場合、データベース エンジンはデータ アクセスのリクエスト間で休止状態になり、各クライアントでは低いパフォーマンスを示します。
- **最小限のディスク I/O。**ディスク I/O は、メモリ I/O より著しく速度低下を招きます。十分なキャッシュを確保して、できる限りディスクにアクセスすることを避けます。
- **十分なリソース割り当て。**大きな物理メモリが使用可能であっても、データベース　エンジンの設定プロパティが人為的に低く設定されている場合には、データベースのパフォーマンスは悪くなります。

結局、最適化されたパフォーマンスは、ネットワークのボトルネック、ディスク I/O のボトルネック、メモリのボトルネック、および CPU のボトルネックの間を綱渡りすることで実現しています。このセクションでは、メモリおよびディスク I/O のボトルネックを減少させる方法のガイドラインをいくつか示します。

速いディスクと速い CPU

ハードウェアへの投入費用に対しパフォーマンス向上の効果を最大にしたければ、既存のパフォーマンスの制約を理解する必要があります。データベースが非常に大きくて、データベースの主要な部分をキャッシュに入れるための十分なメモリを購入してインストールするのが妥当でない場合、パフォーマンスはディスク I/O によって制約を受けがちです。これらの状況では、速い RAID ディスク配列に費用をかけてディスク I/O のパフォーマンスを最大にします。

さらに、アプリケーションがリレーショナル エンジンを使用していて一時ファイルを頻繁に作成する場合、これらのファイルが作成されるディレクトリが速いディスク ドライブであることを望むでしょう。このディレクトリ

の場所および一時ファイルを作成するクエリのタイプの詳細については、『*SQL Engine Reference*』の「テンポラリ ファイル」を参照してください。

データベースが小さくてメモリに全部またはほとんどがキャッシュできる場合、速いディスク配列の追加では著しいパフォーマンス向上を得る可能性はありません。このような状況下では、CPU をアップグレードするか CPU を追加することにより最高のパフォーマンス改善値が得られます。

十分な物理メモリとデータベース キャッシュを確保する

Pervasive.SQL バージョン 8 以降を使用する場合、データベース エンジンは設定プロパティで設定した［**キャッシュ割当サイズ**］のレベル 1 キャッシュに加え、レベル 2 動的キャッシュを提供します。［**MicroKernel の最大メモリ使用量**］にゼロを設定してレベル 2 動的キャッシュを無効にしないとすれば、レベル 1 キャッシュ サイズを手動で調整するのは、以前のリリースよりずっと楽です。これを念頭において、このセクションでは、データベース エンジンの最適なパフォーマンスのための十分なメモリを確実に利用できるようにする方法を説明します。

理想的には、データベース エンジンが十分なメモリを割り当てることができ、管理する各データベースの完全なコピーをキャッシュできれば、ディスク I/O を最大限避けることができます。1 つまたは複数のデータベース全体をキャッシュするのは状況によって、特にデータベースサイズが非常に大きい場合には、明らかに実用的ではありません。さらに、既存のシステム リソースが通常の使用でも大きな負荷がかかっている場合、パフォーマンスを向上させる対策はマシンに RAM を追加することだけです。

データベース エンジンは最初に起動されたときに動的にレベル 1 キャッシュ値を選択します。ただし、この値は使用可能なメモリに基づいていて、その環境で理想的なキャッシュ量になるとは限りません。

▶▶ データベースのメモリ キャッシュの理想的なサイズを計算するには

1　まず、データベース エンジンがサービスするすべてのデータベース ファイルのサイズを合計します。

メモ　エンジンがサービスするデータベースが 2 つ以上あっても、同時に使用されない場合は、大きい方のサイズだけを加算します。

たとえば、サーバーに 2 つのデータベースがあり、それぞれに含まれるファイル サイズは次のとおりで、ユーザーが同時に両方のデータベースにアクセスするとします。

データベース A		データベース B	
file1.mkd	223 MB	Afile.mkd	675 MB
file2.mkd	54 MB	Bfile.mkd	54 MB
file3.mkd	92 MB	Cfile.mkd	318 MB
file4.mkd	14 MB		

これらすべてのファイルの合計は 1430 MB です。

今得た値は、データベース エンジンがホストするすべてのデータをキャッシュした場合に使用するメモリの最大量です。この数値を *MaxCache* と呼びます。

キャッシュ割当サイズに、これより大きい値を指定しようとは考えないでしょう。そうすることは、データベース エンジンにまったく使用しないと思われるメモリを割り当てることになるからです。実際の運用に妥当なのは、**キャッシュ割当サイズ**に *MaxCache* の 20% から 70% を設定することです。この範囲中で低い値は読み取り処理の多いアプリケーションに最適で、高い値は書き込み処理の多いアプリケーションに最適です。それは、すべての書き込み / 更新操作はレベル 1 キャッシュで行われるからです。

メモ ファイル ページは、アクセスされたときにのみデータベース キャッシュに書き込まれます。したがって、データベース エンジンはデータベースのすべてのページがアクセスされるたびに *MaxCache* を必要とします。この評価方式により、長期間安定した状態でデータベース処理を行うことができます。データベース エンジンを毎晩または毎週停止する場合、一定の稼働時間内にデータベース エンジンが、データベースのすべてのページにアクセスすることはあまりありません。

このような状況の場合、一定の稼動期間にアプリケーションが別個のページに平均何回アクセスするかを見積もり、ページ サイズを掛けて、特定の稼動期間のより現実的な *MaxCache* の値を求めたいでしょう。

「MicroKernel キャッシュ用のカウンター」も参照してください。

Windows 32 ビット オペレーティング システムでは、すべてのユーザー プロセスのメモリは 2GB に制限されています。*MaxCache* の計算値が 2 GB より大きく、データベース エンジンが 32 ビット Windows オペレーティング システム上で実行される場合、選択する *MaxCache* 値によってエンジンが絶対に 2 GB の限界を超えないようにしていください。そうしないと、エンジンがメモリを割り当てることができず、その後、エラーになる可能性があります。

▶▶ 必要な物理メモリ総量を決定するには

次の方程式を使用して、データベース エンジンで必要とされる総物理メモリ量の近似値を割り出します。

データベースの最大メモリ使用量 = **L1 キャッシュサイズ** + **内部割り当て** + **L2 キャッシュサイズ**

各項目の説明は次のとおりです。

L1 キャッシュ サイズ：「キャッシュ割当サイズ」

内部割り当て：L1 キャッシュの約 25% のサイズ

L2 キャッシュ サイズ：システムのメモリ ロードに基づいて拡大および縮小するメモリ容量

以下の点に注意してください。

- L1 キャッシュは「キャッシュ割当サイズ」に基づく固定サイズです。このサイズはデータベース操作によって拡大したり縮小したりすることはありません。アプリケーションで多数の書き込み操作を行う場合は、この L1 キャッシュをできるだけ増やしてください。
- すべてのデータを L1 キャッシュに収容することができれば、最高のパフォーマンスが得られます。すべてのデータが L1 キャッシュに収容されない場合は、「キャッシュ割当サイズ」および 「MicroKernel の最大メモリ使用量」を調整して妥当なシステム メモリ量を使用します。
- L2 キャッシュは、システムのメモリ ロードに基づいて拡大および縮小します。たとえば、ほかのアプリケーションでより多くのメモリが必要となった場合、L2 キャッシュは縮小します。十分なメモリが使用可能であれば、L2 キャッシュは上記の方程式に基づいて最大量に達します。L2 キャッシュの拡大や縮小はパフォーマンスに影響します。縮小した場合は、データ ページが L2 キャッシュから除去されるなどの事象が起こります。ディスク ストレージからのデータ ページの読み取りは、キャッシュに保持されたデータ ページの読み取りと比べ、より時間がかかります。
- L2 キャッシュには圧縮データ ページが含まれます(少ないスペースにより多くのページが収まります)。L2 キャッシュからのページ アクセスは、L1 キャッシュからのページ アクセスよりも時間がかかりますが、ディスク ストレージからのページ アクセスに比べれば処理が速くなります。多数の読み取り操作を実行するアプリケーションで L2 キャッシュは役立ちますが、すべての読み取りデータ ページを L1 キャッシュに収容することはできません。

ディスク I/O を最小限にする

ディスクからの読み取りおよびディスクへの書き込みは、メモリに対するそれよりずっと時間がかかります。したがって、パフォーマンスを最適化する 1 つの方法はディスクの動作を最小限にすることです。

ディスク I/O を最小化する上での重要な考慮点はデータの回復の可能性です。ディスク I/O は、トランザクション一貫性および回復可能性に対する直接的な交換条件です。変更のログをディスクに書き込むための停止なしで

データをメモリに保持するほど、データベースのパフォーマンスは速くなります。一方、変更のログをディスクに書き込むための停止を行わずにデータをメモリに保持するほど、システムに異常をきたした場合に失うデータが増えます。

▶▶ ディスク I/O を減少させるには

1 前のサブセクション、「十分な物理メモリとデータベース キャッシュを確保する」で説明したように、最も重要な考慮点の 1 つは、ディスクおよびキャッシュ間のデータ ページの頻繁なスワップを避けるために十分なデータベース メモリ キャッシュを確保することです。詳細については、そのセクションを参照してください。

ディスク I/O を減らす最良の方法の 1 つは、動的なレベル 2 キャッシュを必ずオンにすることです。レベル 2 キャッシュはキャッシュ要求がレベル 1 固定キャッシュの容量を超えたとき、アプリケーションの使用量の変化によってサイズを動的に調整して可能な限り多くのデータをメモリに格納し、それによりディスク I/O を減らします。デフォルトで、レベル 2 キャッシュはオンになっています。データベース エンジンがレベル 2 キャッシュを使用していることを確認するには、設定プロパティの［MicroKernel の最大メモリ使用量］を調べます

2 次に、システム異常の場合に、どれだけのロギングが必要で、どれだけのデータベース　オペレーションを失ってもかまわないかを考慮します。失ってもかまわない変更が多いほど、パフォーマンスの向上を追求することができます。

「アーカイブ ログの使用」、「トランザクション一貫性保持」、および「トランザクション ログ」はすべてログ ファイルを必要とします。デフォルトで、アーカイブ ロギングはオフになっています。定期的なバックアップを実行し、システム異常の時点のデータに回復する能力を必要とする場合にのみこれをオンにします。ログの対象とするファイルを指定するときには、完全にログを必要とするファイルのみを指定するようにしてください。詳細については、第 9 章の「ログ、バックアップおよび復元」を参照してください。

デフォルトで、トランザクション ログはオンになっています。トランザクション ログをオフにすると、パフォーマンスは若干向上しますが、複数ファイルの一貫性とトランザクション アトミシティは保証されません。トランザクション ログをオフにする前に、アプリケーションがこの機能を使用しないで実行できるかどうかをベンダーに確認してください。

注意 トランザクション ログが無効な場合、複数ファイルのデータベースの整合性は保証されません。

デフォルトで、トランザクション一貫性保持はオフになっています。アプリケーションでシステム クラッシュが起きてもトランザクション処理が完了することを必要とする場合は、この機能を有効にします。トランザクション一貫性保持を使用すると、パフォーマンスに最も高いペナルティが課されますが、引き換えに完了したトランザクションの最高の安全性を引き出します。

3 いずれかのロギング機能をオンにしている場合、エンジンがディスクに書き込む前にどれだけのデータを保存するかを指定することができます。変更データは繰り返し作成されるので、この機能は重要です。メモリに構築するログ データの許容量が多いほど、ディスク書き込みの頻度は減ります。

［**トランザクション ログ バッファー サイズ**］設定は、エンジンがデータベース操作をログ ファイルに書き出す前に、メモリに格納しておくバイト数を指定します（サーバーのプロパティ ツリーで［**パフォーマンス チューニング**］をクリックします）。

システム異常の場合、ログ バッファー内のデータは失われます。

4 トランザクション一貫性保持をオンにしている場合、ディスク上のログセグメントの最大サイズを指定することができます。小さいログ セグメントは作成と閉じることを繰り返すので、大きなログ セグメント サイズを指定すると、若干パフォーマンスを向上させることができます。

［トランザクション ログ サイズ］設定は、ログ セグメントを閉じて新しいログ セグメントを開くまでにログ セグメントに格納できる最大バイト数を指定します（サーバーのプロパティ ツリーで［**パフォーマンス チューニング**］をクリックします）。

5 データベースの使用率が高い場合には、データ ファイルが存在するのとは物理的に異なるボリュームにログが保持されるようシステムを構成する必要があります。一般的に高負荷の下では、単一ドライブで I/O 帯域幅を競う代わりに、ログ ファイルへの書き込みとデータ ファイルへの書き込みを別々のドライブに分ける方がパフォーマンスがよくなります。全体的なディスク I/O は減少しませんが、ロードはディスク コントローラー間で効率よく分散されます。

6 アプリケーションの使用方法がデータベース読み取り処理に重点を置いている場合、［インデックス バランスの実行］を調整することにより、パフォーマンスを向上させることができます（サーバーのプロパティ ツリーで［**パフォーマンス チューニング**］をクリックします）。インデックス バランスは通常のインデックス ページのノード数を増やし、読み取り操作を速くします。ただし、Insert、Update、および Delete オペレーションでは、エンジンは隣接ページにまたがってインデックス ノードのバランスをとるため、ディスク I/O 時間が余計に必要になることがあります。

7 MicroKernel および ODBC レベルの両方でトレースを必ずオフにしてください。トレースは、多くのディスク I/O を引き起こすため、パフォーマンスを著しく低下させます。

ODBC のトレースがオフであることを確認するには、Zen Control Center から **ODBC アドミニストレーター**を起動します。ODBC アドミニストレーターで［**トレース**］タブをクリックします。トレースがオフの場合、［トレースの開始］と表示されたボタンがあるので、［**キャンセル**］をクリックします。トレースがオンの場合は、［**トレースの停止**］ボタンをクリックし、［OK］をクリックします。

MicroKernel のトレースがオフであることを確認するには、設定プロパティの［トレース オペレーションの実行］が**オフ**に設定されている（チェックされていない）かを調べます（サーバーのプロパティ ツリーで［**デバッグ**］をクリックします）。

十分なリソース割り当てを確保する

データベース サーバー プラットフォームに十分なメモリおよび CPU パワーがある場合は、データベース エンジンが複数クライアントおよび複数データ ファイルを最も効果的にサービスできるよう、ハードウェア リソースを最大限利用できるようにしてください。

▶▶ 複数クライアントおよび複数ファイルを扱えるように設定するには

1 ［I/O スレッド数］設定により、ファイル操作の処理に利用できるスレッド数を指定することができます（サーバーのプロパティ ツリーで［**パフォーマンス チューニング**］をクリックします）。

ガイドラインとして、この設定の値は、アプリケーションが平均的に開くファイル数のおよそ 1/8 にします。たとえば、アプリケーションがたいてい 40 ファイルを開く場合、I/O スレッドを 5 に設定します。

Monitor を使用して、メニューから［MicroKernel］［**リソースの使用状況**］を選択します。ウィンドウが開き、［**ファイル数**］欄に、開いているファイル数の現在値とピーク値が表示されています。何度か現在値を記録して平均表示度数を出すことができます。その平均値に基づいて、I/O スレッド数に適切な設定をしてください。

大きいシステム キャッシュ

Windows オペレーティング システムによっては " 大きいシステム キャッシュ " 設定を提供しています。これを使用すれば、システムが空きメモリを利用して最近アクセスしたデータをキャッシュすることができます。一部の Windows サーバーではデフォルトでこの設定がオンになっています。これは、単純なファイル共有には都合がよく、積極的にシステム キャッシュを増加させます。

ファイル キャッシュ用のメモリを積極的に使用すると、Zen をスワップ アウトすることがあります。これによってデータベースに伴う多くのパフォーマンス問題が発生する可能性があります。" 大きなシステム キャッシュ " をオンにし、Zen の設定である［システム キャッシュ］の設定もオン（明示的、または［MicroKernel の最大メ

モリ使用量］をゼロに設定）にすると、パフォーマンスが低下する可能性があります。データベースのパフォーマンスを向上させるために考えられる解決策は、この " 大きなシステム キャッシュ " をオフにすることです。

この設定をオフにするには、コントロール パネルまたはマイ コンピューターのプロパティからシステム プロパティにアクセスします。［**詳細設定**］タブをクリックし、［パフォーマンス］セクションの［**設定**］ボタンをクリックします。［パフォーマンス オプション］ダイアログで［**詳細設定**］タブをクリックします。

［メモリの使用量］セクションで、［**システム キャッシュ**］オプションが選択されている場合は、大きなシステム キャッシュがオンになっています。これをオフにするには、［メモリの使用量］の**もう 1 つ**のオプションである " プログラム " を選択します。要求に応じて［OK］をクリックして開いているダイアログを閉じ、設定を反映させるためにオペレーティング システムを再起動します。

ハイパーバイザー製品でのパフォーマンス

Zen で最高のパフォーマンスを実現するには、以下の事項を順守してください。

- ハイパーバイザー ベンダーからのパフォーマンス最良実施例に従うこと。
- Zen をホストする仮想マシンに十分なメモリ（RAM）が確保されていること。
- ハイパーバイザー ホストには十分な数の仮想 CPU があり、同じマシン上の全仮想マシンの中で仮想 CPU の競合を最小化します。これにより、Zen を実行する仮想マシンとの競合が避けられます。共通（アフィニティ）規則を使用する場合は、必ずすべてのコアを同じソケットで実行するようにしてください。
- Zen データ ファイルは高速な物理ストレージに存在し、物理ストレージ デバイスに対するスピンドルの競合およびコントローラーの競合を最小化します。

データベースのグローバル化

6

Zen のグローバル化機能

この章では、以下の項目について説明します。

概要

ここでいう「グローバル化」とは、コンピューター ソフトウェアをさまざまな言語に適応させるという意味です。今や、データが世界中のユーザーによってアクセスされ、アプリケーションがそのデータをユーザー自身の使用言語で表すことが当たり前となっています。Zen のグローバル化対応によって、アプリケーションでは同一データベース内に複数の言語でテキストを保存することができるようになりました。これは、どの言語でもアプリケーションでデータを保存、処理および検索することができるということです。

この章では、作成したアプリケーションのグローバル化をサポートできるようにする Zen 機能について説明します。グローバル化への一般的なアプローチ、およびグローバル化されたアプリケーションをサポートする特定の Zen 機能について説明します。デフォルトで、Zen は従来のテキスト エンコードを使用する下位バージョンとの互換性を維持します。この章では従来のエンコードを使用したアプリケーションのグローバル化を容易にする設定について説明します。

概念と定義

このセクションでは、重要な概念の説明と、本章で使用される用語の定義を示します。

文字セット

文字セットとは、指定のハードウェアやソフトウェア システムで認識されるテキストおよびその他の記号の文字集合（リスト）です。英語で使用される文字のみを認識すればよいシステムであれば、文字セットは A ～ Z と a ～ z の文字、0 ～ 9 の数字、および数個の句読記号という小さいサイズで済みます。ほかの言語もサポートする場合は、文字セットのサイズも大きくなります。たとえば、ヨーロッパ言語ではアクセントや発音区別符号を持つ文字が追加されます。その他の言語には完全に異なる文字が存在します。

従来の文字セット

文字は、コンピューター システム内では数値として表されます（文字を数字に置き換えることを「エンコード」といいます。これについては後で説明があります）。大きな文字セットのすべての文字を表すには、大きい範囲の数値が必要です。かつては限定的で高価だったストレージ リソースを効率よく使用するために、コンピューター システムは一般的に 1 バイト（8 ビット）を使用して文字表現を格納していました。これにより、文字セットのサイズは 256 文字に制限されました。その結果、日本語などの特定の言語、あるいは西ヨーロッパなどの限定的な言語を表すために特化した従来の文字セットを持つようになりました。Zen では従来の文字セットを数多くサポートします。

Unicode 文字セット

Unicode 標準は、世界中の言語で使用されているすべての文字に対応した文字セットを定義しています（www.unicode.org を参照）。また、Unicode は文字幅、文字表記の方向、語および改行などを指定するための追加情報を定義することで文字セットの概念を拡大させています。これにより、アプリケーションでは Unicode のテキストを適切に表示および操作することができます。アプリケーションおよびデータベースでも、大文字小文字の変換や並べ替えなどの作業にこの追加情報が必要です。

Zen は Unicode 文字セットを認識し、アプリケーションでどのような言語が要求されても、その文字データを保管および検索できるようにします。

エンコード

エンコードとは、文字セットの各文字と数値を関連付けることです。当初、コンピューター システムやシステムのプログラミング言語は、文字とバイトを区別しませんでした。また、エンコードは単に特定の文字に相当するバイト値でした。1 バイトでエンコードが可能な文字数よりも多くの文字を表示するニーズに応えるために、別のエンコードが定義されました。大きい文字セットのエンコードでは 1 文字に対して複数バイトを使用する可能性があります。

従来のエンコード

従来の文字セットの場合、エンコードは**コード ページ**で定義されます。コード ページは、文字から値へ（または値から文字へ）変換するためのルックアップ テーブルのように考えることができます。テキストを使用するアプリケーションでは必ず同じコード ページを使用することが重要です。あるコード ページを使用したデータベースに格納されている文字が別のエンコードで読み取られると、別の文字として表示されることがあります。

従来のコード ページはすべて、バイナリ単位が 8 ビット バイトです。現在使用されているほとんどの（従来の）コード ページは ASCII（American Standard Code for Information Interchange）のスーパーセットで、128 個の制御コードと印刷可能文字で構成される文字セットを定義する 7 ビット コードです。8 番目のビットを使用することで、追加の 128 個のコードが、1 バイト値で合計 256 文字のエンコードに対応できます。

Microsoft Windows には ANSI および OEM という 2 つのコード ページ グループがあります。これら両グループのコード ページは拡張 ASCII コード ページですが、どちらのグループも文字とバイトを区別しません。

ANSI コード ページでは、非 ASCII 値（127 よりも大きい値）が国際的な文字を表します。この文字は通常、1 言語または言語グループ用にカスタマイズされます。一般的に、ANSI コード ページは Windows システム上のグラフィカル ユーザー インターフェイスを使用するバイト文字列アプリケーション向けに使用されます。ANSI コード ページはアクティブ コード ページ（ACP、Zen ではクライアント コード ページといいます）と呼ばれることもあります。Windows アプリケーションは常に現在アクティブなコード ページを 1 つ持ちます。たとえば、Windows で英語のアクティブ コード ページのデフォルトは 1252 です。

OEM（Original Equipment Manufacturer）コード ページは、その名称が示すように、特定のシステム向けに特定のメーカーが開発したコード ページです。もともと、これらのコード ページは MS-DOS 向けに使用されていたものですが、今でもコンソール アプリケーション用に使用されています。英語によく使用される OEM コード ページは、コード ページ 437 です。代表的な OEM コード ページには ANSI コード ページに似た文字セットがありますが、127 より大きい値用には別の文字順序を使用します。

Unicode エンコード

文字エンコード システムでは、1 つ 1 つの文字がコード ポイントと呼ばれる一意の値に割り当てられており、それをエンコード データに使用することができます。コード ポイントは面単位で構成されます。各面には、65,536 (2^{16}) 字分のコード ポイントを収録することができます。Unicode では最大 17 面が使用できるように設定されています。1 番目の面である 0 面は BMP（Basic Multilingual Plane：基本多言語面）と定義されており、現在定義されているコード ポイントの大部分がここに含まれています。このドキュメントの作成時点で、0、1、2、15、および 16 面のみにコード ポイントが含まれています。Unicode 標準ではコード ポイントのエンコード方式にいくつかの方法があります。一般的に使用される 2 つの方式は UTF-8 と UCS-2 です。UTF-8 は文字のコード ポイント値をバイト文字列にエンコードします。この場合、1 文字につき 1 バイトから 4 バイト使用します。UCS-2 は 16 ビット値（「ワイド文字」と呼ばれることが多いです）を使用して文字のコード ポイント値をエンコードします。

Zen は BMP のコード ポイントを認識します。また、Unicode エンコード（バイト文字列には UTF-8、ワイド文字の文字列には UCS-2）を使用するアプリケーションに対応しています。UTF-8 のバイナリ単位は 8 ビットです。UCS-2 のバイナリ単位は 16 ビット（ワイド文字）です。

エンコードの宣言

［データベース コード ページ］は Zen のデータベース用プロパティであり、データベースに保存する文字データのエンコードを宣言します。このプロパティの目的は、文字データを正しく解釈できるようにすることです。ただし、この［データベース コード ページ］プロパティは宣言に過ぎません。Zen は、アプリケーションがデータベースに追加するデータおよびメタデータのエンコードを検証しません。特定のエンコードで文字データが保存および検索されることを保証するのはアプリケーション側で行ってください。データベース コード ページが適用されるのは、従来のコード ページまたは UTF-8 でエンコードされたテキストのみであることに留意してください。ワイド文字のテキストは UCS-2 を使用してエンコードされます。データベース エンジンがワイド文字のテキストとバイト文字列テキスト間で変換を行うには、適切な設定が必要です。データベース コード ページのデフォルト値は、データベース エンジンが実行されているオペレーティング システムのシステム コード ページです。

Zen の SQL アクセス方法では、アプリケーションとアクセス方法との間でやり取りされるバイト文字列用の**クライアント コード ページ**を推測します。Windows の場合、このアクセス方法はアプリケーションがバイト文字列にアクティブ コード ページ（ACP）を使用していることを前提としています。Linux、macOS、および Raspbian の場合、このアクセス方法はアプリケーションがロケールのエンコードを使用していることを前提としています。通常、これは LANG 環境変数で設定されます。

Zen では、データベース エンジンとクライアント間でエンコードを確実に適合させる方法があります。たとえば、アプリケーションは、Zen SQL クライアントがデータベース コード ページとクライアント アプリケーション間でデータを自動的に変換するように指定することができます。これは、**自動変換**と呼びます。ただし、この自動変換によって文字を変換するのは、それらの文字がサーバー マシン上のコード ページとクライアント マシン上のコード ページの両コード ページの文字セットに存在する場合のみであることに注意してください。

以前のバージョンとの互換性を保つために、アクセス方法における自動変換はデフォルトで無効になっています。アプリケーションでのアクセス方法の設定で自動変更を有効にする必要があります。可能であれば、データベース コード ページを設定し、その値を読み取って使用するためのアクセス方法を設定する方法をお勧めします。

照合順序と並べ替え

照合順序（コレーション）は、文字列のソート順を決定する処理および機能として用いられる一般的な用語です。照合順序は言語によって異なるので、単純なバイナリ文字列比較で各言語の必要な照合順序を示すように、多言語のコード ページにエンコードを配置することはできません。複数レベルの並べ替えが必要な場合、個々のデータ テーブルを正しいソート順で定義する必要があります。

文字セットとエンコードの選択

一般的に、グローバル化戦略を実行するには、言語またはニーズを満たすその他のテキストや文字に基づいて要求される文字セットを特定することから始めます。次に、その文字セットをサポートするエンコードを選択します。使用されるエンコードはデータベースとクライアント アプリケーションでさえ異なるかもしれません。いくつかの例を見てみましょう。

最もグローバルな文字セットは Unicode 文字セットです。従来の文字セットがクライアントで使用されていたとしても、Zen で保存する際にそれらをすべて Unicode へ変換することができます。新規アプリケーションまたは新規モジュールの場合は、テキストを UCS-2 または UTF-8 で格納するのが簡単な方法です。しかし、すべてのアプリケーションが新規というわけではありません。

アプリケーションに対してもう 1 つ考慮しなければならないことは、クライアント プログラムのテクノロジです。アプリケーションが C/C++ で .NET Framework、Java VM、または UNICODE オプション使用する場合、そのアプリケーションは既にワイド文字の文字列を使用したテキストを処理しています。このような状況で、主に考慮する点は、Zen の設定でそのテキストを保存するようにすることと、その保存方法の選択です。

アプリケーションが C/C++ でバイト文字列を使用している、また、従来の Zen ODBC ドライバーを使用している場合、グローバル化には 2 つの経路が考えられます。1 つはアプリケーションを移植してワイド文字の文字列を使用できるようにすること。もう 1 つはインストールされたクライアントの従来のコード ページを引き続きアプリケーションがサポートするようにし、Unicode 保存用に変換できるようにすることです。

既存のアプリケーションへの対応として非常に保守的な方法は、現在のコード ページを引き続き使用し、サポートする他の言語を利用することです。たとえば、ANSI コード ページ 1252 または OEM コード ページ 850 を使用する Windows の英語圏市場向けに開発されたアプリケーションは、アプリケーション ストレージを変更しなくても西ヨーロッパ言語サポートします。主な変更はプログラムのテキストをローカライズすることです。

メモ ユーザー データとメタデータ

Zen には扱わなければならない 2 種類のテキストがあります。1 つはユーザー データです。このデータを扱うのは主にアプリケーションですが、インデックス順序付けや SQL 文字列関数でも使用します。もう 1 つはメタデータです。このデータはテーブル、列およびインデックスといった SQL オブジェクトの名前です。メタデータは UCS-2 エンコードを処理しないので、データベース コード ページの宣言による従来のコード ページに従います。SQL クエリには、文字列リテラルのユーザー データとオブジェクト名のメタデータの両方が含まれることがあります。したがって、SQL クエリを検討する場合、全体としては SQL テキスト用に Unicode エンコードの 1 つを使用している場合であっても、ユーザー データとメタデータの文字セットを区別する必要があります

Zen は、テキスト ストレージにおけるエンコードの混在に対処できません。アプリケーションではそのようなテキストをバイナリ ストレージと見なし、すべてのエンコード変換をアプリケーションで処理する必要があります。Zen では、すべての CHAR 型データと SQL メタデータがデータベース コード ページを使用すること、また、すべての NCHAR 型データが UCS-2 であることを前提としています。

以下のトピックでは、特定のストレージ状況について説明しています。

- 「Unicode UTF-8 による多言語データベースのサポート」
- 「Unicode UCS-2 による多言語データベースのサポート」

また以下のトピックでは、従来の OEM コード ページの取り扱いについて説明しています。

- 「従来のエンコードおよび OEM エンコードによる多言語データベースのサポート」

Unicode UTF-8 による多言語データベースのサポート

テキストを UTF-8 として格納するようにした場合は、リレーショナル型の CHAR、VARCHAR、および LONGVARCHAR を使用し続けることになるでしょう。また、アプリケーションを実行するオペレーティング システム、アプリケーションで利用可能な文字列操作ライブラリ、アプリケーションで利用する Zen アクセス方法、異なるデータ型を必要とする可能性がある列などの面でも Unicode サポートを考慮する必要があります。

Unicode UTF-8 を使用する場合

Unicode UTF-8 エンコードは以下のような状況に適しています。

- 既存のアプリケーションでサポートする言語を新たに追加したいが、アプリケーションへの変更は最小限に留めたい場合。たとえば、ANSI 文字のみ（英語など）を用いた Zen データベースがあり、アプリケーションを拡張して、英語、ドイツ語、ポーランド語、およびチェコ語でデータを含めることができるようにしたいとします。UTF-8 は、ヨーロッパのスクリプトの場合には、1 文字に対して必要なバイト数が多くても 2 バイトであるため、コンパクトな記憶域を提供します。
- Web アプリケーション。これは多くの Web プラットフォームで UTF-8 が使用されているからです。Unicode UTF-8 は ASCII と互換性があり、ラテン語由来の言語の文字セットにはコンパクトなので、多くの場合、Unicode テキストの交換用の標準エンコードとして使用されます。
- Linux または macOS アプリケーション。これは UTF-8 文字列処理をサポートします。
- macOS 上の Zen サーバー。

Zen における Unicode UTF-8 サポート

Zen でサポートされるコード ページの 1 つは UTF-8 です。UTF-8 のテキスト ストレージのために、お客様の Zen データベース用のデータベース コード ページには UTF-8 を設定します。

UTF-8 の場合、文字列ストレージはバイト文字列であることに留意してください。バイト文字列の場合、Zen はリレーショナル データ型の CHAR、VARCHAR、LONGVARCHAR、そして Btrieve データ型の STRING と ZSTRING を提供します。『*SQL Engine Reference*』の「データ型」も参照してください。ヨーロッパ言語は 1 文字に対して、従来のコード ページの単一バイトではなく 2 バイトを必要とすることが多いので、UTF-8 を保存するときは列が拡大されることもあります。

既存の CHAR、VARCHAR および LONGVARCHAR データ型用のアプリケーションで挿入された文字列データはすべて UTF-8 文字列として解釈されます。Zen SQL アクセス方法を設定して、自動的に UTF-8 へ変換させることができます（「Unicode UTF-8 対応のアクセス方法」を参照）。

データベース コード ページが UTF-8 で、クライアント環境が Unicode（ワイド文字または UTF-8）をサポートする場合、SQL テキストは CHAR リテラルの Unicode 文字をサポートします。その他のコード ページでは、一般的な Unicode 文字は NCHAR リテラルに含まれる必要があります。

照合順序と並べ替え

Zen ではデフォルトで、UTF-8 ストレージでの照合順序と並べ替えに対し、コード ポイント順のみをサポートします。

Unicode UTF-8 対応のアクセス方法

Zen のアクセス方法である ODBC、JDBC および ADO.NET では UTF-8 ストレージへの変換をサポートします。これらのアクセス方法では、UCS-2 ワイド文字列として、または ANSI ODBC ドライバー向けには従来のバイト文字列として、アプリケーションとテキスト値をやり取りします。適切に設定されれば、これらのアクセス方法はアプリケーションのテキスト値を、ストレージ エンジンへの転送のために UTF-8 へ変換します。

- アプリケーションが Windows で ANSI ODBC ドライバーを使用する場合は、Windows 版のドライバー マネージャーによって、すべてのデータがバイト文字列向けのクライアント コード ページ（従来のコード ページ）へ変換されます。これにより、従来の文字セットには存在しない文字がある場合、その文字は失われます。また、Unicode ODBC ドライバーを使用するために、アプリケーションを変換する必要があります。
- アプリケーションが Linux または macOS で ANSI ODBC ドライバーを使用する場合は、アプリケーションのロケールの設定で文字列エンコードとして UTF-8 を指定します。完全を期すため、接続文字列で pvtranslate=auto を宣言し、また、データベース コード ページが UTF-8 となることを宣言します。
- JDBC の場合、アプリケーションが JDBC ドライバーへの接続文字列で pvtranslate=auto を指定する必要があります。『*JDBC Driver Guide*』の「接続文字列の概要」を参照してください。
- ADO.NET の場合、アプリケーションがデータベース エンジンへの接続文字列で pvtranslate=auto を指定する必要があります。『*Data Provider for .NET Guide*』の「接続の追加」を参照してください。

既存のデータベースを Unicode UTF-8 へ移行する

すべてのテキスト データを従来のコード ページから UTF-8 へ変換する必要があります。より長い UTF-8 バイト文字列を収容するために、列は拡張する必要があるでしょう。テーブル名などの非 ASCII メタデータは、従来のコード ページから UTF-8 へ変換する必要があります。このような複合的な変更から、従来のコード ページを使用する古いスキーマから、データベース コード ページとして UTF-8 を使用する新しいスキーマへコピーすることによって、データベースを移行するのが妥当です。

メモ 既存データおよびメタデータがすべて純粋な ASCII という特殊なケースでは、データベース コード ページを UTF-8 へ変更するだけで済みます。

既存の（7 ビット）ASCII バイト文字列はすべて UTF-8 バイト文字列でもあります。

Unicode UCS-2 による多言語データベースのサポート

テキストを UCS-2 として保存するようにした場合は、リレーショナル型の NCHAR、NVARCHAR、および NLONGVARCHAR を使用することになるでしょう。これには、ほかのテキストをリレーショナル型の CHAR ファミリでも格納する能力に影響はありません。また、アプリケーションを実行するオペレーティング システム、アプリケーションで利用可能な文字列操作ライブラリ、アプリケーションで利用する Zen アクセス方法、異なるデータ型を必要とする可能性がある列などの面でも Unicode サポートを考慮する必要があります。

Unicode UCS-2 を使用する場合

Unicode UCS-2 は以下のような状況に適しています。

- アプリケーションでアジアの文字データをサポートする場合。UCS-2 では、すべての文字が 2 バイトとして格納され、アジアの文字データ用の UTF-8 よりもコンパクトなストレージを提供します。
- アプリケーションですべてのストレージをグローバル化しないので、アプリケーションに従来のバイト文字の列とワイド文字の列が一緒に含まれる場合。
- アプリケーションで、ワイド文字データと、Java、ADO.NET、またはワイド文字クライアントとの互換性を向上させる必要がある場合。そのようなアプリケーションでは、アプリケーション文字列に UCS-2 を使用します。

Zen における Unicode UCS-2 サポート

ワイド文字列ストレージは、バイト文字列ストレージとは別の種類で、バイト文字列ストレージと一緒に使用される可能性があります。

ワイド文字の文字列の場合、Zen ではリレーショナル データ型の NCHAR、NVARCHAR、および NLONGVARCHAR、そして Btrieve データ型の WSTRING と WZSTRING を提供します。NCHAR、NVARCHAR、NLONGVARCHAR、WSTRING および WZSTRING データ型対応のアプリケーションによって挿入されたデータはすべてワイド文字の文字列として解釈されます。

Zen は SQL クエリ テキストの NCHAR リテラルで Unicode 文字をサポートします。CHAR リテラルのテキスト データはデータベース コード ページへ変換されます。UTF-8 以外のデータベース コード ページは、ほとんどの Unicode 文字を処理できないことを覚えておいてください。

Unicode UCS-2 での照合順序と並べ替え

Zen ではデフォルトで、UCS-2 ストレージでの照合順序と並べ替えに対し、コード ポイント順のみをサポートします。

Unicode UCS-2 対応のアクセス方法

SQL アクセス方法で NCHAR ストレージを使用する場合、アプリケーションでそのアクセス方法に対する適切な NCHAR SQL 型を指定することが必要です。CHAR 型の使用で、データベース コード ページが UTF-8 でない場合は、NCHAR から CHAR への変換が起こり、データが失われる可能性もあります。

- ODBC アプリケーションでは、パラメーターに SQL_WCHAR、SQL_WVARCHAR および SQL_WLONGVARCHAR SQL 型を使用する必要があります。
  - Windows ODBC アプリケーションでは、ODBC に SQL_C_WCHAR として特定されるワイド文字列を使用する必要があります。また、Windows ODBC アプリケーションは Zen の Unicode ODBC ドライバーを使用する必要があります。従来の ANSI ODBC ドライバーを使用すると、Windows のドライバー マネージャーがすべてのワイド文字列をバイト文字列に変換するので、データが失われます。
  - Linux、macOS、および Raspbian の ODBC アプリケーションでは、UTF-8 エンコード設定のロケールを使用し、アプリケーションのバイト文字列（SQL_C_CHAR）で UTF-8 を使用して、NCHAR 値（SQL_WCHAR）用のデータを提供する必要があります。

- JDBC アプリケーションでは、文字列パラメーターに NVARCHAR 型を設定するか、setNString() などの "N" メソッドを使用して文字列データを NCHAR SQL 型として送る必要があります。また、接続文字列で pvtranslate=auto を設定し、データベース コード ページも設定します。そのように設定しない場合は、JDBC は下位バージョンとの互換性を保ち、文字列データには宣言された、またはデフォルトのバイト文字列エンコードを使用します。『*JDBC Driver Guide*』の「接続文字列の概要」を参照してください。
- ADO.NET アプリケーションでは、データベース エンジンへの接続文字列で pvtranslate=auto を指定する必要があります。『*Data Provider for .NET Guide*』の「接続の追加」を参照してください。

すべての場合において、SQL テキストでは NCHAR リテラルを使用します。

既存のデータベースを Unicode UCS-2 へ移行する

既存のデータベースを UCS-2 アプリケーション対応に移行するための作業には、バイト文字列からワイド文字列への変換が伴います。変換の程度は、対象のアプリケーションに応じて異なります。データベース コード ページが既存の CHAR 列のエンコードと一致するように設定されていれば、ALTER TABLE を使用して列を変換することができます。グローバル化されるデータを格納する列のみを変換する必要があります。データベース コード ページの文字セット外の文字を格納しない列は CHAR 型のままでかまいません。

アプリケーションが Windows で ANSI ODBC ドライバーを使用する場合は、アプリケーションをワイド文字データへ変換し、Zen Unicode ODBC ドライバーを使用する必要があります。

従来のエンコードおよび OEM エンコードによる多言語データベースのサポート

テキストを従来のコード ページまたは OEM コード ページを使用して格納するようにした場合は、リレーショナル型の CHAR、VARCHAR、および LONGVARCHAR を使用することになるでしょう。従来のコード ページへの変換を行うためにアクセス方法を設定する場合は、ワイド 文字アプリケーションを使用することができます。

従来のコード ページを使用する状況

従来のコード ページで、自分のアプリケーションのニーズを満たす文字セットを定義するのであれば、この従来のコード ページを使用することは、テキスト データの格納にはコンパクトで効率的な方法です。

Zen における従来のコード ページのサポート

Zen データベース エンジンは、CHAR データがデータベース コード ページでエンコードされていると見なしています。アクセス方法には、そうなっていることを保証するための設定オプションがあります。

ストレージに OEM コード ページを使用する古いアプリケーションは、データベース コード ページを宣言しませんでした。アプリケーションが適切なリレーショナル文字列関数を慎重に使用している限りは、この状況で作業し続けることができます。

従来のコード ページでの照合順序と並べ替え

Zen でのデフォルトの照合順序は、エンコードされたバイトのバイナリ順です。このデフォルトの照合順序は、宣言されたデータベース コード ページによる影響を受けません。

Zen は、照合順序の制御にオルタネート コレーティング シーケンス（ACS）とインターナショナル ソート規則（ISR）メカニズムの双方を提供します。これらは、Btrieve インデックスおよびリレーショナル列で宣言することができます。リレーショナル列で CASE 宣言を使用する、または Btrieve インデックスで case-insensitive ビットを使用する場合、Zen では常に ASCII 文字セット用に特定の ACS を使用します。

従来のコード ページ向けのアクセス方法

Zen のすべてのアクセス方法で、バイト 文字列テキストをサポートします。アクセス方法の中には、データベース コード ページがアプリケーションのコード ページと同じであることを前提としているものもあれば、コード ページを設定できるものもあります。

- ANSI ODBC ドライバーでは DSN 設定の 1 つに OEM/ANSI 変換を提供します。これにより、アプリケーションの OEM コード ページ（ANSI コード ページではなく）がデータベース コード ページであることを宣言します。ANSI ODBC ドライバーは、接続文字列の encoding プロパティと pvtranslate=auto オプションもサポートします。
- Unicode ODBC ドライバーは常に pvtranslate_true プロパティが設定されているかのように動作し、アプリケーションの CHAR データを必要に応じてデータベース コード ページに変換します。
- JDBC ドライバーには接続文字列に明示的な encoding プロパティがあります。pvtranslage=auto プロパティが指定された場合、このドライバーはエンコードをデータベースのコード ページに設定します。
- ADO.NET プロバイダーには明示的な encoding 接続プロパティがあります。pvtranslage=auto 接続プロパティが指定された場合、このドライバーはエンコードをデータベースのコード ページに設定します。
- Delphi 向けの PDAC アクセス方法には、OEM コード ページまたは ANSI（ACP）コード ページを使用して、エンジンへ CHAR データを送るかどうかを制御する OEMConversion プロパティがあります。

既存のデータベースを別のコード ページへ移行する

データベースのコード ページを変更すると、そのデータを、変更後の新しいコード ページを使用する新規データベースへコピーする必要があります。このコピーはアプリケーションで行う必要があります。Zen はトランザクション内でコード ページを変換しません。

データベースのメタデータで、変換を必要とする文字を使用している場合、その変更は新しいデータベースのスキーマの作成時に行う必要があります。この種類のメタデータの一般的な例として、テーブル名があります。

データベース コード ページとクライアント エンコード

先の「概念と定義」で説明したように、エンコードは Zen データベース エンジンと Zen クライアント アプリケーション間で文字データをどのように変換するかを指定します。Zen では、クライアントとサーバー間のエンコードの複雑性、およびオペレーティング システム、言語、アクセス方法のさまざまな組み合わせに対処します。エンコードの機能拡張は、データベース コード ページとクライアント エンコードに分割されています。この 2 種類のエンコードは、別個のものですが相互に関係しています。

データベース コード ページおよびクライアント エンコードは、リレーショナル エンジンのみに適用されます。MicroKernel エンジンには影響がありません。

データベース コード ページ

［データベース コード ページ］はデータベース用プロパティであり、データベースに保存する文字データのエンコードを指定します。デフォルトのデータベース コード ページは " サーバーのデフォルト " で、データベース エンジン実行中のサーバーのオペレーティング システム コード ページを意味します。(オペレーティング システムのコード ページは一般的に「OS エンコード」と呼ばれます。この章のこれ以降の説明ではこの表現を使用します)。

データベース コード ページは、特に、異なる OS エンコードを使用して Zen DDF を手動で別のプラットフォームへコピーしながら、データベース エンジンにメタデータを正しく変換させたい場合に役立ちます。

Zen でデータベースを作成する場合、デフォルトでは、データベース エンジンを実行しているマシン用のアクティブ コード ページを使用するようになっています。たとえば、Windows の英語版を実行しているマシンの場合、コード ページには 1252 が割り当てられます。Zen のエンコード変換は "None"（なし）に設定されます。作成したアプリケーションでは Zen データベースと同じコード ページを使用するか、エンコード変換を "Automatic"（自動）に設定しておく必要があります。

サポートされるコード ページ

Zen では以下のコード ページをサポートします。ここで挙げるページはすべてバイト文字列ストレージを使用します。

- ASCII
- EUCJP
- ISO8859_1
- UTF-8
- Windows のコード ページ
  - CP437, CP737, CP775, CP850, CP852, CP855, CP857, CP858, CP862, CP866, CP932
  - CP1250, CP1251, CP1252, CP1253, CP1254, CP1255, CP1256, CP1257, CP1258

クライアントのエンコード

クライアントのエンコードは、Zen クライアント上のアプリケーションが使用するデータ エンコードです。アプリケーションは、任意に選択したエンコードでテキストを管理することができます。データベース エンジンとクライアント アプリケーション間で互換性のあるエンコードを確立する必要があります。

データベース エンジンとクライアントで異なるエンコードを使用している場合でも、文字がサーバー マシンのコード ページとクライアント マシンのコード ページの両方に存在していれば、Zen がその異なるエンコード間で自動変換することができます。

データの変換は、要求に応じてクライアントで行われます。変換はいつも必要なわけではありません。たとえば、クライアントとサーバーの OS エンコードが一致している場合は不要です。

ZenCC におけるエンコードのサポート

データベースの作成時にデーターベースのコード ページを設定する、または既存のデータベースのコード ページ設定を編集するために、ZenCC を使用することができます。

メモ データベース コード ページのプロパティを変更しても、データベースのデータが変更されることはありません。ただし、既存のデータベースのデータベース コード ページを変更すると、既存のデータ項目の解釈方法に影響があります。

Zen Control Center（ZenCC）はそれ自体、データベース エンジンのクライアント アプリケーションです。クライアントとして、ZenCC では、各データベース セッションで ZenCC がメタデータおよびデータを読み取りまたは追加する際に使用するエンコードを指定することができます。既存データベースのデフォルトでは、ZenCC が実行されているマシンのエンコードを使用します。これは、ZenCC の旧来の動作です。新しいデータベースのデフォルトでは、自動変換を使用します。『*Zen User's Guide*』の 「ZenCC 接続エンコード」を参照してください。

次の表では、ZenCC 接続エンコードとデータベース コード ページの設定の相互の影響を説明します。ZenCC 接続エンコードは ZenCC にのみ適用されます。ほかのクライアント アプリケーションには影響しません。

ZenCC 接続エンコードが特定のエンコードに設定されている	ZenCC 接続エンコードが自動変換に設定されている
ZenCC はデータベース コード ページを無視し、指定されたエンコードを使用して CHAR 型のデータ、文字列、リテラル、およびメタデータを読み書きします。NCHAR 型のデータの場合は、この設定による影響はありません。これは、ZenCC の旧来の動作です。	ZenCC およびデータベースは、CHAR データとメタデータのエンコードを自動的に確立します。クエリ内の文字列リテラルは Unicode としてエンジンに送られます。NCHAR 型のデータの場合は、この設定による影響はありません。

Btrieve API におけるエンコードのサポート

Btrieve API を使用する場合は、アプリケーションで使用されるローカル エンコードでファイル名とパスを提供する必要があります。Btrieve API は、サーバーおよびクライアント上の OS エンコード間の相違に対処します。

DTI におけるエンコードのサポート

DTI（Distributed Tuning Interface）を使用する場合は、アプリケーションで使用されるローカル エンコードでファイル名とパスを提供する必要があります。DTI は、サーバーおよびクライアント上の OS エンコード間の相違に対処します。

DTI API を使用してデータベースを作成する場合、作成時にデータベース コード ページのプロパティを指定できます。このプロパティは、文字データの自動変換を設定するために SQL アクセス方法で使用できます。

ADO.NET におけるエンコードのサポート

.NET Framework および .NET アプリケーションでは UTF-16 文字列を使用します。これらは、CHAR 列にテキストを格納する場合、コード ページに変換される必要があります。

接続プロパティ PVTranslate=Auto は、接続エンコードをデータベース コード ページに設定します。encoding プロパティを直接設定することもできます。

詳細については、『*Data Provider for .NET Guide*』の「接続の追加」、「PsqlConnectionStringBuilder オブジェクト」および「文字セットの変換」を参照してください。

JDBC におけるエンコードのサポート

Java 仮想マシンや Java アプリケーションでは UTF-16 文字列を使用します。これらは、CHAR 列にテキストを格納する場合、コード ページに変換される必要があります。

接続プロパティ PVTranslate=Auto は、接続エンコードをデータベース コード ページに設定します。encoding プロパティを直接設定することもできます。

PvTranslate=Auto プロパティが指定されると、JDBC ドライバーは文字列リテラルを Unicode としてエンジンに送ります。この設定がなければ、従来の動作として文字列リテラルがデータベース コード ページに変換されます。アプリケーションで NCHAR 文字列リテラル（例："N'ABC'"）を使用する場合は、接続プロパティの設定を PvTranslate=Auto にしてください。

『*JDBC Driver Guide*』の「接続文字列の要素」を参照してください。

ODBC におけるエンコードのサポート

Zen ODBC ドライバーは、多くのメカニズムをサポートしてクライアント エンコードを制御します。

DSN を設定する場合、エンコード オプションとして " 自動 "、"OEM/ANSI"、および " なし " から選択することができます。" 自動 " を設定すると、ドライバーはクライアント エンコードからデータベース コード ページへ変換するようになります。"OEM/ANSI" を設定すると、ドライバーはクライアント エンコードを対応する OEM コード ページへ変換するようになります。" なし " に設定すると、ドライバーがテキストを変換できないようにします。詳細については、『*ODBC Guide*』の「エンコード変換」を参照してください。

OEM から ANSI へのデータ変換に使用する従来の変換方法

データベース内に OEM 文字データがある場合、従来の解決法はアクセス方法で OEM/ANSI 変換を指定することでした。ここでは、OEM 文字データを使用する Linux クライアント向けに従来の方法について説明します。

メモ 従来の方法は引き続きサポートされますが、前に説明したようにデータベースに OEM コード ページを指定し、アクセス方法で自動変換を使用することをお勧めします。

『*ODBC Guide*』の「OEM/ANSI 変換」も参照してください。

ODBC を使用する場合、エンコードは、Win32 の OS では SHIFT-JIS を使用してください。

日本語版の Linux は一般的にデフォルトでそのエンコードを EUC-JP または UTF-8 に設定します。

日本語版の Linux を使用している場合、クライアントは別の Linux サーバー（たとえばローカルに）、あるいは Win32 の SHIFT-JIS サーバーに接続できます。Linux サーバーに置かれている SHIFT-JIS にエンコードされたデータベースに接続することも可能です。

以下に示す手順を使用して必要な設定を行ってください。これらのケースは、アプリケーション自体は何も変換を行わず、そのマシンのネイティブのエンコードを使用することを前提としています。

- 「Linux EUC-JP クライアントを Win32 SHIFT-JIS サーバーへ接続させる」
- 「Linux UTF-8 クライアントを Win32 SHIFT-JIS サーバーへ接続させる」
- 「Linux EUC-JP クライアントを Linux EUC-JP サーバーへ接続させる」
- 「Linux UTF-8 クライアントを Linux UTF-8 サーバーへ接続させる」
- 「Linux UTF-8 クライアントを Linux EUC-JP サーバーへ接続させる」
- 「Linux EUC-JP クライアントを Linux EUC-JP サーバーへ接続させる、サーバーにデータを保存する場合は SHIFT-JIS エンコードを使用する」

Linux EUC-JP クライアントを Win32 SHIFT-JIS サーバーへ接続させる

このサーバーで受け取るものはすべて SHIFT-JIS でなければいけません。サーバーからクライアントに送られるものはすべて EUC-JP でなければいけません。

これを達成するには、指定のデータベースへの接続に使用する ODBC.INI（デフォルトで `/usr/local/actianzen/etc` にあります）内のクライアント DSN 設定が次のようになっている必要があります。

```
[dbclient]
Driver=/usr/local/actianzen/lib/libodbcci.so
```

```
Description=Zen ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/actianzen/lib/libxlate.so.10
TranslationOption=90000932
```

TranslationDLL の行では、ODBC クライアント インターフェイスが使用する変換ライブラリを設定します。

TranslationOption の行は、9000（EUC-JP）から 0932（SHIFT-JIS）への変換が必要であることを指定しています。

この例を使用すると、クライアントからのすべてのデータはサーバーに送られる前に SHIFT-JIS に変換され、サーバーがクライアントにデータを送る前には EUC-JP に変換されます。

Linux UTF-8 クライアントを Win32 SHIFT-JIS サーバーへ接続させる

このサーバーで受け取るものはすべて SHIFT-JIS でなければいけません。サーバーからクライアントに送られるものはすべて UTF-8 でなければいけません。

これを達成するには、指定のデータベースへの接続に使用する ODBC.INI（デフォルトで /usr/local/actianzen/etc にあります）内のクライアント DSN 設定が次のようになっている必要があります。

```
[dbclient]
Driver=/usr/local/actianzen/lib/libodbcci.so
Description=Zen ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/actianzen/lib/libxlate.so.10
TranslationOption=90010932
```

TranslationDLL の行では、ODBC クライアント インターフェイスが使用する変換ライブラリを設定します。

TranslationOption の行は、9001（UTF-8）から 0932（SHIFT-JIS）への変換が必要であることを指定しています。

この例を使用すると、クライアントからのすべてのデータはサーバーに送られる前に SHIFT-JIS に変換され、サーバーがクライアントにデータを送る前には UTF-8 に変換されます。

Linux EUC-JP クライアントを Linux EUC-JP サーバーへ接続させる

この設定を使用する場合は、DSN の定義を変更する必要はありません。dsnadd ツールで作成されたままの DSN を使用します。

Linux UTF-8 クライアントを Linux UTF-8 サーバーへ接続させる

この設定を使用する場合は、DSN の定義を変更する必要はありません。dsnadd ツールで作成されたままの DSN を使用します。『*Zen User's Guide*』の 「dsnadd」を参照してください。

Linux UTF-8 クライアントを Linux EUC-JP サーバーへ接続させる

このサーバーで受け取るものはすべて EUC-JP でなければいけません。サーバーからクライアントに送られるものはすべて UTF-8 でなければいけません。

これを達成するには、指定のデータベースへの接続に使用する ODBC.INI（デフォルトで /usr/local/actianzen/etc にあります）内のクライアント DSN 設定が次のようになっている必要があります。

```
[dbclient]
Driver=/usr/local/actianzen/lib/libodbcci.so
Description=Zen ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/actianzen/lib/libxlate.so.10
TranslationOption=90019000
```

TranslationDLL の行では、ODBC クライアント インターフェイスが使用する変換ライブラリを設定します。

TranslationOption の行は、9001（EUC-JP）から 9000（UTF-8）への変換が必要であることを指定しています。

この例を使用すると、クライアントからのすべてのデータはサーバーに送られる前に EUC-JP に変換され、サーバーがクライアントにデータを送る前には UTF-8 に変換されます。

Linux EUC-JP クライアントを Linux EUC-JP サーバーへ接続させる、サーバーにデータを保存する場合は SHIFT-JIS エンコードを使用する

この状況は Win32 エンジンに SHIFT-JIS データベースがある場合に可能ですが、すべてのファイルを Linux EUC-JP サーバーに移動することができます。この場合、そのデータベースは EUC-JP Linux マシンにありますが、DDF ファイル内のすべてのデータおよびデータ ファイルは SHIFT-JIS です。

この場合、DSN は以下のようになります。

```
[dbclient]
Driver=/usr/local/actianzen/lib/libodbcci.so
Description=Zen ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/actianzen/lib/libxlate.so.10
TranslationOption=90000932
CodePageConvert=932
```

最後の行は、サーバーでは EUC-JP エンコードを使用するが、このサーバー上のデータは SHIFT-JIS として処理することを指定しています。

ワイド文字対応 ODBC ドライバー用のエンコードのサポート

Zen ではワイド文字データ用のドライバーを使用する ODBC で UCS-2 をサポートし、また DSN エンコード変換用のデフォルトをサポートします。『*ODBC Guide*』の「エンコード変換」および「ODBC 接続文字列」を参照してください。

ワイド文字データを扱うアプリケーション用の ODBC ドライバー

Zen は、ワイド文字データを使用する 32 ビットおよび 64 ビット アプリケーション向けの ODBC ドライバーを提供します。このドライバーは Windows オペレーティング システムのみが対象で、以前からあるドライバーのセットに追加されます。

表 14　ワイド文字データ用の Zen ODBC ドライバー

ドライバー名	説明
Zen ODBC Unicode Interface	• ローカルまたはリモートの名前付きデータベースへ接続します。 • 32 ビット ODBC アドミニストレーターでは、ワイド文字データを扱う 32 ビット アプリケーション向けの 32 ビット DSN を作成します。32 ビット ドライバーは Zen の全エディションでインストールされます。 • 64 ビット ODBC アドミニストレーターでは、ワイド文字データを扱う 64 ビット アプリケーション向けの 64 ビット DSN を作成します。64 ビット ドライバーは、64 ビット プラットフォームへ Zen をインストールする場合、全エディションでインストールされます。

Linux では通常、システム エンコードは UTF-8 です。このエンコードを使用すると SQL テキストに Unicode 文字のコード ポイントを含めることができます。アプリケーションでは、UTF-8 対応の Zen ODBC Client Interface ドライバーを使用できるので、Zen ODBC Unicode Interface ドライバーは Linux で使用できません。Linux アプリケーションは、ワイド文字データを UTF-16 文字列（SQL_C_WCHAR）として、または SQL_C_CHAR としてシステム エンコード（通常は UTF-8）へ変換を要求することで処理できます。UTF-8 を使用する SQL テキストは既存の Pervasive ODBC Client Interface ドライバーと互換性があるので、Linux で ODBC ドライバーを追加する必要はありません。

DSN エンコード変換のデフォルト

DSN 用のエンコード変換オプションは、Zen データベース エンジンと ODBC を使用する Zen クライアント アプリケーション間で文字データをどのように変換するかを指定します。エンコード変換のデフォルトは、使用する Zen ODBC ドライバーによって異なります。

表 15　DSN エンコード変換のデフォルト

ドライバー名	エンコード変換のデフォルト	備考
Zen ODBC Unicode Interface	自動	接続文字列パラメーター Pvtranslate もデフォルトで "auto" にします。
Zen ODBC Interface	なし	前バージョンの Zen でのデフォルトと同じ。
Zen ODBC Client Interface	なし	前バージョンの Zen でのデフォルトと同じ。
Zen ODBC Engine Interface	なし	前バージョンの Zen でのデフォルトと同じ。

ODBC ドライバーは、ドライバーや DSN エンコード変換の設定に応じて、SQL テキストを処理します。

表 16　SQL テキストに影響を与える Zen ODBC ドライバーと DSN エンコード変換設定

設定	SQL テキストの処理	Zen ドライバー		
		ODBC Unicode Interface	ODBC Interface および ODBC Client Interface	ODBC Engine Interface
自動	SQL テキストは UTF-8 に変換され、データベース エンジンへ送られます。クライアント、サーバー、およびデータベースのコードページは無視されます。	Yes[1]	No	No
	SQL テキストはデータベース コード ページに変換され、データベース エンジンへ送られます。	No	Yes	Yes
なし	SQL テキストはクライアントとデータベース エンジン間で変換されません。[2]	Yes	Yes	Yes
OEM/ANSI	クライアント コード ページの SQL テキストは OEM/ANSI エンコードに変換され、データベース エンジンへ送られます。	Yes[3]	Yes[3]	Yes[3]

[1] エンコード変換の設定を " 自動 " にすると、ワイド文字データを持つ NCHAR 列および NCHAR リテラルを使用することができます。

[2] これはクライアントとデータベース エンジンが同じオペレーティング システムのエンコードを使用していることが前提です。

[3] SQL テキストがワイド文字の場合、最初にそれがクライアント エンコードへ変換されます。SQL テキストがワイド文字でない場合、既にそれはクライアント エンコードになっています。その後、SQL テキストは OEM エンコードに変換され、データベース エンジンへ送られます。

Zen ユーティリティにおける Unicode のサポート

Zen Control Center における Unicode サポート

「ZenCC におけるエンコードのサポート」も参照してください。

ファイルを開く / ファイルを保存用のダイアログ

ZenCC で使用する、SQL ドキュメントを開く / 保存用、エクスポートされるスキーマの保存、およびテーブル データのインポート / エクスポート用のダイアログでは機能が改善され、さまざまなファイル エンコードに対応できるようになりました。以前のバージョンでは、これらのファイルにはデフォルトのシステム コード ページが用いられるとみなされていました。ファイルの保存時に複数の Unicode エンコードを選択することはできません。本バージョンの新しいダイアログでは、ファイルを開くときに、そのファイルが Unicode エンコードを識別するバイト オーダー マーク（BOM）を使用するかどうかを検出します。また、ファイルを開くダイアログでは、ファイルに対して必要なエンコードを設定することもできます。新しい ZenCC 設定では、これらのダイアログで使用されるデフォルトのエンコードを管理できるようになっています。

これらの新機能の詳細については、『*Zen User's Guide*』の「［ファイルを開く］ダイアログと［ファイルを保存］ダイアログ」、「データのインポート / エクスポート、スキーマのエクスポートでサポートされるワイド文字データ」および「ファイル エンコードの初期設定」セクションを参照してください。

バルク データ ユーティリティ（BDU）

バルク データ ユーティリティ（BDU）はコマンド ライン ツールで、区切り文字付きテキスト ファイルのデータを Zen テーブルに読み込むことができます。データ ファイルの読み込み時に使用するデータ エンコードを指定するために、「**-c エンコード**」というコマンド ライン パラメーターが提供されます。エンコード オプションは UTF-8、UTF-16LE、および UTF-16BE です。データ ファイルに BOM（バイト オーダー マーク）が含まれている場合、BDU はその BOM で指定されたエンコードを用います。つまり、コマンド ラインで encoding パラメーターの値を指定したとしても、データ ファイルで BOM を使用して UTF-8、UTF-16LE、または UTF-16BE のエンコードを示していた場合は、BDU はそのエンコードを優先して使用します。BOM または -c パラメーターがない場合、BDU はデフォルトでシステム コード ページを使用します。

『*Zen User's Guide*』の 「bdu」を参照してください。

照合順序と並べ替えのサポート

照合順序と並べ替えとは

照合順序（コレーション）とは、コード ページ内の文字に対応する、一連のバイナリ値の並び順を定めたものです。照合順序は、重要な違いがあることがあります。たとえば、コード ページによっては、数字、文字の順に配置されることもあれば、文字、数字の順で配置されるものもあります。並べ替えは、テキストが照合順になるように、データを再配置することです。

Zen はバイト文字列のテキスト セグメントに対する名前付き照合順序の仕様をサポートします。インデックスは、指定された照合順序に従ってレコード キーを並べ替えます。

照合順序を指定しない場合のソート順序

照合順序が指定されていない場合、Zen はデフォルトで、コード ポイント順に文字を並べ替えます。デフォルトは昇順、つまり、最小値から最大値の順です。これを降順に変更することができます。詳細については、『*Zen Programmer's Guide*』の「ソート順序」を参照してください。

ワイド文字を含む列における照合順序のサポート

Zen では、コード ポイント順に従って Unicode データのデフォルトの照合順序をサポートします。UTF-16 に加えて、マルチバイトの UTF-8 テキストをコード ポイント順にソートすることができます。

オルタネート コレーティング シーケンス（ACS）を使用した照合順序のサポート

コード ページのデフォルトの照合順序とは別の照合順序を指定することができます。このユーザー定義のオルタネート コレーティング シーケンス（ACS）は、コード ページの照合順序と指定の照合順序間のマッピングです。STRING、LSTRING および ZSTRING 型の文字列キーの照合順序を決定するために 1 つまたは複数の代替シーケンスを定義できます。たとえば、ユーザー定義の ACS を使用して、数字、文字の順で配置する、または大文字と小文字の順序付けを変更する照合順序を指定することができます。Zen には、小文字を大文字にマップして、同等にソートされるようにするための upper.alt という ACS が付属しています。同じ結果は、大文字と小文字を区別しないと設定することによっても得られますが、この例は ACS で何ができるかを示しています。

基本的に、ユーザー定義の ACS は、文字のコード ページの順序位置と、代替の順序位置を関連付けるテーブルです。ACS の作成については、『*Zen Programmer's Guide*』の「オルタネート コレーティング シーケンス」で説明しており、例も提供しています。データ ファイルのレイアウトの定義で、キー値フィールドの ACS を指定します。本ガイドの「キーのオルタネート コレーティング シーケンスの指定」、および『*Zen Programmer's Guide*』の「データ レイアウト」を参照してください。

ACS の設定についての詳細は、『*Btrieve API Guide*』の 「Create（14）」、「Create Index（31）」および 「Get Next Extended（36）」、『*DDF Builder User's Guide*』の「オルタネート コレーティング シーケンス（ACS）ファイル」および『*SQL Engine Reference*』の 「SET DEFAULTCOLLATE」を参照してください。

インターナショナル ソート規則（ISR）を使用した照合順序のサポート

ACS のもう 1 つのタイプはインターナショナル ソート規則（ISR）です。ISR は、言語固有のソート順用にあらかじめ定義されたオルタネート コレーティング シーケンスです。ISR を使用すれば、ドイツ語などの ä、ö、ü（ae、oe、ue としてソート）および ß（ss としてソート）の文字を持つ言語を正しくソートできます。Zen は、インストールに含まれる collate.cfg ファイルで多くの ISR テーブルを提供しています。これらの使用例については、『*Zen Programmer's Guide*』の「インターナショナル ソート規則を使用した照合順序のサンプル」に記載されています。詳細については、前述のオルタネート コレーティング シーケンスの参照情報をご覧ください。

ICU Unicode 照合順序を使用した照合順序のサポート

Zen は、UTF-8 または UTF-16 のデータをデフォルトのバイナリ照合順序以外で並べ替える必要がある場合に使用するための、2 つの Unicode 照合順序をサポートしています。これら代替の Unicode 照合順序は、ICU（International

Components for Unicode）ライブラリのリリース バージョン 54 に基づいています。次の表は、照合順序をまとめたものです。

ICU 照合順序名	インストールされたファイル	説明
u54-msft_enus_0	u54-msft_enus_0.txt	ISR 照合順序の MSFT_ENUS01252_0 をエミュレートします。エミュレーションは、Unicode の 1252 サブセットにのみ適用されます。この範囲外の文字は、ICU root 照合順序に従って並べ替えられます。
root	icudt54l.dat	デフォルトの ICU 照合順序と、その他の構成データを定義します。

ICU 照合順序は ISR テーブル名と同じように使用されますが、`PVSW_` や `MSFT_` で始まる名前を持たない点が異なります。名前は、プレフィックス `u54-` を付けた名前にするか、または単に `root` とする必要があります。また、これらの照合順序は次の Unicode データ型にのみ適用することができます。

- STRING（UTF-8 と見なす）
- ZSTRING（UTF-8 と見なす）
- WSTRING
- WZSTRING

デフォルトの Zen インストールでは、collate.cfg ファイルと同じ場所に 2 つの ICU 照合順序が提供されます。一般的なソートの場合、デフォルトの ICU 照合順序は root という名前で参照され、その構成データは icudt54l.dat ファイルに存在します。ロケール固有のソートの場合、補足データは、ファイル名が u54 で始まり .txt 拡張子で終わるファイルに存在します。Zen で現在サポートされる補足 ICU 照合順序は 1 つのみです。

ICU 照合順序の詳細については、ICU プロジェクトの Web サイトを参照してください。

ロケールのサポート

グローバル化の重要な側面の 1 つはロケールです。これはネイティブ言語環境のモデルと定義です。ロケールは、国によって決まる形式、またはその他の規格が存在する多くのカテゴリで構成されています。たとえば、ロケールは日付と時刻の表示形式の規則、通貨規則、数値の書式規則、および照合順序を定義します。オペレーティング システムによって、ロケールはリージョン（地域）と呼ばれることもあります。

複数のロケールを、特定の言語と関連付けることができます。これにより、地域的な差異にも対応できます。たとえば、英語ではアメリカ英語のロケールとイギリス英語のロケールを持つことができます。

文字列関数を実行する場合、Zen ではデータベース エンジンを実行しているオペレーティング システムのロケールを使用します。Zen では、Zen アクセス方法の 1 つを用いたアプリケーションからの要求でデータ型を変換する際は、クライアントのロケールを使用します。

詳細については、『*SQL Engine Reference*』の 「SET DECIMALSEPARATORCOMMA」、「小数点の記号のカンマ」および 「SET TIME ZONE」を参照してください。

参照整合性の設定

7

参照整合性の構造について

参照整合性（RI）は、データベースに作成することのできる抑制と均衡のシステムであり、これによって、関連付けられたデータがあるテーブル同士を確実に同期させます。

- 「参照整合性（RI）の概念」
- 「主キーの設定」
- 「外部キーの設定」
- 「Btrieve およびリレーショナル制約間の相互作用」

参照整合性（RI）の概念

参照整合性（RI）を使用すると、同一のフィールド値がそのテーブルまたはほかのテーブルに存在するかどうかに基づき、データの変更または、変更や追加や削除の禁止を行うことができます。

定義

参照整合性についての理解は、規則の概念、主キー、外部キー、カスケード規則、および制限規則を理解することにかかっています。このトピックではこれらの定義を説明します。

規則

規則は、原因とその効果の単純なステートメントで、データベースに定義された RI システムによって実行されます。

例 A

たとえば、**削除規則**によって、主キーを持つレコードが削除されたときに、外部キーを持つレコードに何が起きるかが次のように定義されているとします。「'Bhargava Building' を含むレコードが削除されたとき、そのレコードを参照するテーブル A のすべての行は削除される。」

削除規則は、主キー値を参照する外部キーがある場合、その主キー値を含む行が削除されるのを禁止することもできます。

例 B

更新規則は、ユーザーがレコードを更新またはレコードを追加しようとしたときに、外部キーを含むレコードに何が起こるかを定義したもので、次のようなものです。「ユーザーがテーブル B に新しいレコードを追加しようとしたとき、ビル名がテーブル C に存在しなければ、追加することを拒否する。」

主キー

主キーは規則が従属する 1 つまたは複数の列です。どのテーブルでも主キーは 1 つだけ許可されており、重複値は許可されていません。更新規則では、主キーというのは、レコードの更新または追加が許可されるかどうかを判断するために、更新または追加した列と比較される列のことです。

「例 A」では、"Bhargava Building" を含む列が主キーです。

「例 B」では、テーブル C のビル名を含む列が主キーです。

外部キー

外部キーは、処理方法を決定するために主キーと比較される 1 つまたは複数の列です。

前述の「例 A」では、テーブル A の "Bhargava Building" を含む列が外部キーです。

前述の「例 B」では、テーブル B のビル名を含む列が外部キーです。

カスケード規則

カスケード規則は、データベースがある操作が起こることを許可し、ほかのテーブルまたは行を最初の操作と同期をとるように変更することによって RI を確実にする規則です。たとえば、**カスケード削除規則**が定義されていると、主キー テーブルでのレコードの削除により、データベースは削除された行の主キーと同じ値の外部キーを持つデータベース中のすべての行を検索して削除します。

制限規則

制限規則は、データベースに既に存在する値に基づく操作を許可するかどうかをデータベースが決定するのに使用する規則です。たとえば、**更新制限規則**が定義されている場合、外部キーを含む行をテーブルに追加しようとすると、データベース エンジンは外部キー フィールドの値を主キーの値と比較します。同じ値の主キーを持つ行が存在しない場合、その行を外部キー テーブルに追加することは許可されません。

キーおよび規則について

このトピックでは、主キーおよび外部キーの概念についてより詳細に説明します。

表 17　主キーおよび外部キー

テーブル A		テーブル B	
student_ID	Name	stud_ID	Class
20543	John	20543	ENG -101
20577	Mary	20543	AST-202

上記の例では、テーブル A の student_ID（A.student_ID）は IDENTITY データ型で、2 つの行が同じ値を持つことを許可していません。学生はそれぞれユニークな ID 番号を持ちます。student_ID をテーブル A の主キーとして定義します。

それから、テーブル B の stud_ID（B.stud_ID）を A.student_ID を参照する外部キーとして定義します。stud_ID のデータ型は、IDENTITY と比較できる INTEGER などのデータ型である必要があります。主キーと外部キーのデータ型には互換性が必要です。参照整合性を確実にするために、必要なだけの外部キーをいくつでも設定することができます。複数の外部キーは同一の主キーを参照することができます。

主キーを持つテーブルは親テーブルと呼ばれ、外部キーを持つテーブルは子テーブルと呼ばれます。いったんキーが定義されると、表 18 に示すように選択できる動作の範囲が限られます。必要なだけ規則を定義することができますが、タイプごとには 1 つしか定義できません。たとえば、削除制限規則を定義した場合、同じキーにカスケード削除規則を定義することはできません。これは、2 つの動作が相互に排他的であるためです。

表 18　RI 規則の選択

行いたい動作	定義する規則
B.stud_ID の値が A.student_ID のいずれかの値に一致しなければ、テーブル B に行が追加されたり更新されたりすることを許可しない。	更新制限
B.stud_ID の値がその行と一致しなければ、テーブル A の行を削除することを許可しない。	削除制限
テーブル A から行が削除された場合、その削除された A.student_ID の値に一致する B.stud_ID のすべての行をテーブル B から削除する。	削除カスケード

更新制限

前述の例で説明すると、更新制限規則を設定することにより、新規または更新された行の B.stud_ID の値は必ず A.student_ID にも存在していることが保証されます。テーブル B に行を追加できるようにするには、その前に、テーブル A に行があることが必要です。言い方を変えると、子行を作成する前に少なくとも 1 つの親行を作成する必要があります。

削除制限

例では、削除制限規則を設定することにより、テーブル A の行は、テーブル B のいずれかの行がその行を参照している場合は削除できないことが保証されます。Name の値が "John" である行は、John の student_ID がテーブル B で参照されているため、削除できません。

John の student_ID を参照するテーブル B の行がすべて削除されたら、テーブル A から John の行を削除することができます。

削除カスケード

例では、削除カスケード規則を設定することにより、Name 値が "John" の行が削除された場合、テーブル B の両方の行が削除されることを保証します。

Zen では、自己参照するテーブルに対し、循環するカスケード削除を使用できます。このため、削除カスケードは慎重に使用してください。親テーブル、子テーブル、または両テーブルのすべてのレコードを不用意に削除しないようにしてください。

このようなカスケード削除がどのようにして発生するかをわかりやすくするために例を示します。2 つの列を持つテーブル d3 を作成するとします。

```
CREATE TABLE d3 (c1 INT PRIMARY KEY, c2 INT)
INSERT INTO d3 VALUES (2,2)
INSERT INTO d3 VALUES (3,2)
INSERT INTO d3 VALUES (1,3)
INSERT INTO d3 VALUES (4,1)
```

次に、外部キーを追加し、カスケード削除規則を設定するようにテーブルを変更します。

```
ALTER TABLE d3 ADD FOREIGN KEY (c2) REFERENCES d3 ON DELETE CASCASE
```

次のステートメントにより、テーブル内の**すべて**の行が削除されます。

```
DELETE FROM d3 WHERE c1 = 2
```

c1 = 2 の行だけでなく、すべての行が削除されるのはなぜでしょう？

削除カスケードは、削除された主キーと同じ値の外部キーを持つすべての行を削除します。2 番目の行は最初の行と外部キー関係があります。同様に、3 番目の行は 2 番目の行と、4 番目の行は 3 番目の行と外部キー関係があります。外部キー関係があることで、すべての行に渡ってカスケード削除規則が適用されるため、1 行目により 2 行目が、2 行目により 3 行目が、そして 3 行目により 4 行目が削除されることになります。

Zen は、相互に参照するテーブルでの循環削除カスケードを**許可していません**。たとえば、テーブル d1 と d2 があるとして、次のようなシナリオを考えます。

```
CREATE TABLE d1 (c1 INT PRIMARY KEY, c2 INT)
CREATE TABLE d2 (e1 INT PRIMARY KEY, e2 INT)
```

次の変更ステートメントは許可されます。

```
ALTER TABLE d1 ADD FOREIGN KEY (c2) REFERENCES d2 ON DELETE CASCADE
```

次の変更ステートメントは**許可されません**。テーブル d1 と d2 には既に削除カスケードの関係があるからです。

```
ALTER TABLE d2 ADD FOREIGN KEY (e2) REFERENCES d1 ON DELETE CASCADE
```

主キーの設定

主キーは、SQL ステートメントまたは ZenControl Center を使用して作成することができます。『*Zen User's Guide*』の「列のタスク」を参照してください。

テーブル作成中に主キーを作成する

CREATE TABLE ステートメントで PRIMARY KEY キーワードを使用して、テーブル作成時に主キーを作成することができます。主キーは 1 つまたは複数の列で構成されます。次の例では id が作成され、主キーに指定されることを示しています。

```
CREATE TABLE mytable (id INTEGER,
   myname CHAR(20),
   PRIMARY KEY(id))
```

次の例は複数の列を使ってユニークな値を持つ主キーを作成する方法を示しています。

```
CREATE TABLE mytable (id INTEGER,
   myname CHAR(20),
   PRIMARY KEY(id, myname))
```

主キーに指定した単一または複数の列に UNIQUE 属性を指定したかどうかに関わらず、データベース エンジンはその列に重複値とヌル値を許可しないインデックスを自動的に作成します。どのような場合もキー列にヌル値は許可されません。すべての主キー値はユニークです。

ほかの例については、『*SQL Engine Reference*』で CREATE TABLE を参照してください。

既存のテーブルに主キーを追加する

ZenCC を使用して、または ADD PRIMARY KEY を使った ALTER TABLE ステートメントによって、既存のテーブルに主キーを追加することができます。『*Zen User's Guide*』の「列に主キーを設定または削除するには」および「SQL Editor」を参照してください。

主キーは重複およびヌル値を許可しない列にしか作成することができません。

必要な場合には、列属性を変更し、その列を主キーに設定することを同時に行うことができます。SQL を使った例を示します。

```
ALTER TABLE mytable MODIFY id INTEGER UNIQUE NOT NULL PRIMARY KEY
```

複数の列から成る主キーを追加したい場合は、キーを分けて追加する必要があります。

```
ALTER TABLE mytable ADD PRIMARY KEY(id, myname)
```

ほかの例については、『*SQL Engine Reference*』の 「ALTER TABLE」を参照してください。

外部キーの設定

外部キーは、SQL ステートメントまたは Zen Control Center を使用して作成することができます。外部キーを作成するときに、同時に関連する規則も定義できます。同一のキーに複数の規則を定義することができます。関連する規則を指定しないで外部キーを作成した場合、更新と削除はどちらもデフォルトの参照整合性によって制限されます。

テーブル作成中に外部キーを作成する

列の定義で REFERENCES キーワードを使用して、テーブル作成時に外部キーを作成することができます。外部キーは 1 つまたは複数の列で構成されます。列のデータ型は、この外部キーが参照する主キーと同じでなければなりません。次の例では your_id が作成され、mytable.id を参照する外部キーに指定されることを示しています。

```
CREATE TABLE yourtable (your_id INTEGER REFERENCES mytable(id) ON DELETE CASCADE,
   yourname CHAR(20))
```

ステートメントの最後に外部キーの指定を追加することもできます。キーに複数列を使用する場合、この技法を使う必要があります。

```
CREATE TABLE yourtable (your_id INTEGER,
   yourname CHAR(20),
   FOREIGN KEY(your_id, yourname) REFERENCES
   mytable(id, myname) ON DELETE CASCADE)
```

外部キーを作成すると、データベース エンジンは指定された列にインデックスを追加します。

ほかの例については、『*SQL Engine Reference*』の 「CREATE TABLE」を参照してください。

既存のテーブルに外部キーを追加する

ZenCC を使用して、または ADD FOREIGN KEY を使った ALTER TABLE ステートメントによって、既存のテーブルに外部キーを追加することができます。『*Zen User's Guide*』の「外部キーのタスク」および 「SQL Editor」を参照してください。

次の例ではこの外部キーに削除規則と更新規則の 2 つの規則が定義されています。

```
ALTER TABLE yourtable ADD FOREIGN KEY (your_id,yourname) REFERENCES
   mytable(id,myname) ON DELETE CASCADE ON UPDATE RESTRICT
```

DELETE CASCADE は慎重に使用してください。「削除カスケード」に記載されている例を参照してください。

ほかの例については、『*SQL Engine Reference*』の 「ALTER TABLE」を参照してください。

Btrieve およびリレーショナル制約間の相互作用

Zen は、リレーショナル エンジンと MicroKernel エンジンの両方を介して同一データへの同時アクセスをサポートするように設計されていますが、リレーショナル（SQL）データベース アーキテクチャの機能によっては、そのデータへの Btrieve アクセスを妨げるものがあります。たとえば、参照整合性（RI）のようなリレーショナル アクセスを制限するように設計された機能は、データ整合性という点では Btrieve アクセスも制限します。

整合性の設定、バウンド データベース、ODBC/SQL セキュリティ、トリガー、参照整合性、およびオーナー ネームなどの機能をトランザクショナル（Btrieve）アプリケーションが使用するデータベースに実装するには、あらかじめこれらの機能について完全に理解している必要があります。ほとんどの場合、両者の長所を生かすデータベース アクセスを得られますが、セキュリティ、参照整合性、およびトリガーはアクセスやオペレーションに制約を設けるため、一部の Btrieve オペレーションは、それらの実装によって制限されたり妨害されたりする場合があります。

整合性の設定、バウンド データベース、セキュリティ、トリガー、または参照整合性を使用している場合には、データまたはファイルへの Btrieve アクセス時、そのデータに設定されている制約に違反すると、Btrieve アクセスが制限されたり完全にアクセスできなくなったりすることがあります。トリガーと RI は主として Btrieve API を介したデータの操作能力を制限します。

セキュリティとオーナー ネームは、適切なアカウント、アクセス権、およびパスワードを持たないデータへのアクセスまたは操作能力を制限する場合があります。これらの機能について考えられるさまざまな組み合わせはたくさんあるので、最も一般的な組み合わせのみを以下に一覧表示します。

表 19　リレーショナル制約と Btrieve アクセス間の相互作用

条件						結果	
DDF の有無	整合性の設定	バウンド データベース	リレーショナル セキュリティの使用	トリガーの使用	RI の使用	Btrieve アクセス可能	SQL/ODBC アクセス可能
No	-	-	-	-	-	Yes	No
Yes	No	No	No	No	-	Yes	Yes
Yes	No	No (1)	No	Yes (2)	-	Yes (2)	Yes
Yes	Yes	No	No	No	No	Yes	Yes
Yes	Yes	No	Yes	No	No	Yes (3)	Yes (3)
Yes	Yes	No (1)	Yes	No	Yes	Yes (4)	Yes (3) (4)
Yes	Yes	No (1)	Yes	Yes (2)	No	Yes (1)	Yes (3)
Yes	Yes	Yes	No	No	No	Yes	Yes
Yes	Yes	Yes	Yes	No	No	Yes (3)	Yes (3)
Yes	Yes	Yes (1)	Yes	No	Yes	Yes (2) (4)	Yes (3) (4)
Yes	Yes	Yes (1)	Yes	Yes (2)	No	Yes (2) (3)	Yes (3)

(1) データベースにバウンド データベースが設定されているかどうかに関係なく、データベース エンジンは、データ ファイルにトリガーある場合、外部キーがある場合、あるいは外部キーで参照される主キーがある場合には、自動的にデータ ファイルをバウンドとしてスタンプします。バウンド データベースやファイルの詳細については、「バウンド データベース」を参照してください。

(2) テーブルにトリガーを追加すると、Btrieve API はトリガーの実行を引き起こすあらゆるオペレーションについて、そのファイルへのアクセスが妨げられます。トリガーは Btrieve インターフェイスを介したデータベース操作には作用しないため、このロックアウト動作によってデータの一貫性が保持されます。詳細については、「バウンド データベースと整合性の設定」を参照してください。

(3) データベースまたはファイルがセキュリティで保護されている場合、ユーザーがそのファイルに対する権限（リレーショナル ユーザー名とパスワード、または有効な Btrieve オーナー ネーム）を持っているのであれば、アクセスは許可されます。セキュリティで保護されたデータベース内にあるがオーナー ネームが設定されていないファイルには、Btrieve ユーザーはアクセスできません。Btrieve オーナー ネームが既に設定されているファイルにリレーショナル セキュリティを最初に設定するとき、Master ユーザーが Btrieve ファイルのオーナー ネームを使ってユーザーにリレーショナル権限を付与する必要があります。詳細については、「Zen セキュリティ」を参照してください。

(4) テーブルに参照整合性制約が含まれており、そのデータベースで整合性の設定が有効になっている場合、制約に違反する Btrieve および SQL 操作は共に行えません。このメカニズムにより、アクセス方法に関係なくデータの整合性が保たれます。

バウンド データベースと整合性の設定

名前付きデータベースに［整合性の設定］属性を指定しない場合、データベース エンジンは参照整合性、トリガー、セキュリティ規則のいずれも強制しません。名前付きデータベースに［整合性の設定］属性を指定すると、データのアクセスに使用する方法とは無関係に、MicroKernel は定義済みセキュリティ、参照整合性（RI）およびトリガーを設定できます。MicroKernel は以下のようにこれらの規則を設定します。

- Btrieve ユーザーはリレーショナル セキュリティの制約を受けません。ファイルにオーナー ネームが設定されている場合、それは有効なままです。ファイルにオーナー ネームがない場合、Btrieve ユーザーはリレーショナル セキュリティ制約に関係なくデータにアクセスできます。Btrieve オペレーションは RI やそれ以外のデータベースに定義されているすべての制約に加え、以下に示すトリガー制限の影響を受けます。
- ファイルに制約が存在する場合、Btrieve アクセスは以下のように許可されます。

表 20　Btrieve アクセスの制限－整合性の設定

ファイルの制約	Btrieve を使ったアクセスで許可されるレベル
RI 制約の定義あり	ユーザーはデータのアクセス、および RI 制約の範囲内でのオペレーションの実行が可能です。
INSERT トリガーの定義あり	読み取り専用、更新、および削除オペレーションが許可されます。
UPDATE トリガーの定義あり	読み取り専用、挿入、および削除オペレーションが許可されます。
DELETE トリガーの定義あり	読み取り専用、更新、および挿入オペレーションが許可されます。

バウンド ファイルに複数の制約が存在する場合、アクセス レベルは最大限の制約または制約の組み合わせに従います。たとえば、ファイルに INSERT トリガーと UPDATE トリガーが定義されている場合、Btrieve ユーザーは読み取り専用および削除アクセスのみが行えます。

「整合性の設定」が直接「バウンド データベース」に関連付けられていません。データベースは整合性の設定なしでバインドされるか、データベースはバインドされずに整合性の設定を設定されます。

バウンド データベース

名前付きデータベースに「バインド」属性を指定した場合、そのデータベースの DDF およびデータ ファイルはほかのデータベースとは関連付けることができません。また、バウンド データ ファイルを、データベース内の複数のテーブル定義と関連付けることもできません。新しいテーブルまたは DDF をバウンド データベースに追加すると、データベース エンジンが自動的に新しいオブジェクトのバインドを行います。この動作により、予測できない動作やデータ整合性を壊すことになる矛盾を防ぎます。たとえば、同じデータベース内の 2 つの異なるテーブル定義に同じデータ ファイルを使用した場合、一方のテーブルに RI 規則を定義し、もう一方には定義しない

でおくことができます。この場合、RI 規則を持たないテーブルに行を挿入することは、もう一方のテーブルの RI 規則に違反することになります。データ ファイルと DDF をバインドすることによりこのような矛盾を防ぎます。

DDF および データ ファイルは、個々にバインドできます。データ ファイルにトリガーがある場合、外部キーがある場合、または外部キーで参照される主キーがある場合、データベース エンジンは自動的にそのデータ ファイルをバインド データ ファイルとして特徴付けます。これらのファイルは別のデータベースと共有できず、複数のテーブル定義に関連付けられることもできません。

データ ファイルがバインドされていることは、そのデータ ファイルへの Btrieve アクセスには直接影響しません。ただし、バインドされたファイルには Btrieve アクセスを制限するほかの制約があることがあります。

関連項目

データベースへの「整合性の設定」および「バウンド データベース」設定の使用法については、「[データベースの新規作成］GUI のリファレンス」を参照してください。

Zen セキュリティ

8

データベース エンジンのためのセキュリティに関する概念と作業

Zen は、MicroKernel エンジンを呼び出す SQL ベース アプリケーションおよび Btrieve アプリケーションに対し、それぞれリレーショナル データベース レベルおよびファイル レベルのセキュリティを提供します。データ アクセスは、オペレーティング システムや Active Directory アカウントを介したユーザー認証によって、もしくは Zen データベース ユーザーとして保護することができます。ユーザー ID が確認されると、そのユーザー権限は、オペレーティング システムまたは Active Directory の権限グループによって、もしくはユーザー / グループ レベルで定義された Zen データベース権によって承認されます。

以下のセクションでは、これらのセキュリティ モデルとそれらモデルの使用方法、および Zen のデータ ファイルの暗号化機能について説明します。

- 「リレーショナル エンジンのセキュリティ モデル」
- 「MicroKernel エンジンのセキュリティ モデル」
- 「MicroKernel エンジンのセキュリティ計画」
- 「MicroKernel エンジン セキュリティのクイック スタート」
- 「ネットワーク上のデータの暗号化」

リレーショナル エンジンのセキュリティ モデル

Zen では、SQL アプリケーションは、次の表に示す認証設定および許可設定を使用できます。

データベース セキュリティ	認証	許可	セキュリティのレベル
無効	• オペレーティング システム • サーバー接続用のユーザー名とパスワード	• 制限なし • オーナー ネーム（省略可能）	なし。SQL 接続でアクセス可能なすべてのテーブル。
ローカル データベース	• Zen ユーザー • オペレーティング システムやドメインとは無関係のユーザー名およびパスワード	• Zen ユーザー権限（グループがない場合） • Zen グループ権限（ユーザー権限がない場合） • PUBLIC の権限 • オーナー ネーム（省略可能）	• データベース • テーブル • テーブル列 • ストアド プロシージャ（V2 形式の場合） • ビュー（V2 形式の場合）
Windows ドメイン	• Active Directory ユーザー • ユーザー名とパスワード • Zen データベース グループ名を使って付けられた名前を持つドメイン グループに割り当てられたユーザー	• Zen グループ権限（必須） • PUBLIC の権限 • オーナー ネーム（省略可能） • 個々のユーザー権限 / 権利なし	ローカル データベースの場合と同じ

新しいデータベースを作成したとき、デフォルトでは、データベース セキュリティは無効になっています。これを有効にするには、セキュリティを有効にしたときに作成された Master ユーザーのパスワードを入力します。使用可能なパスワードの詳細については、「識別子の種類別の制限」を参照してください。

データベースのセキュリティを有効にした後で、さまざまなデータベース活動のために権限を設定するのに必要となるユーザーやグループを作成する必要があります。ローカル データベース セキュリティを設定するには、ユーザーに直接権限を割り当てる方法と、グループ レベルで権限を割り当ててからそのグループにユーザーを割り当てる方法の 2 つがあります。Windows ドメインのセキュリティを設定するには、グループのみを作成します。このグループの名前には、ネットワーク ユーザーがメンバーになっている権限グループと同じ名前を使用します。各データベースのユーザーやグループを管理するには、ZenCC または SQL スクリプトを使用します。

また、データベースに接続しているすべてのユーザーに適用されるデフォルトの PUBLIC グループの権限を設定することもできます。各ユーザーの権限は、ユーザー権限またはグループ権限と PUBLIC グループの権限の組み合わせとなります。新しいデータベースの PUBLIC グループには、権限が設定されておらず、すべての権限がグループ レベルまたは個々のユーザーのレベルで管理されている場合には設定は必要ありません。

これらの概念について、以降のセクションでさらに詳しく説明します。

- 「Master ユーザー」
- 「特殊なグループである PUBLIC」
- 「ユーザーとグループ」
- 「SQL および Zen のセキュリティ」

Master ユーザー

Zen データベースのセキュリティを有効にすると、そのデータベースへのフル アクセス権を持つ Master という名前のユーザーが作成されます。Master ユーザーにはパスワードが必要です。このパスワードは、セキュリティを有効にするときに設定します。各データベースに設定する Master パスワードは互いに無関係のため、異なる値を

設定できます。データベースのセキュリティ設定を変更するには Master パスワードを入力する必要があるため、Master パスワードを忘れないようにしてください。

データベースの Master ユーザーとして認証された後で、ZenCC を使用することで、そのデータベースのユーザーの作成、ユーザーの割り当て先となるグループの作成、グループ レベルとユーザー レベルでのデータへのアクセス権限の管理を行うことができます。また、グループの作成とそのグループへのユーザーの作成を行う SQL ステートメントを実行することもできます。ユーザーは、作成した 1 つのグループだけのメンバーになることができます。すべてのユーザーは各データベースの特殊なグループである PUBLIC のメンバーに自動的になります。

特殊なグループである PUBLIC

すべてのユーザーに同様のアクセス権を付与したい場合は、それらのアクセス権を PUBLIC という名前の特殊なグループに付与することができます。セキュリティを有効にすると、データベース エンジンは自動的に PUBLIC という特殊なグループを作成します。最初、PUBLIC には何のアクセス権も割り当てられていません。PUBLIC 権限を割り当てるか、独自の権限を持つユーザーやグループを作成するまでは、Zen データへのアクセスはブロックされます。

PUBLIC は**すべて**のユーザーにデフォルトの権限を与える特殊なグループです。別の Zen グループにユーザーを割り当てた場合でも、そのユーザーは PUBLIC のメンバーであり続けます。PUBLIC のアクセス権の適用方法をわかりやすくするために 2 つの例を示します。ZenCC で、PUBLIC に CREATE TABLE 権限を割り当てるとします。次に、ZenCC で myuser という名前のユーザーを作成しますが、ユーザーのアクセス権にテーブルを作成するための個別の権限は含めないものとします。データベース エンジンは最初に PUBLIC のアクセス権を調べるため、この場合、テーブル作成の権限は PUBLIC から付与されることになり、myuser はテーブルを作成できます。

逆に言えば、PUBLIC にアクセス権が付与されて**いない**場合は、個々のユーザーまたはグループに付与されているアクセス権が適用されます。たとえば、ZenCC で、PUBLIC に CREATE TABLE 権限を割り当てないとします。ユーザーまたはユーザーが属するグループのアクセス権がテーブルの作成を許可しない限り、どのユーザーもテーブルを作成できません。

ユーザーとグループ

ローカル データベース認証を有効にした後で、そのデータベースのユーザーとグループの権限を管理できます。Windows ドメイン認証の場合には、ユーザー メンバーシップは Active Directory で割り当てられるため、グループ権限のみを管理することができます。どちらの認証の場合でも、ZenCC の Zen エクスプローラーには管理できるユーザーとグループが表示されます。次の表は、ユーザーとグループに適用される一般的な規則を示します。

ユーザーとグループに関する規則	ローカル データベース	Windows ドメイン
ユーザーは、グループのメンバーになる必要はなく、個人としての権限設定を持つことができる	X	**適用外**
グループ内のすべてのユーザーは、そのグループに定義された権限を持つ	X	X
ユーザーは、グループのメンバーになる場合には、個人としての権限を持たない	X	X
ユーザーの権限は PUBLIC グループの権限と結合される	X	**適用外**
グループの権限は PUBLIC グループの権限と結合される	X	X
ユーザーは、一度に 1 つのグループ（PUBLIC グループはカウントしない）だけのメンバーになることができる	X	X
グループは別のグループのメンバーになることができない	X	X

詳細については、『*Zen User's Guide*』の「ユーザーとグループの作業」を参照してください。

SQL および Zen のセキュリティ

SQL スクリプトを使ってリレーショナル エンジンのセキュリティを管理する場合は、『*SQL Engine Reference*』の以下のセクションに役立つ情報が記載されているのでご覧ください。

- 「ビューおよびストアド プロシージャに対する権限」
- 「ALTER GROUP」
- 「ALTER USER」
- 「CREATE GROUP」
- 「CREATE USER」
- 「DROP GROUP」
- 「DROP USER」
- 「GRANT」
- 「REVOKE」
- 「SET PASSWORD」
- 「SET SECURITY」
- 「psp_groups」
- 「psp_procedure_rights」
- 「psp_table_rights」
- 「psp_view_rights」
- 「psp_users」

複数のデータベースのデータにアクセスする

1 つの接続内の複数のデータベースが同じマシン上に存在する場合には、SQL アプリケーションはそれらのデータベース内のデータにアクセスできます。ただし、一度にログインできるデータベースは 1 つのため、ログインしていない別のデータベースにアクセスできるかどうかは両方のデータベースのセキュリティ設定によります。

表 21　セキュリティ設定に基づくデータベースのアクセス権

ログインしているデータベースのセキュリティ	もう 1 つのデータベースのセキュリティ	もう 1 つのデータベースへのアクセス
有効	なし	すべての権限 / 権利を使用したアクセスが可能です。
有効	有効 データベースは、ログインしているデータベースと**同じ**ユーザー名とパスワードを使用します。	もう 1 つのデータベースに設定されている権限 / 権利を使用したアクセスが可能です。
有効	有効 データベースは、ログインしているデータベースと**異なる**ユーザー名とパスワードを使用します。	アクセスは拒否されます。
なし	有効	アクセスは拒否されます。

MicroKernel エンジンのセキュリティ モデル

Zen では、Btrieve アプリケーションは、次の表に示す認証設定および許可設定を使用できます。

Btrieve セキュリティ	認証	許可
クラシック	• オペレーティング システム • ユーザー名とパスワード	• ファイル システムに対する権限 • オーナー ネーム（省略可能）
ローカル データベース認証によってセキュリティ保護されたデータベース	• Zen ユーザー • オペレーティング システムやドメインとは無関係のユーザー名およびパスワード	• Zen ユーザー権限 / 権利 • Zen グループ権限（省略可能） • PUBLIC 権限（省略可能） • オーナー ネーム（省略可能）
Windows ドメイン認証によってセキュリティ保護されたデータベース	• Active Directory ユーザー • ユーザー名とパスワード • Zen データベース グループと同じ名前を持つドメイン権限グループに割り当てられたユーザー	• Zen グループ権限（必須） • PUBLIC 権限（省略可能） • オーナー ネーム（省略可能） • 個々のユーザー権限 / 権利なし
混合	• オペレーティング システムまたはドメイン • ユーザー名とパスワード	• 個々のユーザー権限 / 権利（ローカル データベース セキュリティを使用する場合） • グループ権限（Windows ドメイン セキュリティを使用する場合） • PUBLIC 権限（省略可能） • オーナー ネーム（省略可能）

これらの設定について、以降のセクションでさらに詳しく説明します。

- 「Btrieve のクラシック セキュリティ」
- 「Btrieve の混合セキュリティ」
- 「Btrieve ファイルのデータベース セキュリティ」
- 「Btrieve のクラシック セキュリティおよび混合セキュリティに関する注意事項」
- 「Btrieve の混合セキュリティおよびデータベース セキュリティに関する注意事項」
- 「Btrieve の混合セキュリティやデータベース セキュリティの設定」
- 「オーナー ネーム」

これらのセクションは、MicroKernel エンジンを呼び出すアプリケーションにのみ適用されます。「**資格情報**」、「**ログイン資格情報**」、または「**ユーザー資格情報**」とは、有効なユーザー名とパスワードのことです。

MicroKernel エンジンに対する Zen データベースの認証および許可は、オペレーティング システムに対して依存させるか、または独立させるかを設定できます。Btrieve ユーザーにデータ ファイルへのオペレーティング システム アクセスが許可されていなくても、そのユーザーにデータベースへのアクセスを許可できます。

Btrieve のクラシック セキュリティ

クラシック セキュリティは、本データベースのすべてのリリースで提供されている Btrieve セキュリティ モデルです。Btrieve ユーザーの場合、認証はオペレーティング システムによって実行され、データ アクセス権限は当該ユーザーのファイル システム権限によって決定されます。ユーザーには Btrieve 経由でファイルにアクセスするための同様の権限があります。オペレーティング システムによって制御される別の種類のファイルに対してもその権限があります。

Btrieve オーナー ネームは、個々のファイルに対するアクセスを制限することによって、別のセキュリティ レイヤーを追加できます。詳細については、「オーナー ネーム」を参照してください。

Btrieve のクラシック セキュリティの設定

クラシック セキュリティでは、アプリケーションのユーザーおよびアクセス権限の設定は、オペレーティング システムのユーザーを作成し、ファイルおよびフォルダーへの権限を割り当てるだけで行うことができます。これら以外の操作を行う必要はありません。

Btrieve の混合セキュリティ

Btrieve の混合セキュリティは、ユーザーにオペレーティング システム ファイルへの権限を付与せずに、Btrieve データ ファイルへのアクセス権を提供します。このオプションを使用できる環境は、ファイルを指定する URI や API ログインを使用するように Btrieve アプリケーションを変更できないようにセキュリティを強化している環境です。

混合セキュリティ モデルでは、Btrieve の Open 要求が発生すると、データベース エンジンは DefaultDB という名前の特別なデータベースに登録されているユーザーのリストに照らして、オペレーティング システム ユーザーを認証します。オペレーティング システムがユーザーを認識すると、データベース エンジンはデータベースの権限テーブルを使って、開こうとしているファイルへの当該ユーザーのアクセス権を決定します。このため、ユーザーの権限が、そのユーザーが使用しようとしているデータベース内に定義されている必要があります。オペレーティング システムの権限に関係なく、データベース エンジンはその権限を適用します。

Btrieve アプリケーションのデータベース許可は、リレーショナル エンジン セキュリティ モデルを拡張することによって提供されているため、Btrieve アプリケーションでも使用できます。ユーザーとグループを作成、定義して権限を設定する機能は、ZenCC によって、また、GRANT、REVOKE、ALTER GROUP、CREATE USER、ALTER USER、DROP USER などの SQL ステートメントによっても提供されます。

混合セキュリティ モデルにおいて、ローカル データベース認証と Windows ドメイン認証は多少異なります。混合セキュリティ モデルでは、DefaultDB データベースに定義されているユーザー名はすべて、オペレーティング システムに定義されているユーザー名と一致していなければなりません。データベース エンジンは、Btrieve ファイルの作成中やオープン操作中は、入力されたユーザー名とパスワードを認証用にオペレーティング システムに渡すだけです。オペレーティング システムが資格情報を認証したら、データベースはユーザー個人の権限またはそのユーザーが割り当てられているグループの権限を使用します。個々のユーザーまたはグループの権限を定義する代わりに、PUBLIC グループを使えば、それらの権限を一度に定義することができます。

Windows ドメイン認証との混合セキュリティでは、Active Directory 内のグループ名はデータベースに定義されているグループと同じである必要がありますが、ユーザー メンバーシップは Active Directory 内でのみ処理されます。データベース エンジンは、Btrieve ファイルの作成中やオープン操作中は、、ユーザーが入力したユーザー名とパスワードを認証用にネットワークに渡します。ネットワークが資格情報を認証したら、データベースはユーザーに割り当てられているグループを使って権限を決定します。データベースにユーザーを追加する必要はありません。事実、ZenCC には［ユーザー］ノードが表示されなくなっています。様々なグループの権限を定義する代わりに、PUBLIC グループを使えば、それらの権限を一度に定義することができます。

混合セキュリティの設定手順については、「Btrieve の混合セキュリティやデータベース セキュリティの設定」を参照してください。

Btrieve のクラシック セキュリティおよび混合セキュリティに関する注意事項

ワークグループ エンジンはオペレーティング システム認証を実行しないため、ワークグループ エンジンを使用している場合の、クラシックおよび混合セキュリティ ポリシーの動作は同じです。ワークグループ エンジンを使って Btrieve データベースをセキュリティで保護したい場合は、データベース セキュリティ ポリシーを選択し、必要なユーザーやグループをすべて設定する必要があります。

Btrieve ファイルのデータベース セキュリティ

Btrieve ファイルのデータベース セキュリティでは、アプリケーションが Btrieve ログイン オペレーションを実行するか、Btrieve URI を使って Open オペレーションのファイルを指定する必要があります。どちらの場合も、デー

タベースはログインまたは URI の中から参照されます。このデータベースは、指定されたユーザーに権限を付与します。Btrieve データベース セキュリティでは、Zen データベースは、ローカルの Zen データベースに定義されたユーザーのリストと照合するか、あるいはユーザーが Zen データベース内のグループと同じ名前を持つ権限グループのメンバーになっている Windows Active Directory に定義された、ネットワーク ログインのリストと照合することで、ユーザーを認証し、それらのユーザーに対して Btrieve データ ファイルへのアクセスを許可します。後記する 1 つの例外を除いて、ユーザーがデータに接続しアクセスする権限は、そのユーザーが必要なアプリケーションを実行するシステムに正常にログインできる限り、ファイル システム権限とは関係ありません。データベースのユーザーとグループを定義して権限を設定する機能は、ZenCC によって、また GRANT や REVOKE などの SQL ステートメントによって提供されます。

メモ 新しいデータベースを作成する場合は、ユーザーはオペレーティング システムで管理者権限を持っている必要があります。

Btrieve の混合セキュリティおよびデータベース セキュリティに関する注意事項

混合またはデータベース セキュリティ モデルは、当該データベースに属するものとして定義されているディレクトリに存在する Btrieve データ ファイルでのみ使用できます。このデータベースには、「デフォルトのデータベースと現在のデータベース」に記載されているデフォルト データベース DefaultDB も含まれます。データベースと関連付けられていないディレクトリにあるデータ ファイルにはアクセスできません。

これらのセキュリティ モデルの主な利点の 1 つは、ユーザーのデータ ファイルへの直接アクセスを制限する一方で、データベース エンジンを介したデータ アクセスにはフル アクセスを許可できることです。対照的にクラシック モデルでは、データ ファイルへのレコードの追加が許可されているユーザーは、必ずオペレーティング システムからもデータ ファイルのコピー、削除ができなければなりません。

Btrieve の混合セキュリティやデータベース セキュリティの設定

混合セキュリティまたはデータベース セキュリティに移行するには、多数の選択を行った上、慎重に計画を立てる必要があります。既に確立している環境では、お使いの製品環境が混乱しないように、Btrieve ファイルをデータベースにまとめる方法を考え、移行の予定を立てる必要があるかもしれません。

デフォルトのクラシック セキュリティから混合またはデータベース セキュリティに移行する手順については、『*Zen User's Guide*』の「セキュリティの作業」を参照してください。

オーナー ネーム

オーナー ネームは、Btrieve データ ファイルへのアクセスを制限するために使用するバイト文字列です。オーナー ネームは、個々のファイルにアクセスできるようにするパスワードと考えることができます。さらに、Zen がファイルを暗号化すると、オーナ ネームはプライベート暗号化キーとして機能します。オーナー ネームはファイル ヘッダーに存在します。このオーナー ネームは、ファイルが暗号化されているかどうかにかかわらず、常に暗号化されます。一般のパスワードと同様に、オーナー ネームでは大文字と小文字が区別されます。

サポートされているすべての Zen データベース エンジンのリリースでは、ファイルに対して 1 ～ 8 文字の ASCII 文字の「短い」オーナー ネームを設定する機能が提供されます。アクセス制限に加え、短いオーナー ネームは、Function Executor の［暗号化ファイル］設定や Maintenance ツールの［ファイルのデータを暗号化する］設定を選択した場合に、独自の暗号化を適用するために使用されます。

Zen 10.10 から、1 ～ 24 文字の ASCII 文字の「長い」オーナー ネームが導入されました。これにより、ファイル セキュリティが強化され、より強力な 128 ビット暗号化を使用できるようになりました。

Zen 13.30 では、AES-192 暗号化が追加され、16 進数文字列の長いオーナー ネームがサポートされました。

Zen 14.00 では、長いオーナー ネームの最大 ASCII 文字数が 32 に延ばされ、それに応じて 16 進数の制限も拡張され、AES-256 暗号化が追加されました。

長いオーナー ネームとファイル暗号化の可能性を次の表に要約します。

Zen バージョン	長いオーナー ネーム	ファイルの暗号化
14.00	**13.0 形式** 1 ～ 32 文字の ASCII 文字 32 ～ 64 文字の 16 進数＋ 0x または 0X のプレフィックス **9.5 形式** 1 ～ 24 文字の ASCII 文字 32 ～ 48 文字の 16 進数＋ 0x または 0X のプレフィックス	13.0 形式で、長いオーナー ネームが 25 ～ 32 文字の ASCII 文字であるか、50 ～ 64 文字の 16 進数であるファイルは、AES-256 暗号化を使用します。 13.0 形式で、長いオーナー ネームが 1 ～ 24 文字の ASCII 文字であるか、32 ～ 48 文字の 16 進数であるファイルは、AES-192 暗号化を使用します。 9.5 形式で長いオーナー ネームを持つファイルは、128 ビット暗号化を使用します。
13.30	**13.0 形式** 1 ～ 24 文字の ASCII 文字 32 ～ 48 文字の 16 進数＋ 0x または 0X のプレフィックス **9.5 形式** 1 ～ 24 文字の ASCII 文字 32 ～ 48 文字の 16 進数＋ 0x または 0X のプレフィックス	13.0 形式で長いオーナー ネームを持つファイルは、AES-192 暗号化を使用します。 9.5 形式で長いオーナー ネームを持つファイルは、128 ビット暗号化を使用します。
10.10 ～ 13.20	**9.5 形式** 1 ～ 24 文字の ASCII 文字	9.5 形式で長いオーナー ネームを持つファイルは、128 ビット暗号化を使用します。

16 進数の長いオーナー ネームは以下の要件を満たす必要があります。

- オーナー ネーム文字列は、先頭に文字 0x または 0X が付いている。
- オーナー ネーム文字列は、16 進数の文字 0123456789abcdefABCDEF のみを使用し、スペースが入っていない。
- ヌル文字（00 としてエンコードされる）を使用していない。
- 13.0 形式ファイルの場合の桁数は、32 ～ 64 文字とプレフィックス 0x または 0X を合わせて偶数である必要がある。9.5 形式ファイルの場合の桁数は、32 ～ 48 文字とプレフィックス 0x または 0X を合わせて偶数である必要がある。ここで留意する点は、プレフィックスは 16 進値を示すためだけに使用され、暗号化アルゴリズムでは使用されないということです。

長いオーナー ネームを持つファイルには、以下の制限が適用されます。

- Zen v14 以降におけるオーナー ネーム文字列の暗号化はより強力になっているため、Zen v14 で長いオーナー ネームを割り当てられた 13.0 形式のファイルは、Zen v14 以降でのみ読み取り可能になります。これは、そのファイル自体が暗号化されているかどうかにかかわらず適用されます。9.5 形式のファイルは強力な暗号化を持たないため、影響を受けません。
- 10.10 バージョンより前の Zen は長いオーナー ネームをサポートしていないため、長いオーナー ネームを持つファイルを読み取ることができません。
- ファイルを 9.5 より前のファイル形式にリビルドするには、まず、そのオーナー ネームを削除しておく必要があります。

メモ オーナー ネームは、ネットワーク、システム、またはデータベース レベルで認証されるユーザー名とは関係ありません。たとえば、ファイル オーナー ネームの Master はデフォルト Master ユーザーとは異なります。

以下のトピックではより詳しい説明を提供します。

- 「オーナー ネームを選択および設定する」
- 「オーナー ネームおよび SQL アクセス」
- 「オーナー ネームと暗号化」

■ 「オーナー ネームの例」

オーナー ネームを選択および設定する

Maintenance および Function Executor ツールで、ファイルのオーナー ネームを設定またはクリアすることができます。SQL スクリプトを使用して、テーブルのオーナー ネームを設定または管理することはできませんが、SET OWNER および GRANT ステートメントを使用すれば必要に応じてオーナーネームを提供できます。

ユーザーは、オーナー ネームが割り当てられたときに選択したオプション（次の表を参照）に応じてファイルにアクセスすることができます。

オプション	説明
読み取り専用	オーナー ネームを指定しなくても、ユーザーはデータ ファイルの変更を行わないデータ アクセス操作を行うことができます。
読み取り専用、暗号化	オーナー ネームを指定しなくても、ユーザーはデータ ファイルの変更を行わないデータ アクセス操作を行うことができます。このオプションを設定すると、データベース エンジンはオーナー ネームをキーに使用してファイルの全レコードを暗号化します。後から追加されるレコードも暗号化されます。
ノーマル	オーナー ネームを指定しないと、ユーザーはファイル アクセス操作を一切行うことができません。
暗号化のみ	オーナー ネームを指定しないと、ユーザーはファイル アクセス操作を一切行うことができません。このオプションを設定すると、データベース エンジンはオーナー ネームをキーに使用してファイルの全レコードを暗号化します。後から追加されるレコードも暗号化されます。

オーナー ネームおよび SQL アクセス

SQL データベースでテーブルとして機能するファイルに Btrieve オーナー ネームが設定されている場合に、そのテーブルにアクセスするには、そのデータベースでセキュリティが有効になっているかどうかによって、2 つある方法のうち、いずれかを使用します。セキュリティが無効な場合は、SET OWNER ステートメントをその他のテーブル アクセス試行より前に実行することにより、オーナー ネームを取得できます。セキュリティが有効な場合は、データベースの Master ユーザーは、そのテーブルに対するアクセス許可を任意のユーザー（Master ユーザー自身を含む）に発行する場合、GRANT ステートメントでオーナー ネームを提供する必要があります。

オペレーティング システムでオーナー ネームが含まれるファイルが読み取り専用の場合、Master ユーザーはオーナー ネームを使って自分自身に SELECT 権を特に付与しなくても、自動的に SELECT 権限を持ちます。この場合、他のユーザーには、この自動的に付与されるアクセス権はありませんが、Master ユーザーはオーナー ネームを提供しなくても SELECT 権限を付与することはできます。

オーナー ネームと暗号化

ファイルに暗号化を指定したオーナー ネームを最初に設定したときに、データベース エンジンはファイル全体を直ちに暗号化します。ファイルのサイズが大きくなるほど、暗号化には時間がかかります。

暗号化されたファイルに対するデータ アクセス操作は、暗号化されていないファイルに比べて時間がかかります。データベース エンジンは、ディスクからページを読み取る際にそのページを復号し、それをディスクに書き戻す前に再度暗号化する必要があります。

注意 ファイルのオーナー ネームを忘れたりなくしたりしないようにしてください。ファイル自体が暗号化されていなければ、Btrieve API を使ってそのファイルを読み取ることはできますが、オーナー ネームを指定せずに、ファイルに書き込むことはできません。ファイルのオーナー ネームは、ファイルの内容を調べても

見つかりません。これは、ファイル データが暗号化されるかどうかにかかわらず、オーナー ネーム文字列が、その割り当て対象のファイルのヘッダーで常に暗号化されるからです。

オーナー ネームの例

Btrieve オーナー ネームを使用してファイルへのアクセスを許可する例については、『*SQL Engine Reference*』の「GRANT」を参照してください。

MicroKernel エンジンのセキュリティ計画

新しいデータベースではすべて、［Btrieve セキュリティ］設定はデフォルトで " クラシック " になります。つまり、データベース エンジンはオペレーティング システムを使って認証を行い、フォルダーとファイルの権限を使ってアクセスの許可を行います。オペレーティング システム認証でデータ ファイルへのアクセスが許可されているユーザーは、Btrieve オーナー ネームを使ってアクセスが制限されていない限り、ファイルに対する権限と同レベルの権限をファイルに含まれているデータ レコードに対しても持ちます。

以下のセクションでは、デフォルト データベースやアクセスを許可されたユーザーなど、Btrieve セキュリティ ポリシーの要素を設定する手順について説明します。

- 「使用可能なオプション」
- 「ポリシーの選択」
- 「セキュリティを設定するための準備」
- 「処理の概要」

使用可能なオプション

使用できるセキュリティ オプションは 3 つあります。これらのオプションのうち最適なオプションを選択できるように、各オプションの機能を次の表に示します。暗号化はどの設定でも任意です。

表 22　セキュリティ設定の機能比較

機能	クラシック	混合	データベース
ローカル データベース セキュリティを使用する場合は、管理者がユーザーごとにオペレーティング システム（OS）とデータベースのユーザー アカウントを別々に設定する必要がある。		X	X
データベースのユーザー アカウントは OS のユーザー アカウントから直接派生する。これは、Windows ドメイン認証によるデータベース セキュリティが使用される場合には常に該当する。	X	X	
データ アクセス権はファイル システム権には関連付けられていない。管理者は、データベースのデータ アクセス権限を個々のユーザーまたはグループに割り当てる必要がある。		X	X
ユーザーのデータ アクセス権は、OS ユーザーのファイル システム権から直接派生する。	X		
Zen をベースとする Windows アプリケーションから、データベースのユーザー名とパスワードを入力する自動ログイン ダイアログを表示できる。	X[1]	X	X
データベースは OS にログインできたユーザーを有効なデータベース ユーザーとして受け入れる。	X	X	X[2]
ユーザーは、コンピューターにログインするのとは別にデータベースにログインする必要がある。			X[3]
[1] ログイン ダイアログは、リクエスターがオペレーティング システムを介して ID を決定できない場合に現れます。 [2] Windows ドメイン認証を使ってセキュリティ保護されたデータベースの場合。 [3] ローカル データベース認証を使ってセキュリティ保護されたデータベースの場合。			

Btrieve **データベース** セキュリティおよびリレーショナルのローカル データベース セキュリティの下では、データベースのユーザー アカウントはオペレーティング システムのユーザー アカウントとはまったく無関係です。

これとは対照的に、**クラシック** セキュリティでは、コンピューターへのログインに成功したユーザーは、データを含んでいるファイルに対してユーザーに割り当てられているファイル システム権限のレベルに従った、そのデータベース コンテンツへのアクセス権を持ちます。

最後に、Btrieve **混合**セキュリティ ポリシーは、ほかの 2 つのポリシーの要素を持っています。この方式では、ユーザーはオペレーティング システムのユーザー名とパスワードを使ってログインしますが、データに対するユーザーのアクセス権はセキュリティ保護されたデータベースに設定されているユーザーまたはグループの権限によって決定されます。

ポリシーの選択

以下のセクションでは、複数のセキュリティ ポリシーの中から 1 つを選び出す上での主な判断材料のいくつかを述べます。

クラシックを選択する理由

- ユーザーがデータ ファイルへのファイル システム アクセス権を持っていることを安心して受け入れられる。たとえば、データ ファイルからレコードを削除する権限を持っているユーザーは、オペレーティング システムからそのファイル自体を削除することもできます。
- 管理上の面倒な作業は最小限にしたい。各ユーザーの OS ユーザー アカウントと、少なくとも 1 つのデータベース アカウントの両方を設定することはしたくない。
- 各ユーザーのファイル システム権限によって異なる、さまざまなデータ アクセス権を持つ必要がない。
- ユーザーにデータベースのログインを個別に行わせたくない。

混合を選択する理由

- Btrieve ログインや、ファイルを開くための Btrieve URI パスを使用するように変更できない既存の Btrieve アプリケーションがある。
- ユーザーにデータベースのログインを個別に行わせたくない。
- 有効なデータベース ユーザーには、オペレーティング システムにおけるデータ ファイルへのいかなる権限も持たせないようにしたい。たとえば、データベースに対するあらゆる権限を持っているユーザーに、オペレーティング システムからデータ ファイルを削除する権限を持たせないようにすることができます。
- ローカル データベース セキュリティを使用しており、OS のユーザー アカウントと同じユーザー名のデータベース ユーザー アカウントを作成するつもりである。また、データベース ユーザーまたはデータベース ユーザーが所属する各グループに権限を割り当てるつもりである。選択すれば、特殊グループの PUBLIC から権限を継承することによって、すべてのユーザーが同レベルの権限を持つことができます。
- Windows ドメイン セキュリティを使用しており、データベースに定義されている Zen グループと同じ名前の権限グループを Active Directory 内に作成するつもりである。また、各 Zen グループに権限を割り当てるつもりである。Active Directory では、権限は割り当てられず、ユーザーのグループ メンバーシップだけが割り当てられます。

データベースを選択する理由

- Btrieve ログイン オペレーションや Btrieve URI パス形式を使用する Btrieve アプリケーションがある。
- ユーザーにデータベースのログインを個別に行わせたい。これは、オペレーティング システムへのログイン後、ユーザーはもう一度データベースにログインする必要があるということです。この動作は、権限を与えられたコンピューター ユーザーの一部にはデータベースへのアクセスが許可されているが、一部には許可されていない場合に有用です。
- 有効なデータベース ユーザーであっても、オペレーティング システムにおけるデータ ファイルへの権限を持たせないようにしたい。たとえば、データベースに対するあらゆる権限を持っているユーザーに、オペレーティング システムからデータ ファイルを削除する権限を持たせないようにすることができます。これは、**混合**セキュリティ ポリシーを使っても実現できます。

- データベースのユーザー アカウントには、オペレーティング システムのアカウントとは異なる名前を使いたい。たとえば、オペレーティング システムのユーザー jsmith が john としてデータベースにログインすることを要求されるかもしれません。
- ユーザーとその権限は、サーバーやマシンでなくデータベースの下にある。これにより、データベースのユーザーやユーザーの権限を再作成しなくても、データベースをあるマシンから別のマシンへ移動することができます。

セキュリティを設定するための準備

MicroKernel エンジンのセキュリティ設定は 1 つの簡単な処理です。しかし、この処理には十分な柔軟性があるため、いくつかの準備を必要とします。このセクションでは、Btrieve セキュリティの設定を開始する前に知っておくべき情報について説明します。

Btrieve アプリケーションでデータにアクセスする方法

アプリケーションが Btrieve ログイン オペレーションを使用したり URI でファイルを開いたりするように変更されているか、変更可能な場合は、データベース セキュリティまたは Btrieve の混合セキュリティを選択できます。どちらを選択した場合でも、セキュリティで保護するデータベースは ZenCC に登録されます。アプリケーションで Btrieve ログインや URI を使用しない場合にアクセス権を管理するには、混合セキュリティを選択して、DefaultDB データベースを設定します。

データベースの数

混合またはデータベース セキュリティの場合は、すべてのユーザーに同レベルの権限を割り当てるか、もしくはデータベースごとに定義済みユーザーのセットを作成する必要があります。

まったく関連のないデータの本体を含んでいる Btrieve データ ファイルが 2 つ以上ある場合は、別個のデータベースを 2 つ以上設定し、それぞれに固有の許可されたユーザーのセットを持ちたいと思われるかもしれません。しかし、一般的に言えば、定義済みユーザーのセットを複数作成して管理する必要がないよう、別々にするデータベースの数は最小限にしたいでしょう。ほとんどの場合は、1 つのデータベースで十分です。データベースにおけるユーザー権限では、各ユーザーのデータベースへのアクセスを規制できるため、ある特定のユーザーのアクセスを制限するためだけに別のデータベースを作成する必要はありません。

1 つのデータベースのみが必要であると判断した場合は、既存のデータベース DefaultDB を Btrieve ファイルと関連付けられているデータベースとして使用することができます。そうではなく、独自の名前を付けたデータベースを設定することもできます。

データ ファイルの場所

名前付きデータベースの**データ ディレクトリ**に、データ ファイルが格納されているディレクトリを指定することによって、Btrieve データ ファイルをデータベースと関連付けます。そのため、データベースと関連付けるすべてのデータ ファイルが格納されているディレクトリを知っておく必要があります。データ ファイルがすべて特定のディレクトリのサブディレクトリ ツリーに存在する場合は、トップレベルのディレクトリのパス名さえわかっていれば十分です。ハード ディスク ドライブ上の**すべての**データ ファイルを含めたい場合は、"C:¥" を使用することもできます。

ユーザー名の使用方法

ローカル データベース認証との混合セキュリティを使用する予定ならば、すべてのユーザーに同一の権限を割り当てるか、もしくは異なる権限を持つユーザーのユーザー アカウントを設定する必要があります。個々のユーザーを設定するつもりならば、データベース ユーザー名にしたいオペレーティング システム ユーザー名の一覧が必要です。データベース ユーザー名は、オペレーティング システム ユーザー名と完全に一致している必要があります。

Windows ドメイン認証との混合セキュリティを使用する予定ならば、すべてのユーザーに PUBLIC グループの同一権限を割り当てるか、もしくは異なる権限を持つ複数の Zen ユーザー グループを作成する必要があります。ど

ちらの場合も、ネットワーク グループ名の一覧が必要です。このネットワーク グループ名をデータベース グループ名としてコピーします。データベース グループ名はネットワーク グループ名と一致している必要があるためです。

セキュリティ ポリシーとは？

セキュリティを設定する前に、使用する予定のポリシーがどういうものであるかを知っておく必要があります。セットアップ処理はポリシーによって若干異なります。ポリシーを選択する上での考慮点は、「ポリシーの選択」に記載されています。

処理の概要

データベースのセキュリティを設定する手順の概要を以下に示します。手順の詳細については、「MicroKernel エンジン セキュリティのクイック スタート」で説明しています。

1 準備をします。「セキュリティを設定するための準備」の説明に従って、必要な情報を収集し、開始するために必要な決定を行います。データベースの数はいくつにしますか？ Btrieve ファイルが保存されている場所は？ユーザー名は何ですか？どのセキュリティ ポリシーを使用しますか？

2 Btrieve ファイルとともに使用するデータベースを選択し、そのデータベースにデータ ファイルの場所を示すデータ ディレクトリを設定します。この手順は、混合またはデータベース セキュリティでのみ必要となります。

この手順の詳細については、『*Zen User's Guide*』の「定義済みの DefaultDB も含め、既存のデータベースを Zen ファイルと使用するには」を参照してください。

3 セキュリティを有効（オン）にします。

この手順の詳細については、『*Zen User's Guide*』の 「Zen エクスプローラーを使用してセキュリティを有効にするには」を参照してください。

4 ユーザー（および必要に応じてグループ）を作成し、SQL ステートメントまたは ZenCC を使ってユーザー（およびグループ）に対して権限を設定します。この手順は、混合セキュリティまたはデータベース セキュリティでのみ必要となります。

ユーザーのアクセス権を付与する最も簡単な方法については、『*Zen User's Guide*』の「Zen エクスプローラーを使用してすべてのユーザーに権限を割り当てるには」を参照してください。

5 データベースの Btrieve セキュリティに［混合］または［データベース］を設定します。

この手順の詳細については、『*Zen User's Guide*』の「データベースのセキュリティ ポリシーを設定または変更するには」を参照してください。

6 オペレーティング システムでデータ ファイルをセキュリティ保護します。混合またはデータベース セキュリティの場合、これでユーザーは、オペレーティング システムにおけるデータ ファイルへのアクセス権を何も持っていなくても、データにアクセスできるようになります。ファイルへのセキュリティ保護されたアクセスに関する情報は、オペレーティング システムのドキュメントを参照してください。

MicroKernel エンジン セキュリティの作業の概要

次の表は、異なるセキュリティ モデルを使用するために必要となる作業の基本レベルを示しています。セキュリティ モデルを実装するために必要な作業については、『*Zen User's Guide*』の「セキュリティの作業」を参照してください。

セキュリティ モデル	認証 / 許可	動作および高レベル セットアップ作業の概要
クラシック	オペレーティング システム/オペレーティング システム	• ユーザーにすべてのデータベース ファイルへのファイル アクセス権限を付与します。 • よりアクセスを制限するために、Btrieve ファイルにオーナー ネームを追加します（任意）。
混合	オペレーティング システム / データベース	• オペレーティング システムでユーザーを作成します。このユーザー名とパスワードと対照することで、ユーザーが認証されます。 • ZenCC を使って、データベース内の複数のドメインに同じ名前のユーザーを作成します。認証は OS によって行われますが、権限はデータベースに格納されるため、OS のユーザーやドメイン グループはデータベース内のユーザーやグループと一致する必要があります。 • ZenCC または SQL ステートメントを使って、ユーザーやグループの権限を定義します。もう 1 つの方法として、PUBLIC グループに権限のセットを定義します。認証されたすべてのユーザーは、PUBLIC と同様の権限を持ちます。どのユーザーも、PUBLIC に与えられた権限より低い権限を持つことはありません。 • ワークグループ エンジンでは、このセキュリティ モデルはクラシック セキュリティ モデルと同様に動作します。
データベース	データベース/データベース	• オペレーティング システムのユーザー名とパスワードは、Zen データベース セキュリティとは関係ありません。 • ZenCC または SQL ステートメントを使って、ユーザーやグループを定義します。 • ZenCC または SQL ステートメントを使って、データベース権限を定義します。

MicroKernel エンジン セキュリティのクイック スタート

このセクションでは、オペレーティング システムで Btrieve データ ファイルをセキュリティ保護する一方、データベース ユーザーには引き続きデータへのアクセスを許可するための、最も簡単な方法についての手順を説明します。

この手続きが完了したら、アプリケーションを介してデータにアクセスするためのデータベース ユーザー権利に影響を与えることなく、データ ファイルに対するオペレーティング システム ユーザー権利を取り消すことができます。

メモ データベース エンジンがインストールされているコンピューターに、管理者権限を持つオペレーティング システム ユーザーとして、または Zen_Admin セキュリティ グループのメンバーであるユーザーとしてログインする必要があります。

1 Zen Control Center（ZenCC）を起動します。ZenCC の起動方法については、『*Zen User's Guide*』の「Windows での ZenCC の起動」を参照してください。

2 混合セキュリティを使用する場合、以下の手順で使用するデータベースは DefaultDB になります。データベース セキュリティを使用する場合は、お使いのデータベースが ZenCC に登録されていることを確認してください。データベース エンジンの登録方法については、『*Zen User's Guide*』の「リモート サーバー エンジンを登録するには」を参照してください。

3 使用するデータベースが DefaultDB とアプリケーションのデータベースのどちらである場合も、データベースのノードの左にある展開アイコンをクリックします。

4 ZenCC で、データベース DefaultDB を右クリックしてから［**プロパティ**］をクリックします。

5 **ディレクトリ** ノードを選択し、［**新規**］ボタンをクリックします。

6 Btrieve ファイルのパスを入力したら［OK］をクリックし、その後［**適用**］をクリックします。

ファイルが複数のディレクトリに散在している場合は、ファイルすべてに共通の上位ディレクトリを指定します。必要であればルート レベルを指定することもできますが、そうすると、ルート レベルとその下位ディレクトリにあるすべての Btrieve ファイルが DefaultDB に含まれてしまいます。たとえば、Windows の場合のルート レベルは C:¥ となります。『*Zen User's Guide*』の「定義済みの DefaultDB も含め、既存のデータベースを Zen ファイルと使用するには」を参照してください。

すべてのディレクトリを入力する必要はありません。データベースに含めたい Btrieve ファイルすべてに共通する最下位のディレクトリのみ入力します。

7 このデータベースのセキュリティを有効にします。それには、［プロパティ］のツリーで［セキュリティ］ノードをクリックします。

8 ［Btrieve セキュリティ］タブをクリックし、" 混合セキュリティ " または " データベース セキュリティ " を選択します。

9 ［データベース セキュリティ］タブをクリックし、" ローカル データベース認証 " または "Windows ドメイン認証 " オプションを選択します。

10 Master ユーザーに使用するパスワードを入力します。指示に従って、2 回入力します。

これでセキュリティは有効になりました。しかし、ユーザーは現在のところ以前と同様のアクセス権を持っているため、デフォルトでは OS のユーザー権利に基づいてアクセスが行われます。次の段階では、この状況に対処します。

パスワードは最大 8 バイトに制限されているので注意してください。パスワードには、セミコロン（;）と疑問符（?）以外のあらゆる表示可能な文字を使用できます。

11 ［OK］をクリックして、［プロパティ］ダイアログを閉じます。

12 お使いのデータベースの下の［グループ］を展開（ノードの左にある展開アイコンをクリック）し、グループ PUBLIC を右クリックします。

13 ［**プロパティー**］をクリックした後、ツリーで［**権限**］をクリックします。

14 ［データベース］タブをクリックします。

15 必要な権限をクリックします。

たとえば、すべての認証ユーザーに読み取り専用の権限を付与したい場合は、［**選択**］をオンにします。このオプションにより、データの読み取り専用の権限がすべてのユーザーに与えられます。すべてのユーザーに更新権限を付与する場合は［**更新**］をオンにし、その他同様に行います。

ユーザーによって異なる権限を付与する場合は、グループ アカウント（ドメイン認証を使用する場合）または個々のユーザー アカウント（ローカル データベース認証を使用する場合）を作成する必要があります。これは、SQL で GRANT ステートメントを使用するか、または ZenCC を使用して行います。詳細については、『*Zen User's Guide*』の「セキュリティの作業」を参照してください。

16 ［OK］をクリックします。

17 お使いのオペレーティング システムの手順に従って、オペレーティング システムでデータ ファイルをセキュリティ保護します。これで、データベース エンジンを介してデータにアクセスする能力に影響を与えることなく、オペレーティング システム ユーザーがデータ ファイルに対するあらゆる権限も持つことを拒否できるようになりました。

注意 必ずオペレーティング システムでデータ ファイルをセキュリティ保護してください。この手順を実行しないと、ユーザーは依然としてこの処理より前の段階と同じレベルの権限によって、オペレーティング システムからファイルにアクセスできてしまいます。ユーザーが直接ファイルを変更、削除できないようにするには、ユーザーのデータ ファイルに対するオペレーティング システム権限を取り消す必要があります。

ネットワーク上のデータの暗号化

Zen では、Zen とそれを呼び出すアプリケーションの間のネットワーク トラフィックを暗号化することができます。この種の暗号化は、多くの場合「**ワイヤ暗号化**」と呼ばれます。これは、データがネットワーク ワイヤ上、あるいはワイヤレスも含め、あらゆるネットワーク インフラストラクチャ上を通るときに保護されるからです。ワイヤ暗号化の使用が要求されていないときは、アプリケーションによって送信されるデータへの不正アクセスに対する追加の抑止力を提供します。

Zen のワイヤ暗号化は、特定のセキュリティ モデルとは関係ありません。どの Zen セキュリティ設定も、ワイヤ暗号化をオンにして、またはオンにしないで使用できます。このトピックの残り部分では、以下の項目について説明します。

- 「ワイヤ暗号化の設定プロパティ」
- 「ワイヤ暗号化に関する注記」
- 「暗号化を設定する」
- 「暗号化の影響」
- 「ディスク上のファイルの暗号化」

ワイヤ暗号化の設定プロパティ

ワイヤ暗号化に関連する設定は 2 つあります。これらの設定は、サーバーはもちろん、各クライアント マシンで設定する必要があります。これらの設定の詳細については、以下のトピックを参照してください。

- 「ワイヤ暗号化」
- 「ワイヤ暗号化レベル」

▶▶ ワイヤ暗号化の設定にアクセスするには

1 ZenCC で、次のいずれかを実行します。

- サーバーの場合は、［**エンジン**］ノードの下にあるサーバー名を右クリックします。プラス（+）記号をクリックすると、ノードを展開させることができます。
- クライアントの場合は、［**ローカル クライアント**］ノードの下にある［MicroKernel ルーター］を右クリックします。

2 ［**プロパティ**］をクリックします。

3 ツリーで［**アクセス**］をクリックします。

ワイヤ暗号化に関する注記

Zen では、ネットワーク経由でデータを渡す前にデータの暗号化を行うために、Blowfish という有名かつ実績のあるパブリック ドメイン アルゴリズムを使用します。この手法では 40 ビット、56 ビット、および 128 ビットのキーを実装します。40 ビット キーを使用する暗号化では、最少量の保護がデータに提供されます。56 ビット キーまたは 128 ビット キーを使用する暗号化では、解析が次第に困難になります。

暗号化と複合化の実行にはある程度のプロセッサ時間が必要なので、暗号化を使用するすべてのセキュリティと同様に、抑止力が向上すればパフォーマンスは低下します。

以前のバージョンとの互換性

ワイヤ暗号化をサポートしていない旧バージョンの Zen は、暗号化を提供する新しいリリースのクライアントまたはサーバーと通信することができません。暗号化をサポートしていないクライアントまたはサーバーは、暗号化を使用するクライアントまたはサーバーに接続しようとすると、エラーを返します。

暗号化を設定する

お使いの環境で暗号化の設定をオンにする前に、まず暗号化の必要性についてお考えください。状況に応じて、暗号化環境を次の 4 つの方式から選択できます。

- 暗号化なし
- すべての通信を暗号化
- 特定のクライアントとの通信を暗号化
- 特定のサーバーとの通信を暗号化

暗号化なし

まず最初に、そのデータに暗号化が必要であると考えられる特性があるかどうかを検討します。データは機密か？権限のないユーザーの管理下にあって利用価値のあるデータか？組織に危害を加えるために利用することができるか？これらの質問やその他類似する質問の答えが「いいえ」である場合は、データを暗号化する必要はまったくなさそうです。このような状況では、暗号化を必要とする代わりにパフォーマンスが低下することを受け入れるメリットがありません。確信が持てない場合は、データ セキュリティのエキスパートにご相談ください。

データを保護する必要があるとしても、まだ暗号化する必要はない可能性があります。アプリケーションを LAN 上で単独に実行しており、ネットワークの既存のセキュリティに満足しているのであれば、暗号化を実行しても利点がない可能性があります。

特定のクライアントとの通信を暗号化

次に、あなたのデータベースに接続する大口の取引先がリモート サイトにあるとします。リモート クライアントとの通信でのみ暗号化を使用したいと思われるかもしれません。これは、リモート クライアントの［**ワイヤ暗号化**］を［**常時**］に設定し、そのリモート クライアントがアクセスするサーバー値を［**必要な場合**］に設定することによって実現できます。内部クライアントはすべて［**しない**］に設定します。このようにして、サーバーは、暗号化を要求するリモート クライアントと通信する場合にのみ暗号化を使用するようになります。

特定のサーバーとの通信を暗号化

立場が逆転して、お使いの環境に 1 つ以上のリモート サーバーが入っており、それらのサーバーはあなたが 100% 信頼していないネットワーク インフラストラクチャによってアクセスされているものとします。この場合は、それらのサーバー値を［**常時**］に設定し、ローカル クライアント値を［**必要な場合**］に設定することができます。そうすると、暗号化を要求するリモート サーバーとの通信のみが暗号化されます。

すべての通信を暗号化

最後に、Zen アプリケーションを頻繁に WAN や VPN、またはその他の完全に信頼していない外部ネットワーク上で実行している場合は、データベース通信を 100% 暗号化したいと思われるかもしれません。この状況では、**すべて**のクライアントとサーバーで、［**ワイヤ暗号化**］を［**常時**］に設定します。

暗号化レベルの選択

クライアントおよびサーバーは暗号化された通信を必要とするという結論に達したら、どの抑止レベルが要望に沿うかを決定する必要があります。

Actian Corporation は、お客様の特定の要望に沿った暗号化レベルについてアドバイスすることはできませんので、お客様が適切なデータ セキュリティ専門家を交えて検討される際に情報提供できるよう、いくつかのガイドラインを用意しています。これらのガイドラインは、暗号化されたデータを第三者から読み取られたりデコードされたりしないことを Actian Corporation が保証するものではありません。あらゆる暗号化方式と同様に、「解読できない」コードの類はありません。暗号化の種類によって解析する難易度が変化するだけです。Zen で使用する 128 ビット暗号化は、極めて専門的な個人ハッカーが持てる技術と設備をもってしても、デコード（復号）することが非常に困難であると認められています。

低度（40 ビット）暗号化

データが渡ってはいけない人の手に渡ってしまったとしても、そのデータによって組織や顧客が被る被害レベルがそれほど深刻でない場合には、この暗号化レベルの使用を検討してください。低度の暗号化を検討するもう 1 つの判断材料は、単純に、データがワイヤを通っている最中に不用意なネットワーク監視者にデータを読み取られたくないかどうかということです。

中度（56 ビット）暗号化

セキュリティに無頓着な監視者に備えてもっと強力な保護が何か必要であると確信しているが、最強レベルのセキュリティが必要だとは思わないような状況では、この暗号化レベルの使用を検討してください。

高度（128 ビット）暗号化

データにクレジット カード番号、社会保障番号、金融口座番号、または法律で保護されているその他の情報など、非常に秘匿性の高い情報が含まれているような状況では、この暗号化レベルの使用を検討してください。特に、お使いのデータベースが、インターネット ショッピング Web サイトやオンライン証券会社 Web サイトなど、慎重を期するデータを含んでいることで知られるネットワークのエンティティと関連付けられている場合には、この暗号化レベルを考慮してください。また、所属する組織が以前にデータ セキュリティを解析されそうになったことがある場合には、この暗号化レベルを検討してください。

暗号化の影響

暗号化を使用すると、クライアント / サーバーのパフォーマンスは低下します。暗号化をオンにした場合は、送信側でデータの各断片を暗号化し、受信側でそれをデコードする必要があります。暗号化なしで同様の操作を実行した場合と比べ、この処理には余分な CPU サイクルが必要となります。暗号化のレベルはパフォーマンスに影響しません。暗号化の使用によるパフォーマンスの低下は、3 つの暗号化レベルのうちどれを選択しても、おおよそ同じです。

ディスク上のファイルの暗号化

Zen を使用すると、データ ファイルをディスクに書き込む際に暗号化することができます。この機能を使用するには、各ファイルにオーナー ネームを設定し、暗号化オプションを選択する必要があります。詳細については、「オーナー ネーム」を参照してください。

ログ、バックアップおよび復元

9

ログ、バックアップおよびデータの復元について

Zen には、データ整合性を確実にし、オンライン バックアップおよびデータ被害の復旧をサポートするいくつかの強力な機能があります。

- 「トランザクション ログおよびトランザクション一貫性保持」
- 「アーカイブ ログおよび Continuous オペレーションについて」
- 「アーカイブ ログの使用」
- 「Continuous オペレーションの使用」
- 「Backup Agent および VSS Writer によるデータ バックアップ」

トランザクション ログおよびトランザクション一貫性保持

Zen はトランザクションに関わるデータベース オペレーションのデータ整合性を「トランザクション ログ」および「トランザクション一貫性保持」の 2 つのレベルで保証します。

ここでは、以下の項目について説明します。

- 「これらの機能の使用法」
- 「機能の比較」
- 「どちらの機能を使用するか」
- 「ログの機能」
- 「関連項目」

これらの機能の使用法

これらの機能はいずれも、Zen Control Center 内の設定を使用して、または Distributed Tuning Interface を使用してプログラムから、データベース エンジン内で有効または無効にすることができます。「トランザクション一貫性保持」および「トランザクション ログ」を参照してください。

「トランザクション一貫性保持」のデフォルト値は**オフ**で、「トランザクション ログ」のデフォルト値は**オン**です。

機能の比較

これらの機能は共に複数ファイルのトランザクション アトミシティを提供し、確実にデータ ファイルの一貫性を保ち、未完了のトランザクションはどのデータ ファイルにも絶対に書き込まれないようにします。

アトミシティとは、トランザクション内のあるデータ オペレーションが完了しなかった場合にはトランザクション内のどのオペレーションも完了させないことを意味します。アトミックな変更はデータベースに部分的またはあいまいな変更を残しません。個々のファイルに対する変更は、トランザクション ログおよびトランザクション一貫性保持がオンでもオフでも、常にアトミックです。ただし、トランザクションでは、複数ファイルへの変更をグループ化して 1 つのアトミック グループにすることができます。これら複数ファイルのトランザクションのアトミシティは、アプリケーション内でトランザクションを使用し、トランザクション ログまたはトランザクション一貫性保持がオンに設定されている場合にのみ、MicroKernel が保証します。

これらの利点に加え、「トランザクション一貫性保持」はシステム障害時に、障害前にアプリケーションに正常終了のステータス コードを返したトランザクションの結果がすべてデータ ファイルに含まれていることを保証します。

高パフォーマンスが求められる場合には、「トランザクション ログ」はこの保証を提供できません。「トランザクション一貫性保持」は、エンジンが正常終了のステータス コードを返す前に完了したトランザクションを完全にトランザクション ログに書き出すことを確実にするため、トランザクション ログはロガー スレッドがログ バッファーをディスクにフラッシュするように指示されるとすぐに正常終了のステータス コードを返します。

［トランザクション ログ］は［トランザクション一貫性保持］のサブセットです。つまり、［トランザクション一貫性保持］の設定をオンにすると、ロギングが行われ、［トランザクション ログ］の設定はデータベース エンジンによって無視されます。

［トランザクション ログ］と［トランザクション一貫性保持］の主な違いを次の表で示します。

表 23　［トランザクション ログ］と［トランザクション一貫性保持］: 利点

機能	複数ファイル間でのデータの整合性とトランザクション アトミシティの保証	正常終了のステータス コードを返したすべての完了済みトランザクションに対するコミットの保証
トランザクション ログ	**可**	**不可**
トランザクション一貫性保持	**可**	**可**

表 24 ［トランザクション ログ］と［トランザクション一貫性保持］: 機能

機能	ログ バッファーをディスクに書き込むタイミング
トランザクション ログ	ログ バッファーがいっぱいになったとき、または［起動時間制限］の設定値に達したときに、ログ バッファーはログ ファイルに書き込まれます。各 End Transaction オペレーションの正常終了を示すステータス コードは、ロガー スレッドがバッファーをディスクにフラッシュするように指示されるとすぐにアプリケーションに返されます。
トランザクション一貫性保持	End Transaction オペレーションごとに、ログ バッファーはトランザクション ログ ファイルに書き込まれます。各 End Transaction オペレーションの正常終了を示すステータス コードは、ログのディスク書き込みが正常終了するまでアプリケーションに返されません。トランザクションには含まれない Insert または Update オペレーションの場合は、ログ バッファーがいっぱいになったとき、または［起動時間制限］の設定値に達したときにログ バッファーがログ ファイルに書き込まれます。

どちらの機能を使用するか

最高のパフォーマンスを得るためには、トランザクションの安全性のニーズに合った最低レベルのロギングを使用することができます。適切なロギング レベルを決定するには、アプリケーション ベンダーに確認することが最良の方法です。同じコンピューター上の Zen を使用する複数のアプリケーションがある場合は、それらアプリケーションのなかで要求される最高レベルのロギングを使用する必要があります。

データ ファイルが 1 つのみであるか、複数データ ファイルによるトランザクションを行うアプリケーションがない場合は、一般的に「トランザクション一貫性保持」または「トランザクション ログ」を使用する必要はありません。このような状況下では、Zen はログの有無に関わらず、各データ ファイルの内部的な整合性を保証します。

トランザクション ログ

Zen アプリケーションのうち 1 つでも複数データ ファイル間でトランザクションを行うアプリケーションがある場合は、［トランザクション ログ］をオンにします。「トランザクション ログ」を使用しないと、Zen ではトランザクションの複数ファイルのアトミシティまたは複数ファイルのデータ整合性を保証することはできません。

システム障害が発生した場合、このレベルのロギングでは、完了した各トランザクションがデータ ファイルに書き込まれることが保証されません。

トランザクション一貫性保持

Zen アプリケーションのうち 1 つでも、複数データ ファイル間の完了したトランザクションが確実にデータ ファイルに書き込まれるよう保証することが必要な場合は、「トランザクション一貫性保持」をオンに設定します。

システム障害が発生した場合、このレベルのロギングは正常終了した各トランザクションがデータ ファイルに書き込まれることを保証します。

ログの機能

これらの機能は、オペレーションではなくトランザクションのアトミシティを確実にするものであることに注意してください。SQL を使用する場合、トランザクションは BEGIN ステートメントまたは START TRANSACTION ステートメントと END または COMMIT ステートメントの間にある一連のオペレーションと定義されます。Btrieve を使用する場合、トランザクションは Start Transaction オペレーションと End Transaction オペレーションの間にある一連のオペレーションと定義されます。

すべてのデータ ファイルの追加および更新はログ バッファーに格納されます。トランザクションが完了するか（トランザクション一貫性保持）、バッファーがいっぱいになるか、「起動時間制限」に達する（トランザクション一貫性保持またはトランザクション ログ）と、バッファーはトランザクション ログ ファイルに書き込まれ（フラッシュされ）ます。

トランザクション ログの場合、エンジンがトランザクション終了のオペレーションを受け取り、ロガー スレッドにログ バッファーをディスクにフラッシュする指示を出すことに成功した場合、エンジンはアプリケーションに正常終了のステータス コードを返しトランザクションが開始します。トランザクション一貫性保持の場合、エン

ジンはロガー スレッドがバッファーをディスクに書き込むことに成功したことを示すまで正常終了のステータスを返しません。

トランザクション ログ ファイル セグメントは［トランザクション ログのディレクトリ］に設定したロケーションに保存されます。ログ セグメントは *.LOG という名前で、00000001 から FFFFFFFF までのプレフィックスが付きます。

メモ トランザクション ログまたはトランザクション一貫性保持が有効な場合、すべてのオペレーションは、トランザクション内で行われたかどうかに関わらず、ログ ファイルに書き込まれます。ただし、トランザクション内で実行されたオペレーションのみがアトミックであることが保証されます。システム障害が発生し、トランザクション ログがロール フォワードされる場合、完了したトランザクションのみがデータ ファイルにコミットされます。対応する End Transaction オペレーションを持たないオペレーションは拒否され、データ ファイルにコミットされません。

ヒント データベースの使用率が高い場合には、データ ファイルが存在するのとは物理的に異なるボリュームにトランザクション ログが保持されるようにシステムを構成する必要があります。一般的に高負荷の下では、単一ドライブで I/O 帯域幅を競う代わりに、ログ ファイルへの書き込みとデータ ファイルへの書き込みを別々のドライブに分ける方がパフォーマンスがよくなります。全体的なディスク I/O は減少しませんが、ロードはディスク コントローラー間で効率よく分散されます。

設定プロパティの［トランザクション ログのディレクトリ］設定を使用して、トランザクション ログのロケーションを指定することができます。

システム障害の発生時期が、ログ ファイルが書き込まれた後で、コミットされたオペレーションがシステム トランザクション内でデータ ファイルにフラッシュされる前であれば、コミットされたオペレーションが失われることはありません。コミットされたオペレーションをフラッシュするには、影響を受けるファイルをシステム障害後に開き、オペレーションを実行する必要があります。ファイルを開いてオペレーションの実行を試みると、システム障害の発生時点で影響を受けていたファイルにデータがロール フォワードされます。単にデータベース エンジンを再起動すると、ロール フォワードは呼び出されず、データの一貫性が保たれません。

メモ ロール フォワードされたファイルに関連付けられているログ ファイルは自動的に削除されません。これは、ログ ファイルが 2 つ以上のデータ ファイルに関連付けられている可能性があるからです。

この機能により、個々のクライアント トランザクションは、成功を示すステータス コードをすばやく受け取ることが可能になると同時に、複数のクライアント トランザクションをグループにまとめてデータ ファイルに順次書き出すことでパフォーマンスが向上するという利点が得られます。

データベース サーバーのデータ ファイルが格納されているボリュームのディスクに障害が発生し、アーカイブ ログからデータを復元する必要がある場合、エンジンはトランザクション ログ ファイルをロール フォワードしません。アーカイブ ログにはトランザクション ログのすべてのオペレーションが含まれているため、トランザクション ログをロール フォワードする必要がありません。

ヒント システム障害後、すべてのデータ ファイルを開き、それらのファイルに対して Stat オペレーションまたは読み取りオペレーションを実行してください。すべてのデータが復元されていることを確認できたら、古いログ ファイルは安全な場所に保管してもかまいません。

関連項目

詳細については、以下を参照してください。

- 「トランザクション一貫性保持」
- 「トランザクション ログ」
- 「トランザクション ログのディレクトリ」

アーカイブ ログおよび Continuous オペレーションについて

この製品は、オンライン バックアップおよびデータ被害復旧をサポートするための相互に排他的な 2 つの機能を提供します。

状況	使用する機能
バックアップ実行中もデータベース アプリケーションを実行し続ける必要がある。	Continuous オペレーション
バックアップを実行するのにエンジンをシャット ダウンすることができる。	アーカイブ ログ

アーカイブ ログを使用すると、最後のバックアップ以降のデータベース オペレーションのログを保存することができます。システム障害が発生した場合、バックアップからデータ ファイルを復元し、ログ ファイルから変更をロールフォワードして、システムをシステム障害発生前の状態に戻すことができます。

注意 アーカイブ ロギングは、アーカイブ ログからの復元後、すべてのデータ ファイルが一貫性のある状態になることを保証しません。速度向上のために、データベース エンジンはログ関数から成功を示すステータス コードが返されるのを待たずに、ログ バッファーを空にします。そのため、ディスク フルやオペレーティング システムの書き込みエラーなどのめったに起こらない状況において、データ ファイルで正常に行われた更新がアーカイブ ログに記録されないことがあります。また、アーカイブ ログではすべてのファイルのログをとる必要がないため、複数のファイルの更新を行うトランザクションがあるとき、そのうち一部のファイルのログしかアーカイブしていない場合は、トランザクションがアーカイブ ログに完全に記録されないことがあります。結果として、あるファイルがほかのファイルと矛盾する可能性があります。トランザクションを使用しており、複数ファイルのトランザクション アトミシティが必要な場合は、「トランザクション ログおよびトランザクション一貫性保持」を参照してください。

Continuous **オペレーション**を使用すると、データベース エンジンが実行中でユーザーが接続中でもデータベース ファイルのバックアップを行うことができます。Continuous オペレーションの開始後、データベース エンジンはアクティブなデータ ファイルを閉じ、すべての変更をテンポラリ データ ファイル（**デルタ** ファイルと呼びます）に格納します。Continuous オペレーションが機能している間に、データ ファイルのバックアップを実行します。バックアップ中にデータ ファイルに対して行われた変更はすべてデルタ ファイルに記録されます。

バックアップが完了したら、Continuous オペレーションを解除します。データベース エンジンはデルタ ファイルを読み取り、元のデータ ファイルにすべての変更を適用します。この一時デルタ ファイルは、Continuous オペレーション中に多くの変更が行われた場合、元のデータ ファイルのサイズを超えることがあります。

Continuous オペレーションが設定されているファイルは、リレーショナル エンジンおよび MicroKernel エンジンからそのデータ ファイルが削除されないようロックします。さらに、このファイルはキーの変更などファイル構造の変更も受け付けません。

メモ アーカイブ ログおよび Continuous オペレーションは相互に排他的な機能で、同時に使用することはできません。

アーカイブ ログとトランザクション ログの違い

トランザクション ログは、システム障害時のデータの完全性を守るためにデザインされたもう 1 つの機能ですが、アーカイブ ログとは直接関連を持ちません。トランザクション ログは、アーカイブ ログまたは Continuous オペレーションのいずれかと同時に機能させることができます。トランザクション ログは、短期間のログ ファイルを使用して、トランザクションが確実にディスクに書き込まれるようにします。トランザクション ログは、シ

ステム トランザクションのために、完了したクライアント トランザクションが物理データ ファイルへ移行するときに、頻繁にリセットされます。システム障害が発生すると、データベース エンジンが再起動されたときにトランザクション ログを読み取り、システム障害より前に完了していたトランザクションをデータ ファイルにいっきに移します。

アーカイブ ログは、各システム トランザクションの完了時に書き込まれるので、システム トランザクション中にシステム障害が起こらない限り、アーカイブ ログおよびトランザクション ログは適切に同期しています。

トランザクション ログの詳細については、「トランザクション ログおよびトランザクション一貫性保持」を参照してください。

ファイルの復元が必要になったら

バックアップからデータ ファイルを復元する必要のあるシステム障害が発生した場合、アーカイブ ログを使ってバックアップから復元してデータベースの動作状態をシステム障害の時点に回復することができます。

アーカイブ ログを使用せずに同様の障害が起きた場合（たとえば、バックアップの実行に Continuous オペレーションを使用していた場合）、最後のバックアップからシステム障害の間のデータベースの動作を回復することはできません。

表 25　システム障害後のデータ回復の限界

アーカイブ ログ	障害後に回復できないデータ
オン	障害時に未完了のトランザクション
オフ	データ ファイルの最後のバックアップ以降に起きたすべてのデータベース オペレーション

この章では、これ以降、アーカイブ ログおよび Continuous オペレーションに関連するオプションと手順について説明します。

アーカイブ ログの使用

このセクションでは、アーカイブ ログの設定、データ ファイルのバックアップおよび復元を行う方法について説明します。以下のトピックに分かれています。

- 「全般的な手順」
- 「アーカイブ ログの設定」
- 「ロール フォワード コマンド」

全般的な手順

アーカイブ ログが適切に動作するためには、明確に定義された手順に従って設定する必要があり、バックアップからの復元が必要になった場合には別の手順が必要です。

注意 この手順のいずれかの段階が省かれたり、十分でない場合、バックアップからデータを復元することができません。

▶▶ アーカイブ ログを適切に使用するには

1 まだ動作していない場合は、アーカイブ ログをオンにします。設定の詳細については「アーカイブ ログの設定」を参照してください。

2 データベース エンジンをシャット ダウンします。

3 データ ファイルをバックアップします。

4 バックアップに成功したら、既存のアーカイブ ログをすべて削除します。

注意 データ ファイルでの作業再開**前**にそれぞれのログ ファイルを削除します。バックアップ データ ファイルとそのログ ファイルが同期していることは、回復作業を成功させるための重要事項です。

5 データベース エンジンを再起動します。

▶▶ バックアップからデータ ファイルを復元し、アーカイブ ログの変更を適用するには

メモ ハード ディスクがクラッシュし、アーカイブ ログおよびデータ ファイルの両方がそのディスク上にあった場合は、この手順でアーカイブ ログをロール フォワードすることはできません。

1 システム障害後コンピューターを再起動したら、データベース エンジンが実行中でないことを確認し、復元しようとしているデータ ファイルにほかのデータベース エンジンがアクセスしていないことを確認してください。

2 バックアップからデータ ファイルを復元します。

3 データベース エンジンを起動し、どのようなアプリケーションもエンジンに接続していないことを確認します。

注意 アーカイブ ログがデータ ファイルに適用される前にデータベース アクセスが行われないようにすることは重要です。ほかのデータベース エンジンがファイルにアクセスしていないことを確認してください。システム障害にあったのと同じエンジンを使用してアーカイブ ログをロール フォワードする必要があります。

4 「ロール フォワード コマンド」に説明されているロール フォワード コマンドを発行します。

5 ロール フォワードが正常に完了したら、データベース エンジンを停止し、データ ファイルの新しいバックアップを作成します。

6 データ ファイルが正常にバックアップできたら、アーカイブ ログ ファイルを削除します。これで、データベース エンジンを再起動し、アプリケーションがデータ ファイルにアクセスできます。

アーカイブ ログの設定

アーカイブ ログの設定には 2 つの段階があります。

- アーカイブ ログ機能を有効にする
- アーカイブするファイルとそれぞれのログ ファイルを指定する

メモ これらの処理を行うには、データベース エンジンが起動しているコンピューターに対し管理者権限を持っているか、データベース エンジンが起動しているコンピューター上の Zen_Admin グループのメンバーである必要があります。

▶▶ アーカイブ ログを有効にするには

1 オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から Control Center にアクセスします。

2 Zen エクスプローラーで、ツリーの**エンジン** ノードを展開します（ノードの左の展開アイコンをクリックします）。

3 アーカイブ ログを指定するデータベース エンジンを右クリックします。

4 ［プロパティー］をクリックします。

5 ツリー内で［**データ整合性**］をクリックし、そのカテゴリに含まれるオプションの設定内容を表示します。

6 ［**選択ファイルのアーカイブ ロギング**］チェックボックスをクリックします。

7 ［OK］をクリックします。

設定を有効にするにはエンジンの再起動が必要であることを知らせるメッセージが表示されます。

8 ［**はい**］をクリックして、エンジンを再起動します。

▶▶ アーカイブするファイルを指定するには

アーカイブ ログを実行するファイルは､そのファイルを含むボリュームに作成したアーカイブ ログ設定ファイルに、エントリを追加することによって指定します。設定ファイルは、以下の方法で設定します。

1 ログを行う対象のデータ ファイルが存在する物理ドライブのルート ディレクトリに、¥blog ディレクトリを作成します（マップされたルート ディレクトリは使用しないでください）。ファイルが複数のボリュームに分かれている場合は、その各ボリュームに ¥blog ディレクトリを作成します。

たとえば、C:¥ および D:¥ にデータ ファイルが存在し、どちらのドライブもデータベースと同じコンピューター上に存在する物理ドライブである場合、次のように 2 つの blog ディレクトリを作成します。

```
C:¥blog¥
D:¥blog¥
```

メモ Linux、macOS、および Raspbian では、ログ ディレクトリは blog という名前で、ACTIANZEN_ROOT 環境変数で指定したディレクトリ（デフォルトでは /usr/local/actianzen）に作成する必要があります。

2 各 ¥blog ディレクトリに、空の blog.cfg ファイルを作成します。blog.cfg の作成には、メモ帳のような任意のテキスト エディターを使用してください。Linux、macOS、および Raspbian では、ファイルの名前は blog.cfg（小文字）である必要があります。

3 各 blog.cfg ファイルに、そのドライブ上にあるアーカイブ ログを実行するデータ ファイルのエントリを入力します。エントリの入力には、以下の形式を使用します。

¥ パス 1¥ データ ファイル 1 [=¥ パス 2¥ ログ ファイル 1]

パス 1	ログを行う対象のデータ ファイルのパス。パスには、ドライブ名は含めません。
データ ファイル 1	ログを行う対象のデータ ファイルの名前。
パス 2	ログ ファイルのパス。ログ ファイルとデータ ファイルは、異なるドライブ上に存在する場合があるため、必要に応じてパスにドライブ名を追加します。
ログ ファイル 1	ログ ファイルの名前。名前を指定しない場合、データ ファイルと同じディレクトリおよびファイル名プレフィックスがデフォルト値となりますが、ファイル名サポートフィックスは ".log" に置き換えられます。異なる物理ドライブを指定して、ログとデータ ファイルを同じドライブに置かないようにすることができます。ログをとるデータ ファイルごとに別々のログ ファイルが必要です。

エントリにはスペースを使用せず、1 行以内に収める必要があります。（各行最大で半角 256 文字）余裕があれば同一行に複数のエントリを置くことができます。エントリはスペースで区切ります。

注意 ログをとるデータ ファイルすべてに異なるログ ファイルを使用する必要があります。2 つ以上のデータ ファイルに同一のログ ファイルを使用すると、MicroKernel はロールフォワードが必要になったときにそのログ ファイルを使用できません。

ログ ファイルの名前を入力しない場合は、ログ ファイルを最初に開いた際に、元のファイル名に拡張子 .log を付けたファイル名が割り当てられます。たとえば、ファイル b.btr では、ログ ファイルに b.log が割り当てられます。

注意 データベースのすべてのファイルのログをとる必要はありません。ただし、データベースに参照整合性（RI）規則が指定されている場合は、各 RI の関係に含まれるすべてのファイルのログを指定するかまたは 1 つも指定しないでください。RI の関係に含まれるファイルのサブセットのみのログを指定すると、システム障害後にアーカイブ ログをロール フォワードするとき、RI 規則違反となります。

例

以下は、ドライブ C 上にある blog.cfg のエントリの例です。すべてのエントリは、同じ結果になります。C:¥data¥b.bti の処理は、C:¥data¥b.log に記録されます。

```
¥data¥b.bti
¥data¥b.bti=¥data¥b.log
¥data¥b.bti=c:¥data¥b.log
```

次の例では、C:¥data¥b.bti の処理が、ログ ファイル D:¥data¥b.lgf に記録されます。これは、アーカイブ ログ ファイルが、データ ファイルと同じドライブ上に存在する必要がなく、拡張子が「.log 」である必要もないことを表します。ただし、.log 以外の拡張子が使用されない場合は、デフォルトで .log 拡張子が提供されます。

```
¥data¥b.bti=d:¥data¥b.lgf
```

ヒント 同じコンピューターの異なる物理ドライブにログを書き込むことをお勧めします。ログ ファイルを異なる物理ドライブに置けば、ハード ディスクがクラッシュした場合にデータ ファイルといっしょにログ ファイルも失うことを防げます。

次に、MicroKernel が複数のファイルのログを異なるドライブ（ドライブ D）に作成するようにする blog.cfg の例を示します。blog.cfg ファイルはドライブ C にあるものとします。

```
¥data¥file1.mkd=d:¥backup¥
¥data¥file2.mkd=d:¥backup¥file2.log
¥data¥file3.mkd=d:¥backup¥file3.log
```

ロール フォワード コマンド

Btrieve Maintenance ツール一（GUI またはコマンド ラインの butil）は、アーカイブ ログ ファイルをデータ ファイルにロール フォワードするコマンドを提供します。「アーカイブ ロギングの実行」を参照してください。

Continuous オペレーションの使用

Continuous オペレーションは、データベース アプリケーションが実行中でユーザーが接続されている間にデータ ファイルのバックアップを行う機能を提供します。ただし、ハード ドライブ障害の場合、Continuous オペレーションを使用してバックアップを作成していた場合には、最後のバックアップ以降のデータの変更はすべて失われます。アーカイブ ログおよび Maintenance ツールのロール フォワード コマンドを使用して最終バックアップ後に起こったデータ ファイルへの変更を復元することはできません。

Zen ではバックアップ機能として Continuous オペレーションを butil コマンド内で提供します。

メモ Continuous オペレーションが設定されているファイルは、リレーショナル エンジンおよびトランザクショナル エンジンからそのデータが削除されないようロックします。さらに、このファイルはキーの変更などファイル構造の変更も受け付けません。また、Actian Corporation は関連製品として Backup Agent を提供し、Continuous オペレーションを設定および管理します。

このセクションは、以下のトピックに分かれています。

- 「Continuous オペレーションの開始と停止」
- 「Butil を使用してデータベースのバックアップを行う」
- 「Continuous オペレーション使用時のデータ ファイルの復元」

Continuous オペレーションの開始と停止

このトピックでは、「STARTBU」および 「Endbu」コマンドについて説明します。

表 26　Continuous オペレーションの開始と停止を行うコマンド

コマンド	説明
「STARTBU」	バックアップを対象として定義されたファイルの Continuous オペレーションを開始します。（BUTIL）
「Endbu」	バックアップを対象として定義されたファイルの Continuous オペレーションを停止します。（BUTIL）

注意 Continuous オペレーション モードによって作成されたテンポラリ デルタ ファイルは対応するデータ ファイルと同じ名前ですが、拡張子 ".^^^" を使用します。同じディレクトリに、ファイル名が同一で拡張子のみが異なるようなファイルを置かないでください。たとえば、データ ファイルに invoice.hdr および invoice.det というような名前の付け方をしないでください。もしそうした場合、MicroKernel はステータスを返し、ファイルは Continuous オペレーションに入れられません。

Continuous オペレーション モードは、MicroKernel のパフォーマンスに大きな影響を及ぼすことはありません。しかし、ファイルのバックアップにサーバーを使用した場合は、影響が出ることがあります。

▶▶ Continuous オペレーションを使用してデータの消失を防止するには

1　STARTBU コマンドを使用し、ファイルに Continuous オペレーションを適用します。butil のコマンド構文の説明については、「STARTBU」を参照してください。

2　データ ファイルをバックアップします。

3　endbu コマンドを使用し、ファイルに対する Continuous オペレーションを解除します。butil のコマンド構文の説明については、「Endbu」を参照してください。

Butil を使用してデータベースのバックアップを行う

このトピックでは、butil コマンドの 「STARTBU」および 「Endbu」を使用したデータベースのバックアップに関する詳細について説明します。

STARTBU

startbu コマンドは、バックアップの目的でファイル（複数可）の Continuous オペレーションを開始します。

形式

```
butil -startbu <sourceFile | @listFile> [/UID<name> </PWD<word>>] [/DB<name>]]
```

sourceFile	Windows プラットフォームのドライブ指定など、バックアップのために Continuous オペレーションを開始するデータ ファイルのフル パス名。 この完全に修飾された名前は、butil を実行しているのと同じマシン上に存在している必要があります。startbu では、割り当てられたドライブは使用できません。
listFile	Continuous オペレーションを開始するファイルの、フル パス名を含むテキスト ファイルの名前。これらのファイル名は、キャリッジ リターン / ライン フィードで区切ります。ファイル名にはスペース文字が含まれる可能性があります。 指定したすべてのファイルを対象に Continuous オペレーションを行えない場合、Maintenance ツールによるファイルへの Continuous オペレーションは一切行われません。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワード。*uname* が使用された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名。省略した場合はデフォルトのデータベースと解釈されます。

メモ この startbu コマンドは、指定したファイルのみに対して Continuous オペレーションを開始します。startbu コマンドでは、ワイルドカードは使用できません。

Linux、macOS、および Raspbian の場合、すべてのスラッシュ（/）パラメーターはスラッシュの代わりにハイフン（"-"）を使用します。たとえば、/DB パラメーターは -DB です。

ファイルに関する考慮点

バックアップ対象のファイルを選択する場合、Continuous オペレーション モードによって作成されるテンポラリ デルタ ファイルは除外することをお勧めします。これらのファイルは、バックアップ時に開かれ使用中だからです。デルタ ファイルがバックアップに含まれる場合、データの復元後、これらのファイルはデータベース エンジンが起動する前に削除しておく必要があります。

Windows サーバーの例

例 A：最初の例は、course.mkd の Continuous オペレーションを開始します。

Windows サーバー の場合

```
butil -startbu file_path¥Zen¥Demodata¥course.mkd
```

Zen ファイルのデフォルトの保存場所については、『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

例 B：以下の例は、startlst.fil にリストされているすべてのファイルに対して Continuous オペレーションを開始します。

```
butil -startbu @startlst.fil
```

startlst.fil は、以下のようなエントリで構成されます。

```
file_path¥Zen¥Demodata¥course.mkd
file_path¥Zen¥Demodata¥tuition.mkd
file_path¥Zen¥Demodata¥dept.mkd
```

Endbu

endbu コマンドは、バックアップに定義されているデータ ファイル（複数可）の Continuous オペレーションを終了します。このコマンドは、**startbu** コマンドを使用して Continuous オペレーションを開始し、バックアップを実行した後に使用します。

形式

```
butil -endbu </A | sourceFile | @listFile> [/UID<name> </PWD<word>> [/DB<name>]]
```

/A	**/A** を指定した場合は、**startbu** で初期化され、現在 Continuous オペレーション モードにあるすべてのデータ ファイルに対する Continuous オペレーションが停止します。
sourceFile	Continuous オペレーションを終了するデータ ファイルのフル パス名（Windows プラットフォームのドライブ指定を含む）。 この完全に修飾された名前は、butil を実行しているのと同じマシン上に存在している必要があります。endbu コマンドではマップ ドライブは使用できません。
@listFile	Continuous オペレーションを終了するデータ ファイル名のリストを含むテキスト ファイルの名前。テキスト ファイルには、各データ ファイルの完全修飾ファイル名が含まれている必要があります。これらのファイル名は、キャリッジ リターン / ライン フィードで区切ります。ファイル名にはスペース文字が含まれる可能性があります。 通常、このデータ ファイルのリストは、「STARTBU」コマンドで使用するリストと同じものになります。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

メモ Linux、macOS、および Raspbian の場合、すべてのスラッシュ（/）パラメーターはスラッシュの代わりにハイフン（"-"）を使用します。たとえば、**/A** パラメーターは **-A** で、`butil -endbu -A` のようになります。

Windows サーバーの例

以下の例では、course.mkd の Continuous オペレーションを終了します。

```
butil -endbu file_path¥Zen¥Demodata¥course.mkd
```

ここでは、現在のディレクトリが F:¥demodata の場合、フル パスを省略して単に `butil -endbu course.mkd` とすることもできます。

Continuous オペレーション使用時のデータ ファイルの復元

バックアップの方法として Continuous オペレーションを使用している場合、最後のバックアップ以降の変更を回復するためのリカバリ ログはありません。最後のバックアップ以後のデータベースの変更はすべて失われますが、残っている可能性があるのは、トランザクション ログに残されたトランザクションだけです。このようなトランザクションは、データベース エンジンの再起動時に自動的にロール フォワードされます。

▶▶ データとデータベース オペレーションを復元するには

1 エラーを解決します。

障害が起きたコンピューターを再起動するのに必要な保守を行います。

2 バックアップからデータ ファイルを復元するか、バックアップからハード ドライブ イメージを復元するか適切な方法をとります。

3 ディスク イメージの一部として Zen が復元されなかった場合は、再インストールします。

注意 デルタ ファイルがバックアップ ファイルに含まれる場合、これらのファイルは、次の手順でデータベース エンジンが起動する前に削除しておく必要があります。

4 データベース エンジンを再起動します。

最後のバックアップ以後に実行されたデータベース オペレーションを再度実行する必要があります。

Backup Agent および VSS Writer によるデータ バックアップ

今まで説明されてきたトピックに加え、Zen Enterprise Server および Cloud Server ではデータのバックアップに対して以下のソリューションも適用します。

- 「Backup Agent」
- 「Zen VSS Writer」

ご自分のバックアップ ソフトウェアが Microsoft のボリューム シャドウ コピー サービス（VSS）を**認識しない**場合は、Backup Agent をご自分のバックアップ ソフトウェアと連携して使用することができます。バックアップ ソフトウェアが VSS を認識する場合は、VSS によるバックアップ時に Zen VSS Writer が自動的に起動します。バックアップ ソフトウェアが既に VSS を認識している場合は、Backup Agent を使用する必要はありません。

Backup Agent および Zen VSS Writer は併用できますが、それに伴う利点は特にありません。どちらか一方の方法を選択すれば、バックアップ処理はより簡潔になります。

Backup Agent

Backup Agent はオプション製品です。この製品はデフォルトではインストールされません。Zen Enterprise Server または Cloud Server のインストール後に、インストールする必要があります。

Backup Agent を使用すれば、Zen データベース ファイルに対する Continuous オペレーションの設定と管理を簡単かつ迅速に行うことができます。Continuous オペレーションの設定と管理は、Zen データベースのバックアップを行う際の重要な部分です。これには Microsoft のボリューム シャドウ コピー サービス（VSS）を使用しません。Backup Agent は開いているファイルに対する Continuous オペレーションの設定と管理を処理し、バックアップ中でもアプリケーションからデータを利用できるようにします。バックアップ作業が完了すると、Backup Agent は自動的に Continuous オペレーションからファイルを取り出し、バックアップ中にキャプチャされたすべての変更をロール インします。

Backup Agent は、市販されている多くのバックアップ アプリケーションと互換性があります。そのバックアップ アプリケーションでは、ほかのアプリケーションを開始および停止できるコマンドを発行できる必要があります（そのコマンドで Backup Agent を開始および停止できます）。

Zen VSS Writer

Microsoft のボリューム シャドウ コピー サービス（VSS）は、ライター、プロバイダーおよびリクエスター コンポーネントで構成されています。Zen はライター コンポーネントである Zen VSS Writer のみで VSS をサポートします。

Zen VSS Writer はデータベース エンジンの機能であり、Zen Enterprise Server および Cloud Server で使用可能です。Zen VSS Writer は、この製品のインストール後に使用できるようになります。Zen VSS Writer は現時点で Zen Workgroup では使用できません。

Zen VSS Writer は Windows オペレーティング システムでのみ使用可能です。ボリューム シャドウ コピー サービスの詳細については、Microsoft Web サイトのテクニカル ドキュメント「SQL Server バックアップ アプリケーション ベンダ向けガイド」を参照してください。

概要

VSS によるスナップショット時、Zen VSS Writer は Zen データおよびトランザクション ログ ファイルすべてに対し、それらが存在するボリュームに関係なく、すべてのディスク I/O **書き込み**動作を阻止します。スナップショットの作成後、Zen VSS Writer はスナップショット中に遅延された書き込みなど、すべてのディスク I/O を再開させます。

Zen VSS Writer はディスク I/O **読み取り**動作を阻止することはありません。書き込みが不要である限り、通常のデータベース処理を継続させることができます。Zen VSS Writer は、VSS サービスおよび VSS リクエスターのバックアップ動作によりパフォーマンスが低下するかもしれませんが、バックアップ フェーズ時は正常に動作します。

Microsoft ボリューム シャドウ コピー機能を使用すれば、バックアップおよび復元製品はバックアップ用にシャドウ コピーを作成することができます。それに含まれるファイルは次のいずれかの状態になります。

1 定義済みで、かつ整合状態にある

2 "crash-consistent state"（クリーン リストアに適さない可能性があります）

以下の条件をすべて満たせば、VSS スナップショットのファイルは定義済み、かつ整合性のとれた状態になります。

1 ファイルのライターが VSS 対応である

2 バックアップおよび復元製品が VSS 対応ライターを認識し、そのライターにスナップショットの準備を通知する

3 VSS 対応ライターはスナップショットの準備に成功する

条件を満たさない場合、ライターのファイルは "crash-consistent state" でバックアップされます。

VSS Writer の詳細

ここでは、Zen VSS Writer の仕様について説明します。

- サポートされるオペレーティング システム

  Zen Server 製品がサポートされる Windows オペレーティング システムは Zen VSS Writer もサポートします。Zen VSS Writer は、マシンのオペレーティング システムおよびインストールされている Zen Server 製品と同じビット数に対応して機能します。Zen VSS Writer の 32 ビット版は 32 ビット マシンのみでサポートされ、64 ビット版は 64 ビット マシンのみでサポートされます。ビット数が一致しない場合、Zen は正しく機能しますが VSS Writer は利用できません。

- サポートされるバックアップの種類

  Zen VSS Writer はデータ ボリュームの手動バックアップまたは自動バックアップをサポートします。Zen VSS Writer は完全ボリューム バックアップおよびコピー ボリューム バックアップに対応し、増分、差分およびログ バックアップはサポートされません。VSS は Zen VSS Writer をコンポーネントとして認識します。ただし、Zen VSS Writer はコンポーネントのバックアップをサポートしません。VSS リクエスターがコンポーネントのバックアップで Zen VSS Writer を呼び出すと、VSS Writer は完全ボリューム バックアップまたはコピー ボリューム バックアップでの同じアクションを実行します。

- 仮想化環境のサポート

  Zen VSS Writer は仮想化環境で VSS バックアップをトリガーする VSS リクエスターをサポートします。仮想マシンのスナップショットを実行しても VSS バックアップは起動しません。

- マルチ ボリューム Zen データ ファイル

  Zen ファイルおよびトランザクション ログが複数のボリューム上に存在することもあります。Zen ファイルをバックアップする場合、ほかのボリューム上にあるトランザクション ログと関連ファイルは同時にバックアップすることを覚えておいてください。互いに独立しているファイルは、関連する Zen ファイルと同時にバックアップする必要はありません。

- バックアップ ソリューションの互換性

  特定のバックアップ製品が Zen VSS Writer を認識するかどうか、スナップショットの準備をライターに通知するかどうかを判断するためには、その製品でバックアップを開始します。バックアップ処理の進行中に、zen.log ファイルで Zen VSS Writer が Frozen または Thawed 状態を記録するかどうかを調べます。バックアップおよび復元製品が、バックアップの準備を Zen VSS Writer へ通知しなかった場合は別のソリューションを使用する必要があります。たとえば、Backup Agent を使用すれば Zen データ ファイルを定義済み、かつ整合性のとれた状態でバックアップできます。

- Zen VSS Writer と復元操作

バックアップ ソフトウェアで復元操作を実行する前に Zen エンジン サービスを停止してください。これを行わないと、VSS Writer が VSS リクエスターに対し、復元に参加できないことを通知します。データの整合性を保証するためには、トランザクション ログがデータ ファイルと共に復元される必要があります。Zen が実行している間に Zen データとトランザクション ログ ファイルが復元された場合、その復元結果は予測不能でデータの破損を招く恐れがあります。

- Zen VSS Writer および Zen Continuous オペレーション

  もう既に Zen Continuous オペレーションまたは Backup Agent を使用する既存のバックアップ処理を利用されている場合もあるでしょう。そのバックアップ処理を Zen および Zen VSS Writer でも引き続き使用することができます。Zen VSS Writer は Continuous オペレーションや Backup Agent の機能を妨げることはありません。ただし、Zen VSS Writer と Continuous オペレーション（または Backup Agent）を**併用**しても利点はありません。どちらか一方の方法を選択すれば、バックアップ処理はより簡潔になります。

  Zen VSS Writer が呼び出されたときにファイルが Continuous オペレーション モードに置かれている場合、VSS Writer はすべての Continuous オペレーションから独立して動作します。VSS バックアップが進行中にファイルが Continuous オペレーション モードに置かれた場合は、バックアップが完了した後に zen.log を見てください。Frozen および Thawed 状態が問題なく完了していること、またデータが定義済みかつ整合状態にあることを確認してください。

  Zen VSS Writer は Microsoft VSS フレームワークも必要とします。Backup Agent では Microsoft VSS フレームワークを使用しません。したがって、Microsoft VSS フレームワークが Zen VSS Writer を呼び出し、I/O 操作が停止された場合でも、Backup Agent はバックアップに参加しません。Backup Agent はバックアップ処理に個別に追加する必要があります。そのバックアップ処理は Backup Agent の開始と停止を行う必要もあります。

高可用性のサポート

10

可用性の高い環境での Zen の使用

この章では、以下の項目について説明します。

技術の概要

Zen は、物理環境および仮想環境において稼働時間を最大化する数々のソリューションと互換性があります。そのようなソリューションは絶えず進化し続けていますが、通常、そのタイプは高可用性、フォールト トレランスおよび障害回復に分類されます。

高可用性

「**高可用性**」の定義は、高可用性ソリューションを提供するソフトウェア ベンダーによって異なる可能性があります。一般的には、ハードウェア障害やソフトウェア障害、または必要なメンテナンスの発生にかかわらず、稼働し続ける時間の基準レベルに対するシステム設計手法を指します。

物理環境において高可用性を保証するための一般的な手法はフェールオーバー クラスタリングです。仮想マシン環境において高可用性を保証するための一般的な手法はマイグレーション（移行）です。

フェールオーバー クラスタリング

Zen はフェールオーバー クラスター環境のリソースとして機能するよう設計されています。この環境では一度に 1 つのサーバー ノードのみが共有記憶域サブシステムにアクセスします。プライマリ ノードがエラーになった場合は、セカンダリ ノードへのフェールオーバー（切り替え）が発生します。フェールオーバー クラスタリングを使用すると、ソフトウェアのアップグレードやハードウェアのメンテナンスの実行時にもシステムを利用可能な状態にしておくことができます。

Zen は Microsoft フェールオーバー クラスター サービスおよび Linux Heartbeat と互換性があります。フェールオーバー クラスタリングの定義および実装については、各ベンダーのドキュメントを参照してください。Zen Enterprise Server と Cloud Server はフェールオーバー クラスタリングに推奨されるエディションです。

「フェールオーバー クラスタリング」を参照してください。

マイグレーション

一般論として、マイグレーションとは、実行している仮想マシンまたはアプリケーションを、クライアントやアプリケーションを切断することなく異なる物理マシン間で移動させることです。通常は、仮想マシンのメモリ、記憶域およびネットワーク接続が移行先に移行されます。

Zen は Microsoft Hyper-V、VMware vSphere および Citrix XenServer で提供されるマイグレーション機能と互換性があります。ホスト名が仮想マシンの移動後も同じままであれば、Zen は正常に動作し続けます。マイグレーションの定義および実装については、各ベンダーのドキュメントを参照してください。

「マイグレーション」を参照してください。

フォールト トレランス

高可用性が稼働時間の予測可能な基準レベルを目指すのに対し、フォールト トレランスはコンポーネントに障害が発生してもシステムを中断することなく稼働させることを目的としています。フォールト トレランスでは同期された共有記憶域が必要です。仮想化環境では、障害が発生した仮想マシンと、置き換わる仮想マシンは異なる物理ホスト上に存在している必要があります。

フォールト トレランスは物理マシンのみを使用して実現します。ただし、仮想環境は、互いに足並みを揃えた仮想サーバーを保持することが容易であるため、物理環境のみという状況は次第に少なくなっています。Zen サーバーは物理環境のみでのフォールト トレランス機能と互換性があります。

仮想環境の場合、Zen は VMware vSphere や Citrix XenServer で提供されるフォールト トレランス機能と互換性があります。フォールト トレランスの定義および実装については、各ベンダーのドキュメントを参照してください。

「フォールト トレランス」を参照してください。

障害回復

障害回復では、大きな障害の発生後にコンピューター操作の復元が必要となります。通常、これには日常的なオフサイトのデータ バックアップや新しい場所で主要な情報システムをアクティブ化する手続きが含まれます。

Zen は、バックアップ物理マシンまたは仮想マシンを初期化する障害回復技術をサポートする主要なハイパーバイザーと互換性があります。すべてのホスト名が仮想マシンの移動後も同じままであれば、Zen は正常に動作し続けます。これにより、短時間でのサーバー置換および回復が可能です。

障害回復の定義および実装については、各ハイパーバイザー ベンダーのドキュメントを参照してください。

「障害回復」を参照してください。

ハードウェア要件

このセクションで述べているすべての技術を利用する場合、ベンダーが提供するハードウェア互換性リストで挙げられているサーバー、ディスク サブシステムおよびネットワーク コンポーネントを選択することをお勧めします。弊社でも、ベンダーの製品との互換性テストを行う際にはこの同じハードウェア要件に従っています。

フェールオーバー クラスタリング

フェールオーバー クラスタリングでは、それぞれが独立したサーバー ノードのグループを提供します。それらのノードはすべて共有データ ファイルまたはボリュームの共有システムにアクセスできます。このフェールオーバー サービスでは、一度に 1 つのサーバーだけがデータ ファイルまたは記憶域ボリュームを制御します。現在アクティブなサーバーで障害が発生した場合は、自動的にクラスター内の次のノードに制御が渡され、そのノードのサーバーが制御を引き受けます。

Zen では、1 つのデータベース エンジンだけがデータにアクセスできます。クラスター内では、すべてのノードに Zen エンジンがインストールされるので、それらのエンジンを、そのまま複数のエンジンではなく、1 つのエンジンとして見えるよう構成する必要があります。

各 Zen エンジンは、クラスター ノードがアクティブである間にそのクラスター ノード単位で別々のライセンス キーで認証される必要があります。これはそのノードが物理マシンまたは仮想マシンのどちらでも同じです。1 つのライセンス キーで、両方のノードを認証することはできません。Actian Zen Enterprise Server ライセンスが 2 本含まれている Enterprise Server - SS-Set（Standby Server Set）製品の購入を推奨します。

ここでは、以下の項目について説明します。

- 「Windows Server 用 Microsoft フェールオーバー クラスター」
- 「Linux Heartbeat」
- 「クラスター環境における Zen の管理」

Windows Server 用 Microsoft フェールオーバー クラスター

ここでは、Microsoft フェールオーバー クラスターへの Zen エンジン サービスの追加について説明します。これは次のことを前提とします。

- フェールオーバー クラスターのインストールおよび構成方法を知っていること。また、Zen エンジン サービスを追加および管理するための情報が必要であること。
- Zen および Zen Control Center（ZenCC）などのユーティリティの使用法をよく理解していること。
- Zen ODBC アドミニストレーターを使用して DSN を設定できること。

一般的な手順

Zen エンジンを含む Windows フェールオーバー クラスターを作成する手順は次のとおりです。

1 Windows フェールオーバー クラスターのノードを設定します。

クラスターに Windows サーバーをセットアップしてから、Zen エンジンを追加する前に、フェールオーバーが正常に動作することを確認する必要があります。たとえば、フェールオーバーが完了し、すべてのリソースが引き続き利用できることを確認したら、次にフェールオーバーを元のノードへ戻して同様の確認を再度行います。サーバー マネージャー ダッシュボードやフェールオーバー クラスター マネージャーを使用してフェールオーバー クラスタリングをセットアップおよび検証する方法については、Microsoft のドキュメントを参照してください。

2 独立した専用システムに共有ファイル サーバーをセットアップします。

Zen エンジンをインストールしたら、この共有データの場所を使用するためにそれらのエンジンを設定します。

3 Zen エンジンを各クラスター ノードにインストールします。

下の表にある手順をご確認ください。次の手順に進む前に、すべての Zen エンジンをインストールしてください。

4 エンジンを設定するために、フェールオーバー クラスター マネージャーに戻ります。クラスター内でインストールされている複数の Zen を、単独のデータベース サーバーとして見えるように設定します。

下の表の手順に進みます。

Windows フェールオーバー クラスターへ Zen をインストールし設定する

次の表では、Windows Server のフェールオーバー クラスターに Zen を追加する際の推奨手順を説明しています。

操作	注記
Zen をクラスター ノードにインストールする	各ノードに Zen サーバーをインストールし、それらすべてに同一の設定を選択します。 Zen データ ファイルが存在する専用の共有記憶域システムには Zen を**インストールしないでください**。 デフォルトで、Zen エンジン サービスは Windows で自動的に開始します。各ノードにインストール後、Actian Zen Enterprise Server サービスのプロパティで、スタートアップの種類を**手動**に変更してサービスを停止します。フェールオーバー クラスター マネージャーが必要に応じて開始します。
役割を追加し、Zen サービスを汎用サービスとして選択する。 アプリケーション データ ファイルに使用される記憶域を選択する。 リソースを追加し、Zen サーバーを汎用サービスとして選択する。	フェールオーバー クラスター マネージャーでこちらの手順を実行します。 Zen エンジンがサービスとして実行している場合、レジストリ レプリケーションはサポートされないので、そのオプションはスキップします。この表の後述の操作で述べているように、各ノードの Zen エンジンに対してデータベース プロパティを手動で設定する必要があります。 ファイル共有を Zen エンジン サービスの依存関係として設定します。[**ネットワーク名をコンピューター名として使う**] オプションを選択します。
必要なファイルとディレクトリを共有記憶域に置く。	通常、Zen エンジン サービスは、ローカル システム アカウント下で実行されます。ローカル システム アカウントに共有の場所への読み取り / 書き込み権限があることを確認します。 Zen をインストールしたアクティブ ノードで、ProgramData ディレクトリにある dbnames.cfg を、共有記憶域内の任意のディレクトリにコピーします。 アクティブ ノードで、ProgramData ディレクトリから以下のディレクトリを、共有記憶域内の同じディレクトリにコピーします。任意で、dbnames.cfg と同じディレクトリにこれらをコピーできます。 • defaultdb • Demodata • tempdb • Transaction Logs

操作	注記
ZenCC を使ってデータベース エンジンのプロパティを設定する。	ZenCC で、クラスター内の現在のアクティブ ノードのエンジンを設定した後、ほかのノードにもそれぞれ同じ設定を行います。 **ディレクトリ**に関するエンジン プロパティを設定します。エンジンを再起動するよう指示されたら［**いいえ**］をクリックします。 • ［**トランザクション ログのディレクトリ**］には、Transaction Logs ディレクトリをコピーする共有ディスク上の場所を入力します。 • ［**DBNames 設定ファイルのディレクトリ**］には、dbnames.cfg ファイルをコピーする共有ディスク上の場所を入力します。 フェールオーバー クラスター マネージャーで、Zen リソースを一旦オフラインにし、オンラインに戻して設定を適用します。 ZenCC で、お使いのサーバーのデータベース ノードの下で、以下のプロパティを設定します。 • DEFAULTDB プロパティの［ディレクトリ］では、［**辞書のロケーション**］と［**データ ディレクトリ**］に、Defaultdb ディレクトリをコピーする共有ディスク上の場所を設定します。 • DEMODATA プロパティの［ディレクトリ］では、［**辞書のロケーション**］と［**データ ディレクトリ**］に、Demodata ディレクトリをコピーする共有ディスク上の場所を設定します。 • TEMPDB プロパティの［ディレクトリ］では、［**辞書のロケーション**］と［**データ ディレクトリ**］に、Tempdb ディレクトリをコピーする共有ディスク上の場所を設定します。

これで、フェールオーバー クラスターの設定が完了しました。フェールオーバーが発生したときには、クライアント接続も失敗し、クライアント アプリケーションを再起動する必要があるかもしれないので注意してください。

メモ クラスター環境の Zen サーバーにパッチを適用する必要がある場合は、弊社テクニカル サポートまでお問い合わせください。

Linux Heartbeat

Heartbeat プログラムは Linux-HA（High-Availability Linux Linux）プロジェクトの中心的なコンポーネントの 1 つです。Heartbeat はあらゆる Linux プラットフォームで動作し、1 つのプロセスで停止ノードの検出、通信およびクラスター管理を行います。

ここでは、Linux Heartbeat への Zen エンジン サービスの追加について説明します。これは次のことを前提とします。

- Heartbeat プログラムのインストールおよび構成方法を知っていること。また、Zen をクラスター サービス グループに追加するための情報が必要であること。
- Zen および ZenCC などの主要ユーティリティの使用法をよく理解していること。

事前要件

Zen をクラスターに追加する前に、Linux Heartbeat が正常に機能していることが必須です。Heartbeat のインストール、動作確認、タスクの実行方法については、「The High Availability Linux プロジェクト」（http://www.linux-ha.org/ja/HomePage_ja）ドキュメントを参照してください。

ほかのアプリケーションの場合と同様に、Zen を追加する前には必須のクラスター コンポーネントを設定します。

Linux Heartbeat フェールオーバー クラスターへ Zen をインストールし設定する

次の表では、Linux Heartbeat に Zen を追加する際の推奨手順を説明しています。

操作	説明
Zen をクラスター ノードにインストールする	各クラスター ノードに Zen サーバーをインストールし、それらすべてに同一のオプションを選択します。Zen データ ファイルが存在するクラスター共有記憶域には Zen を**インストールしないでください**。 インストールが完了すると、オペレーティング システムの起動時にデータベース エンジンが自動的に起動するよう設定されます。ただし、クラスタリングでは、Linux Heartbeat がデータベース エンジンの起動と停止を制御します。クラスターの実稼動ノードがエンジンを起動し、それ以外のノードはエンジンを起動しません。 Zen サーバーをインストールしたら、zen-data および zen-adm 用のグループ ID と zen-svc 用のユーザー ID がすべてのノードにおいて一致していることを確認してください。ID が一致していなければ、同一になるよう変更します。
共有記憶域を構成する	共有記憶域は、Zen データ ファイルが存在する場所です。Heartbeat の共有記憶域はさまざまな方法で実装できます。このマニュアルではその数多くの実装方法のすべてを説明することはできません。そのためここでは、NFS マウントの使用を想定しています。 共有記憶域上で、データベースを配置する場所を作成します。この場所は自由に選択できます。ユーザー zen-svc がその場所に対して読み取り、書き込み、および実行権限を持っていることを確認してください。 共有記憶域に 2 つのグループと 1 人のユーザーを作成し、各クラスター ノードがデータベース ファイルにアクセスできるようにします。 • zen-data グループおよび zen-adm グループはそれぞれ、クラスター ノード上の zen-data グループ ID および zen-adm グループ ID と一致している必要があります。 • ユーザー zen-svc は、クラスター ノード上の zen-svc ユーザー ID と一致している必要があります。
共有記憶域マウント用のディレクトリを作成する	クラスター ノードごとに、ユーザー zen-svc としてログインし、共有記憶域へマウントされるディレクトリを作成します。ユーザー zen-svc にはパスワードがありません。su コマンドを使用した root アカウントによるアクセスのみを行うことができます。ディレクトリ名は自由に選択できます。
Heartbeat サーバーを構成する	Zen データベース エンジンを制御する*ノードごとに* Heartbeat サーバーを構成します。次のような構成を行います。 • ノード。クラスターに参加させるすべてのノードを追加します。 • 認証。ノード間のネットワーク通信に使用する認証のタイプを指定します。 • 媒体。Heartbeat がノード間の内部的な通信に使用する方法を指定します。 • スタートアップ。Heartbeat サーバーの起動のタイミングについての設定を指定します。これを**オン**に設定すると、Heartbeat サーバーはブート時に起動するようになります。
Heartbeat ユーザーのパスワードを割り当てる	Linux Heartbeat では Heartbeat Management Client へログインする際のデフォルトのユーザー名として hacluster を提供します。Heartbeat Management Client を実行する*ノードごとに*、ユーザー hacluster のパスワードを割り当てます。
Zen のリソース グループを追加する	root としてログインし、クラスター ノードの 1 つで Heartbeat Management Client を開始します。ユーザー hacluster としてログインし、新しいグループを追加します。［ID］には、Zen グループの名前を指定します。［Ordered］と［Collocated］には true を設定します。

操作	説明
リソースをグループに追加する	次の 3 つのリソースを Zen グループに追加します。 • IPaddr（IP アドレス） • Filesystem（ファイルシステム） • Zen（OCF リソース エージェント） IPaddr（IP アドレス） Heartbeat Management Client で、新しい native 項目を追加します。［Belong to group］には、Zen 用に追加したグループを選択します。［Type］には IPaddr を選択します。 追加したリソースで、IP の値に対してクラスターの IP アドレスを指定します。Linux Heartbeat がインストールされ構成されたときには、（ノードではなく）クラスターに割り当てられた IP アドレスを使用します。 Filesystem（ファイルシステム） 新しい native 項目をもう 1 つ追加します。［Belong to group］には、Zen 用に追加したグループを選択します。 ［Type］には Filesystem を選択し、fstype というパラメーターを削除します。新しいパラメーターを追加し、［Name］には "device" を選択します。［Value］には、共有記憶域のデバイス名、コロン、共有マウントの場所を指定します。 新しいパラメーターをもう 1 つ追加し、［Name］には "directory" を選択します。［Value］には、NFS マウントで使用するディレクトリを指定します。 Zen（OCF リソース エージェント） 新しい native 項目をもう 1 つ追加します。［Belong to group］には、Zen 用に追加したグループを選択します。［Type］では、［Description］フィールドに "Zen OCF Resource Agent" の記載がある zen-svc エントリをクリックします。この他に設定は必要ありません。
マウントされた共有記憶域上にサブディレクトリを作成する	Filesystem リソースを追加できた場合は、クラスター サーバーと共有記憶域間にマウントが存在するようになります。クラスター ノードの 1 つに、ユーザー zen-svc としてログインします。共有記憶域マウントで、"log" と "etc" という名前のディレクトリを作成します。 たとえば、マウント ディレクトリが /usr/local/actianzen/shared の場合、/usr/local/actianzen/shared/log と usr/local/actianzen/shared/etc というディレクトリを追加します。
ZenCC でクラスター サーバーを構成する	それぞれのクラスター ノードで、ZenCC を使ったクラスター サーバーを構成する必要があります。 ZenCC を実行するノード以外のすべてのクラスター ノードを代替モードにします。zen-svc ユーザーとして、1 つのアクティブ ノードで、あるいはアクティブ ノードにアクセスできるクライアントから ZenCC を実行します。 Zen エクスプローラーで、新しいサーバーを追加し、クラスターの名前（または IP アドレス）を指定します。 追加したサーバーのプロパティにアクセスします。ログインを指示された場合は、admin ユーザーとしてログインします。パスワードは空のままにします。［**ディレクトリ**］プロパティにアクセスします。［**トランザクション ログのディレクトリ**］には、"log" ロケーション用に作成したディレクトリを指定します。［DBNames **設定ファイルのディレクトリ**］には、"etc" ロケーション用に作成したディレクトリを指定します。「マウントされた共有記憶域上にサブディレクトリを作成する」を参照してください。 ZenCC で新しいサーバーを追加し、そのプロパティをほかの各クラスター ノードから設定します。ZenCC を実行するノード以外のすべてのノードを代替モードにします。

操作	説明
共有記憶域上にデータベースを作成する	クラスター ノードのうち 1 つのノードのオペレーティング システムから、ユーザー zen-svc としてログオンし、データベースを置くファイル システム共有下にディレクトリを作成します（ユーザー root としてディレクトリを作成する場合、ユーザー zen-svc はそのディレクトリに対して読み取り、書き込み、および実行権限を持っていることを確認してください）。 ZenCC を実行するノード以外のすべてのクラスター ノードを代替モードにします。 zen-svc ユーザーとして、1 つのアクティブ ノードで、あるいはアクティブ ノードにアクセスできるクライアントから ZenCC を実行します。「ZenCC でクラスター サーバーを構成する」で追加したサーバーに新しいデータベースを作成します。［**場所**］には、データベースを置くディレクトリを指定します。必要に応じてその他のデータベース オプションを指定します。 新しいデータベースの場合、必要であればテーブルを作成します。
各ノードからのデータベースへのアクセスを確認する	各クラスター ノードは共有記憶域上の Zen データベースにアクセスできる必要があります。データベースを作成したクラスター ノードを代替モードにします。これは zen-svc リソース（データベース エンジン）を実行しているノードです。 クラスターの次のノードへフェール オーバーします。次のノードが、zen-svc リソースの実行の制御を受け取ることを確認してください。クラスター内の各ノードに対して、上記の手順（代替、フェールオーバー、確認）を繰り返し、開始したノードまで戻ったら終了します。

クラスター環境における Zen の管理

フェールオーバー クラスター環境に Zen をインストールすると、リソースとして管理できます。ここでは、一般的な管理の内容について説明します。

- 「Zen ライセンスおよびノードのメンテナンス」
- 「Zen 停止時の動作」
- 「Actian Zen エンジン サービスの停止と再起動」
- 「Zen 設定の変更」
- 「ソフトウェアのパッチ」

Zen ライセンスおよびノードのメンテナンス

通常の Zen ライセンスやマシンのメンテナンスに関する手順は、フェールオーバー クラスター環境のノードにも同様に適用されます。データベース エンジンがインストールされた物理マシンまたは仮想マシンの構成を変更する前には Zen 製品キーを認証解除してください。構成の変更が完了したら再度認証してください。

『*Zen User's Guide*』の「キーを認証解除するには」および「キーを認証するには」を参照してください。

Zen 停止時の動作

クラスター ノードでエラーが発生した場合、Zen クライアントはスタンバイ ノード上の Zen エンジンに自動的に再接続しません。アプリケーションでクライアントを Zen データベースに再接続するか、アプリケーションを再起動する必要があります。データベース エンジンで［自動再接続の有効化］がオンに設定されていたとしても同様の対処が必要です。

トランザクション一貫性保持がオフで、トランザクションが完了する前にエラーが発生した場合には、自動的にトランザクションを開始する前の状態にロール バックされます。つまり、すぐ前の完了済みチェック ポイントへ戻るということです。ロール バックは、アクティブ サーバーがデータ ファイルへのアクセスを要求した時点で発生します。

トランザクション一貫性保持を設定している場合、クラスター ノードに障害が発生した時点と最後のチェック ポイントの時点との間に発生した完了済みの変更は回復させることができます。トランザクション一貫性保持はすべてのノードで同じように構成される必要があります。また、トランザクション ログは共有記憶域に置かれます。ただし、トランザクション一貫性保持が有効だった場合でも、クラスターの障害が発生した時点で完了していないトランザクションは失われます。

Actian Zen エンジン サービスの停止と再起動

オペレーティング システムから Zen データベース エンジン サービスを手動で停止すると、アクティブ ノードからクラスターのフェールオーバーが発生します。サービス ノードのメンテナンスを実行しており、そのようなフェールオーバーを回避したい場合は、サービスをクラスター ユーティリティから停止します。

Zen 設定の変更

変更する設定によってはデータベース エンジンを再起動する必要があります。「設定リファレンス」を参照してください。

▶▶ 設定の変更を適用するために Zen データベース エンジン サービスを停止し、再起動するには

Windows クラスター アドミニストレーターで、次の手順を記載されている順序で実行します。

1 Actian Zen エンジン サービスを右クリックして、[**このリソースをオフラインにする**] をクリックします。

2 Actian Zen エンジン サービスを右クリックして、[**このリソースをオンラインにする**] をクリックします。

ソフトウェアのパッチ

ある時点で、Zen またはフェールオーバー クラスター ソフトウェアにパッチを適用する必要があるかもしれません。この実行のサポートが必要な場合は、弊社テクニカル サポートまでお問い合わせください。

マイグレーション

マイグレーションでは、Zen を実行している仮想マシンを、ある物理ホストから別のホストへ移動します。通常は、仮想マシンのメモリ、記憶域およびネットワーク接続が移行先に移行されます。ハイパーバイザーによって、マイグレーションは " ライブ " マイグレーションまたは " ホット " マイグレーションと呼ばれます。

" ライブ " マイグレーションまたは " ホット " マイグレーションでは、クライアントから Zen への接続がそのまま維持されます。これにより、ハードウェア バランスまたはリソース バランスの変更が可能です。" コールド " マイグレーションでは、仮想マシンを一時停止する必要があるため、ネットワーク接続が中断されます。クライアントから Zen への接続を再確立する必要があります。

マイグレーション環境には、実行中の Zen のインスタンスが 1 つだけあります。これにより、ホスト マシンがクラッシュした、またはホスト マシンを直ちにオフラインにする必要がある場合、その環境はやや脆弱になります。また、共有記憶域にエラーが生じた場合、データベース エンジンは物理記憶域に対する読み込みまたは書き込みを処理することができません。ハイパーバイザーによっては、共有記憶域を使用しないマイグレーション ソリューションを提供しています。

ホスト名が仮想マシンの移行後も同じままであれば、Zen は正常に動作し続けます。製品キーの状態は " アクティブ " のままです。

マイグレーション環境で Zen をインストールまたは構成する際、特別な手順は必要ありません。ハイパーバイザーのドキュメントを参照してください。

フォールト トレランス

フォールト トレラント環境はマイグレーション環境と似ていますが、さらにコンポーネントがエラーになっても動作が中断されないようする機能も備えています。フォールト トレラント環境では、ネットワーク接続、継続的なサービスおよび同期された共有記憶域を介したデータ アクセスが確保されます。コンポーネントの切り替えが起こっても、クライアント マシンおよびアプリケーションは正常に機能し続け、データベース エンジンは中断されません。

フォールト トレラント環境で Zen をインストールまたは構成する際、特別な手順は必要ありません。ハイパーバイザーのドキュメントを参照してください。

障害回復

障害回復にはデータの回復やサイトの回復が含まれています。データの回復はデータを保護および復元する方法です。サイトの回復は、データを含むサイト全体を保護および復元する方法です。

データの回復は、ハイパーバイザーの共有記憶域および Zen のトランザクション ログとトランザクション一貫性保持によって容易に行うことができます。「トランザクション ログおよびトランザクション一貫性保持」を参照してください。Zen Enterprise Server および Cloud Server ではトランザクション ログとトランザクション一貫性保持を使用できます。

サイトの回復は物理マシンと仮想マシンのどちらでも実行できます。回復されたサイトでホスト名が同じままであれば、Zen は正常に動作します。これは一般に仮想マシンの場合です。物理マシンを回復する場合、回復したサイトでホスト名が回復前と異なっていると、Zen の起動時に Zen 製品キーの状態が検証失敗に変わります。Zen は検証失敗の状態でもしばらくの間は正常に動作し続けます。その間にキーを修復するか、元のサイトへ戻すことができます。

障害回復環境で Zen をインストールまたは構成する際、特別な手順は必要ありません。ハイパーバイザーのドキュメントを参照してください。

Zen とハイパーバイザー製品

11

さまざまなハイパーバイザー製品と共に Zen を使用するためのヒント

以下のトピックにより、ハイパーバイザー製品を用いて Zen を最も効果的に使用することができます。

ハイパーバイザー製品のインストール

ほかの複雑なソフトウェアと同様に、ハイパーバイザー製品は非標準および特殊な設定を用いるなど、さまざまな方法でインストールすることができます。Zen との互換性のため、製品ベンダーが推奨する最善手法を使用してハイパーバイザー製品をインストールおよび設定してください。

Zen の使用法のヒント

ここでは、Zen の使用に関する以下の項目について説明します。

- 「物理マシンから仮想マシンへの移行」
- 「設定」
- 「仮想マシン リソース プールおよびテンプレート」
- 「フェールオーバー クラスター サポート」
- 「パフォーマンス」
- 「データのバックアップ」

物理マシンから仮想マシンへの移行

最初は物理マシンンに Zen Enterprise Server、Cloud Server、または Workgroup をインストールしていたものを、後にビジネスのニーズの変化に応じて仮想マシンに移行することができます。物理マシンから仮想マシンへ移行する場合は、必ずホスト名を移行前と同じ名前にしてください。

Zen エンジンは IP アドレスに依存しませんが、仮想マシン自体は依存する可能性があります。お使いの仮想マシンが DNS（Domain Name System）ではなく、生の IP アドレス、またはホスト ファイルに依存する場合は、必ず IP アドレスに関して適切な対応を行ってください。

設定

ライブ マイグレーション、フォールト トレランス、高可用性、準仮想化、リソースのスケジューリングおよび障害回復などのハイパーバイザー製品機能を使用するために、Zen を特別な手順で設定する必要はありません。ホスト名に一貫性があれば、Zen は認証され、完全に機能し続けます。

障害回復など特定の状況ではネットワークやハードウェアの変更が必要となることもあります。以下の項目を変更しても Zen には悪影響を与えません。

- 仮想マシンの IP アドレスまたは MAC アドレス
- 仮想マシンのハードウェア（CPU の種類、CPU の速度、メモリの量および記憶域の種類やサイズなど）

データベース エンジンが起動している場合、Zen はメモリの増加や物理ストレージなどハードウェアの特定の変更を認識しないので注意してください。このような変更を認識させる場合は、データベース エンジンを停止して再起動する必要があります。

仮想マシン リソース プールおよびテンプレート

Zen は仮想マシン リソース プールおよびテンプレートで使用することができます。両方を使用する場合は、それぞれの Zen で独自の製品キーが必要となります。『*Zen User's Guide*』の「ライセンスの適用」を参照してください。

リソース プール

リソースプール内の、データベース エンジンを含む各仮想マシンで Zen が認証される必要があります。

テンプレート

テンプレートから作成した仮想マシンで Zen を認証するには、構成スクリプトを使用することができます。このスクリプトは、ゲスト オペレーティング システムのカスタマイズ時に、Zen キーを認証するための CLI License Administrator ツールを起動することができます。『*Zen User's Guide*』の 「License Administrator のコマンド ライン インターフェイス」を参照してください。

ゲスト オペレーティング システムのその他のプロパティ（ホスト名など）をカスタマイズすることを忘れないでください。それらのプロパティは Zen の実行には依存しません。

フェールオーバー クラスター サポート

一般的なガイドラインとして、共通（アフィニティ）規則を使用する場合は、必ずすべてのコアを同じソケットで実行するようにしてください。Zen はマルチコアをサポートするため、これがパフォーマンスの向上に役立ちます。ご自身の構成に応じて非共通規則を使用することもできます。

MSCS 構成用のデータ デバイスとして RDM（Raw Device Mapping）を使用する場合は考慮すべき事項があります。RDM については、ベンダーのドキュメントを参照してください。

DRS（Distributed Resource Scheduler）でフォールト トレランス / 高い可用性を使用する場合、ロード バランスはフェールオーバー後にのみ行うことができることに留意してください。DRS については、ベンダーのドキュメントを参照してください。

パフォーマンス

Zen で最高のパフォーマンスを実現するには、以下の事項を順守してください。

- ハイパーバイザー ベンダーからのパフォーマンス最良実施例に従うこと。
- Zen をホストする仮想マシンに十分なメモリ（RAM）が確保されていること。
- ハイパーバイザー ホストには十分な数の仮想 CPU があり、同じマシン上の全仮想マシンの中で仮想 CPU の競合を最小化します。これにより、Zen を実行する仮想マシンとの競合が避けられます。共通（アフィニティ）規則を使用する場合は、必ずすべてのコアを同じソケットで実行するようにしてください。
- Zen データ ファイルは高速な物理ストレージに存在し、物理ストレージ デバイスに対するスピンドルの競合およびコントローラーの競合を最小化します。

データのバックアップ

「Backup Agent および VSS Writer によるデータ バックアップ」を参照してください。

ワークグループ エンジンの詳細

12

ワークグループ エンジンの技術的な詳細および高度な処理

このトピックでは、ワークグループ エンジンを最大限に活用する方法について説明します。

- 「ネットワーク」
- 「サーバーとワークグループの技術的な相違」
- 「ワークグループの問題のトラブルシューティング」
- 「リダイレクト ロケーター ファイルの作成」

ネットワーク

サーバーおよびワークグループ製品には同一のネットワーク コンポーネントが含まれています。したがって、ワークグループ エンジンを Zen サーバー エンジンにアップグレードすることが可能です。クライアント側のリクエスターはいずれのタイプのエンジンにも接続することができます。

クライアント側の MicroKernel ルーターは、完全に確立されたアルゴリズムに従ってファイルのゲートウェイ オーナーシップを見つけます。

表 27　ゲートウェイ検索の優先順位

優先順位	手順
1	データ ファイルと同一のコンピューター上でデータベース エンジンに接続を試みます。
2	ローカル エンジン（クライアント マシン上の）を使用してデータ ファイルのオープンを試みます。
3	ファイルを所有するゲートウェイ エンジンを見つけて接続を試みます。

クライアント側のルーターが最初に試みることは、データと同じコンピューター上のエンジンに接続することです。このため、データが存在する場所でエンジンが実行されていることは常に非常に効果的です。

Zen ネットワーク サービス レイヤーは、リモート データベース エンジンを検索し接続するために多種多様な方法を用いるため、データベース エンジンが実行されていないファイル サーバー上のファイルを最初に開こうとするときに時間がかかることがあります。「ゲートウェイ一貫性保持」をオンにすると、ルーターがエンジンの位置を特定するのに失敗した各マシンを記憶するので、その後その接続は試行されません。

リモート ファイル サーバー上のエンジンに接続できなかった場合、ルーターはローカル エンジンがリモート ファイルを開くことを許可します。ローカル エンジンはまず新しいロケーター ファイルを作成し、リモート ディレクトリのオーナーシップをとることを試みます。既にほかの MicroKernel がそのディレクトリを所有している場合、ローカル エンジンはルーターにステータス コード 116 を返します。

最後に、ルーターはゲートウェイ コンピューターを探そうとします。ロケーター ファイルを開き、ゲートウェイ エンジンの名前を読み取ります。それからエンジンにリクエストを送ります。ルーターは MicroKernel からステータス コード 116 を受け取らなければロケーター ファイルを読み取らないことに注意してください。したがって、ゲートウェイ機能を使用するためには、ローカル ワークグループ エンジンがインストールされている必要があります。ローカル エンジンがないためにローカル エンジンを使用してリモート ファイルを開くことに失敗した場合、ルーターはロケーター ファイルを読み取らず、ゲートウェイ エンジンを見つけることができません。

サーバーとワークグループの技術的な相違

Zen サーバー エンジンとワークグループ エンジンにはいくつかの重要な相違点があります。

プラットフォーム

Zen サーバー エンジンは、さまざまなプラットフォーム用の 64 ビット版および 32 ビット版を提供します。ワークグループ エンジンの場合は 32 ビット Windows 版のみです。ワークグループ エンジンを 64 ビット Windows オペレーティング システムで実行することはできますが、アドレス領域が 4 GB に制限されているため、一般的に 64 ビット システムでインストールされる（2 GB を超える）大量のメモリを活用することはできません。ワークグループ エンジンを 32 ビット オペレーティング システムにインストールする場合、アドレス領域の上限は 2 GB です。

ユーザー インターフェイス

Windows の場合、Zen サーバー エンジンは常に Windows のサービスとして実行するようにインストールされます。ワークグループ エンジンは、アプリケーションまたはサービスのどちらとしてインストールするかを設定できます。新規インストールの場合、デフォルトでは、サービスとしてインストールされますアプリケーションとして実行するようにインストールした場合は、タスクバーの通知領域にあるアイコンをインターフェイスとして使用します。『*Getting Started with Zen*』の 「Workgroup エンジンのセットアップ」を参照してください。

認証と Btrieve セキュリティ ポリシー

Zen データベース エンジンは、オペレーティング システムのユーザー認証に基づいて、オペレーティング システムにおけるファイルのアクセス許可を適用します。ワークグループ エンジンは自身でユーザーの認証を行いません。ワークグループ エンジンは、ネットワーク上のシステムに接続することができれば、そのデータにアクセスできます。緩和度の高いワークグループ セキュリティは、セキュリティよりも使いやすさを優先する小規模のオフィスを対象としています。

ワークグループ エンジン用のオペレーティング システム認証がないということは、Btrieve の " 混合 " セキュリティ ポリシーと " クラシック " セキュリティ ポリシーが同じであることを意味します。セキュリティ ポリシーの違いは、Zen サーバー エンジンとワークグループ エンジン間の動作の違いです。詳細については、「Zen セキュリティ」を参照してください。

ゲートウェイのサポート

ワークグループ エンジンは、ローカル、リモート共にファイルを開いたすべての場所にロケーター ファイルを作成することにより、ワークグループ エンジンがゲートウェイのオーナーシップを絶えず動的に適合できるようにします。デフォルトで、ワークグループ エンジンはほかのコンピューターやネットワーク デバイスで認証されるユーザー ID でも実行されます。このため、ワークグループ エンジンはゲートウェイ環境での使用に最適です。『*Getting Started with Zen*』の「ゲートウェイ構成のセットアップ」を参照してください。

Zen サーバー エンジンは必ずしもゲートウェイ ロケーター ファイルを作成する、または引き受けるわけではありません。このため、サーバー エンジンをゲートウェイ環境で使用するような設計やテストを行っていません。そのため、ワークグループ環境でワークグループ エンジンに代わってサーバー エンジンをゲートウェイとすることはサポートしていません。

非同期 I/O

Windows 版の Zen サーバーは非同期 I/O を使用します。また、データベース ページの書き込みの結合は Zen サーバーでのみ行われます。これらの機能によって、I/O の負荷が高いときは、ワークグループ エンジンに比べサーバー エンジンが大きなパフォーマンスの利点を得ることができます。

デフォルトの設定

いくつかのデータベース設定（キャッシュ サイズやシステム キャッシュなど）のデフォルト値は、Zen サーバー エンジンとワークグループ エンジンでそれぞれ異なります。ワークグループ エンジン設定用のデフォルト値は、システム リソースの消費をより抑えるよう設定されます。「設定リファレンス」を参照してください。

ライセンス モデル

Zen Enterprise Server および Zen Workgroup ではユーザー カウント ライセンス モデルを使用します。Zen Cloud Server では容量ベース ライセンス モデルを使用します。『*Zen User's Guide*』の「ライセンス モデル」を参照してください。

ワークグループの問題のトラブルシューティング

このセクションでは、ワークグループ環境での問題の解決に役立つヒントを提供します。

最初の接続での待ち時間

最初のファイル オープン要求が発行されるときに何度も遅延が起きるのであれば、次の方法が役に立つかどうか調べてください。

可能であれば、データが存在するシステムでエンジンが実行されるようにしてください。

ファイル オープン要求をどこに送るかを決定するクライアントにとって、データと同じマシン上のエンジンに接続することが最優先事項です。ワークグループ エンジンがアプリケーションとして実行されるのを確実にするため、次のコマンドを使用してエンジンのアイコンをスタートアップ フォルダーに入れます。

```
zenengnapp.exe
```

もう 1 つの方法としては、ワークグループ エンジンをサービスとしてインストールします。詳細については、『*Getting Started with Zen*』を参照してください。また、Zen ファイルのデフォルトの場所については、「ファイルはどこにインストールされますか？」を参照してください。

ゲートウェイ トポロジを実行している場合

データが存在する場所でエンジンを実行できない場合、最初の接続の待ち時間はより重要な問題になります。試してみることがいくつかあります。

1 クライアントの設定で［サポート プロトコル］を減らし、ネットワークで使用されていないプロトコルの使用が試みられないようにします。

2 ゲートウェイ一貫性保持を使用します。ゲートウェイ一貫性保持はクライアントの設定で、ゲートウェイ環境で最初の接続を確立するときの待ち時間をほとんどなくすことができます。［ゲートウェイ一貫性保持］がオンに設定されていると、クライアント ルーターがエンジンの実行されていないコンピューター名をレジストリに書き込むようにします。接続に失敗すると、ルーターはこのサーバー名を実行中のみ保存するのではなく、レジストリに保存します。次回アプリケーションが起動したとき、データのある場所のエンジンへの接続は試行されません。直ちに現在のゲートウェイを決定する次の段階に進みます。

   この設定は ZenCC で切り替えることができます。ZenCC の Zen エクスプローラーで［ローカル クライアント］ノードを展開し、［MicroKernel ルーター］を右クリックします。［**プロパティ**］をクリックし、［**アクセス**］を選択します。［**ゲートウェイ一貫性保持**］オプションをクリックしてオンに設定し（チェック マークは、その設定が「オン」であることを表します）、［OK］をクリックします。

メモ この機能はトポロジを固定するため、デフォルトではオフになっています。データが存在する場所にサーバー エンジンまたはワークグループ エンジンを追加した場合、この設定をオンにした各クライアントでオフに戻す必要があります。この設定をオフに戻すと、エンジンが実行されていないコンピューターのレジストリを消去するので、その後すぐにオンにすると新しいトポロジに基づいた新しいリストが生成されます。

ステータス コード 116

ステータス 116 は MicroKernel のステータス コードで、ゲートウェイとして動作する別の MicroKernel エンジンによってファイルが使用されていることを示します。アプリケーションがステータス コード 116 を受け取った場合、MicroKernel ルーターはロケーター ファイルを読むことができたけれどもゲートウェイ コンピューター上で実行されているエンジンと交信できなかったことを示します。

最初にしなければならないことは、ゲートウェイがどれかを見極めることです。この作業は Gateway Locator ツールを使用して行うことができます。

次に Zen System Analyzer（ZenSA）ネットワーク テストを使用してそのコンピューターに接続してみます。ZenSA は問題を分離するための役に立つ情報を提供します。

これが起こる 1 つの状況は、2 つのコンピューターがルーターで分けられていて、両方ともファイル サーバーを見ることができるが、互いを見ることができない場合です。

リダイレクト ロケーター ファイルの作成

ゲートウェイ エンジン操作のこの機能は、複数ディレクトリのデータベースのトランザクション アトミシティを保証し、また複数のデータ ディレクトリにわたるゲートウェイ エンジンの名前の変更を容易にします。

Zen クライアントがリモート データ ファイルにアクセスする際に次の方法を使用することを思い出してください。

1 データ ファイルと同一のコンピューター上でデータベース エンジンに接続を試みます。

2 次に、そのリモート マシンでデータベース エンジンが使用可能でない場合、ローカル エンジンを使用してリモート ディレクトリのオーナーシップをとり、ロケーター ファイルを作成します。ゲートウェイ ロケーター ファイルが既に存在する場合、ローカル エンジンは使用されません。

3 3 番目に、指定されたゲートウェイ エンジンを使用することを試みます。

ゲートウェイ設定は、データ ファイルが存在するコンピューター上に使用可能なデータベース エンジンがない場合にのみ有効になることを覚えておいてください。

この機能により、同一ボリューム上の複数ディレクトリのデータベースのトランザクションの一貫性を保持しながら動的（変動）ゲートウェイ エンジンを使用することができます。この利点は、別のゲートウェイ ロケーター ファイルを指示する新しいゲートウェイ ロケーター ファイルによってもたらされました。新しいタイプは**リダイレクト ロケーター ファイル**と呼びます。ディレクトリ D のロケーター ファイルを指示するリダイレクト ロケーター ファイルをディレクトリ A、B、C に持つことにより、ディレクトリ D のロケーター ファイルが指定するゲートウェイ エンジンが、ほかのディレクトリにあるデータ ファイルのサービスも確実に提供することができるようになります。

ディレクトリ D のロケーター ファイルが固定ゲートウェイを指定しているか、これらのファイルを最初に開いたエンジンによって動的に作成されるかにかかわらず、このアーキテクチャは指定されたすべてのディレクトリが必ず同一のゲートウェイ エンジンを使用します。同様に、いくつかのディレクトリの固定ゲートウェイを変更しようとする場合、リダイレクト ロケーター ファイルを使用すると、すべてのロケーター ファイルではなく、たった 1 つのロケーター ファイルを変更するだけで済みます。したがって、一定のハード ドライブ上のすべてのデータ ファイルが固定ゲートウェイを指定しているといないにかかわらず、同一ゲートウェイ エンジンを使用するように指定することができます。

リダイレクト ロケーター ファイルの要件

リダイレクト ロケーター ファイルの最初の行は "=>" で始め、その後に必ず同一ドライブ上にある別のロケーター ファイルのパスを指定する必要があります。パス名にはスラッシュおよび円記号の任意の組み合わせを使用することができます。スラッシュはすべてローカル オペレーティング システムの使用するタイプの区切り文字に変換されます。

スラッシュで終わるパスを指定した場合、データベース エンジンはパスの最後にデフォルトのロケーター ファイル名（~PVSW~.LOC）を追加します。指定したパスがスラッシュで終わっていない場合、データベース エンジンはパスにファイル名が含まれているものと見なします。

次の表は、リダイレクト ロケーター ファイルのパスの指定方法の一覧です。

表 28　リダイレクト ロケーター ファイル パスの記述方法

パス	意味
=>¥*path_name*	現在のロケーター ファイルが格納されているルート ドライブからパスを指定します。
=>.¥*path_name*	現在のディレクトリからの相対パスを指定します。
=>..¥*path_name*	現在のディレクトリの親ディレクトリからの相対パスを指定します。

これらのロケーター ファイルに複数レベルのリダイレクトを割り当てることができます。たとえば、最初のロケーター ファイルが 2 番目のロケーター ファイルを指示し、2 番目のロケーター ファイルが 3 番目のロケーター ファイルを指示するという具合です。各ワークグループ エンジンはそれぞれのロケーター ファイルを順に開き、実際のゲートウェイ名を探します。"=>" で始まらないロケーター ファイルを見つけると検索は中止されます。エンジンはこのロケーター ファイルがゲートウェイ エンジンを指定しているものと見なします。

リダイレクト ロケーター ファイルの作成

ほかのロケーター ファイルと同様、リダイレクト ロケーター ファイルは普通のテキスト ファイルです。リダイレクト ロケーター ファイルは手作業でもプログラムからでも作成することができます。リダイレクト ロケーター ファイルには読み取り専用フラグを設定しておかないと、そのディレクトリにあるデータ ファイルに最初にアクセスを試みたエンジンによって上書きされてしまいます。

▶▶ リダイレクト ロケーター ファイルを作成するには

1 テキスト エディターを開き、新規テキスト ファイルを作成します。

2 完了時にファイルをどこに保存するかを決定します。別のロケーター ファイルにリダイレクトしようとするデータ ファイルと同じディレクトリにファイルを保存します。

たとえば、C:¥data にあるデータ ファイルが、確実にほかのデータ ファイルと同じゲートウェイ エンジンによってアクセスされるようにするには、C:¥data を忘れないようにしておきます。

3 => と次のロケーター ファイルのパス名を入力します。前の手順の例で続けると、C:¥data にある現在のデータ ファイルが、C:¥moredata にあるロケーター ファイルで指定されたゲートウェイ エンジンに属するようにしたい場合、次のように入力します。

```
=>..¥moredata¥ （推奨）または
=>¥moredata¥ （推奨しません）
```

最初のケースでは現在のディレクトリからの相対パスを指定しています。2 番目のケースは現在のドライブからの絶対パスを指定しています。この例の場合では、両方とも同じ目的のディレクトリを解決することができます。

メモ リダイレクト ロケーター ファイルでは相対パス（.¥ または ..¥ で始まる）を使用すること、また、同じデータにアクセスするすべてのワークステーションで同じ共有名を使用することを強くお勧めします。この 2 つの忠告に従えば、割り当てられたドライブでのネットワーク パス名の解決で起こるエラーを防ぐことができます。

4 ゲートウェイ エンジンを構成したいデータ ファイル ディレクトリに ~PVSW~.LOC という名前でファイルを保存します。

5 テキスト エディターを閉じます。

6 このテキスト ファイルに読み取り専用フラグを設定します。

Windows でファイルを読み取り専用にするには、Windows エクスプローラーでファイル アイコンを右クリックして［プロパティ］ダイアログ ボックスを使用するか、［プログラム］の DOS セッションかプログラムで ATTRIB コマンドを使用します。

```
attrib +r ~pvsw~.loc
```

▶▶ 固定ゲートウェイで多数のデータ ディレクトリの同期をとるには

1 手作業または Gateway Locator プログラムを使用して、リダイレクトしない読み取り専用の（固定）ロケーター ファイルを作成します。ゲートウェイとして使用するワークグループ エンジンを指定する必要があります。

たとえば、ロケーター ファイルに workgroup1 というコンピューター名をゲートウェイ エンジンとして指定し、ファイルが C:¥data¥db1 にあるとします。

2 前の手順で指定したゲートウェイ エンジンを使用しようとする、その他のデータ ディレクトリごとに、リダイレクト ロケーター ファイルを作成する必要があります。各リダイレクト ロケーター ファイルは前の手順で作成したファイルを示す必要があります。

前の例でいうと、C:¥data¥db2 および C:¥data¥db3 にあるリダイレクト ロケーター ファイルは次のテキストを含みます。

```
=>..¥db1¥
```

これにより、このファイルを読み取ったエンジンはすべて相対パスに従って、指定された C:¥data¥db1 で別のロケーター ファイルを検索します。この場合、指定したディレクトリには workgroup1 をゲートウェイ コンピューターとして挙げているロケーター ファイルがあります。

▶▶ 動的ゲートウェイで多数のデータ ディレクトリの同期をとるには

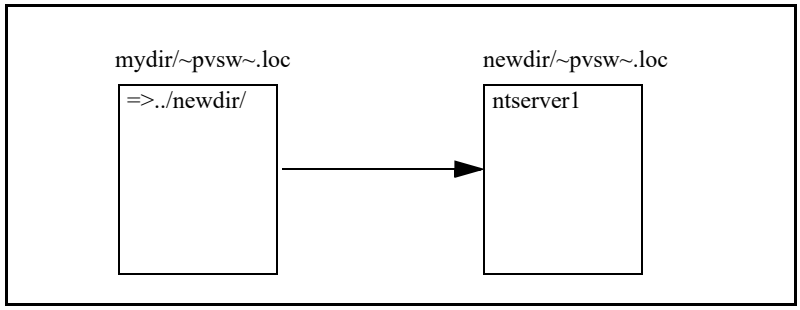
1 上記の手順に従いますが、手順 1 でロケーター ファイルが固定的に割り当てられないように書き込み可能にするだけです。

この場合、リダイレクト階層内にあるデータ ファイルにどのエンジンもアクセスしていなければ、目的のディレクトリにはロケーター ファイルがないということを覚えておいてください。この状態は正常です。動的ロケーター ファイルはエンジンの最初のデータ アクセスによって各セッションに作成され、最後のユーザー セッションが終了したときに削除されます。ロケーター ファイルが存在しないデータ ディレクトリを示すリダイレクト ロケーター ファイルを持つことは問題ありません。この場合、データ ファイルを最初に開いたエンジンがロケーター ファイルを作成します。

例

図 4 に示したロケーター ファイルの例を使用すると、左側のリダイレクト ロケーター ファイルはデータベース エンジンに 1 つ上のディレクトリに行き、そのサブディレクトリの newdir でデフォルト名（~PVSW~.LOC）を持つ別のロケーター ファイルを探すように指示しています。今度は、このロケーター ファイルが ntserver1 という名前のコンピューター上にあるワークグループ エンジンがゲートウェイ エンジンであることを指定します。その結果、mydir ディレクトリにあるデータ ファイルにアクセスするために、ntserver1 にあるデータベース エンジンが使用されます。

図 4 リダイレクト ロケーター ファイルの例



監視

13

Zen 環境を監視する

Zen は使いやすく、また最小限の管理で利用できるようになっています。しかし、Zen 環境のさまざまな状況やリソースについて監視することもできます。たとえば、アプリケーションの一部を調整したり、データベースに関するさまざまなパフォーマンス面を監視したりする必要があるかもしれません。あるいは、ビジネス ニーズに応じて Zen の各種設定を調整しており、デフォルト設定からの変更を監視することもできます。

以下のトピックで監視について説明します。

- 「データベースの状態の監視」
- 「データ ファイルの断片化の監視」
- 「パフォーマンス カウンターの監視」
- 「ライセンスの使用状況の監視」
- 「データベース アクセスの監視」
- 「メッセージ ログの見直し」
- 「メッセージの電子メール通知を受け取る」

データベースの状態の監視

Monitor ツールではデータベースの状態を監視することができます。次の項目について監視が可能です。

- アクティブ ファイル
- MicroKernel セッション
- リソース使用状況
- MicroKernel 通信統計情報
- SQL アクティブ セッション

Monitor を初めて使用する場合、またはこのツールの概要を知りたい場合は、「Monitor の概要」を参照してください。

Monitor の概要

Monitor はデータベース エンジンの特定の動作と属性を体系的に監視することができるツールです。このツールでは、データベース管理およびアプリケーションのプログラムの診断に役立つ情報を提供します。これは MicroKernel エンジンおよびリレーショナル エンジンのどちらの状況も監視できます。

Monitor には次の 2 つのインターフェイスがあり、いずれも同じ機能を提供します。

- 「Monitor のグラフィカル インターフェイス」は、一連のタブ形式で情報を表します。
- 「Monitor のコマンド ライン インターフェイス」では、選択した場所に情報を表すように指示する実行可能プログラムを使用します。

Monitor のグラフィカル インターフェイス

ZenCC には、情報を一連のタブ形式で構成して表す Monior ツールが含まれています。タブの配置は必要に応じて変えることができます。タブ内のデータ列も配置を変えたり並べ替えたりすることができます。ある特定の時点のスナップショットを表示し、その情報は手動または自動でリフレッシュすることができます。

GUI Monitor の使用

ZenCC 内で、Zen エクスプローラーから Minitor にアクセスできます。

▶▶ データベース エンジン用の Monitor にアクセスするには

Monitor ウィンドウを開くには、以下のオプションのうちの 1 つを使用します。同時に複数のエンジンの監視を行うこともできます。

- Zen エクスプローラーで［エンジン］ノードを展開します。監視対象のデータベース エンジンを右クリックし、コンテキスト メニューから［Server Monitor］を選択します。
- ［ツール］>［Monitor］をクリックします。監視するサーバーを選択できるダイアログが表示されます。
- Monitor を、サードパーティ アプリケーション向けなど、スタンドアロン ウィンドウで起動するには、プロンプトで -monitor_any パラメーターを使用できます。

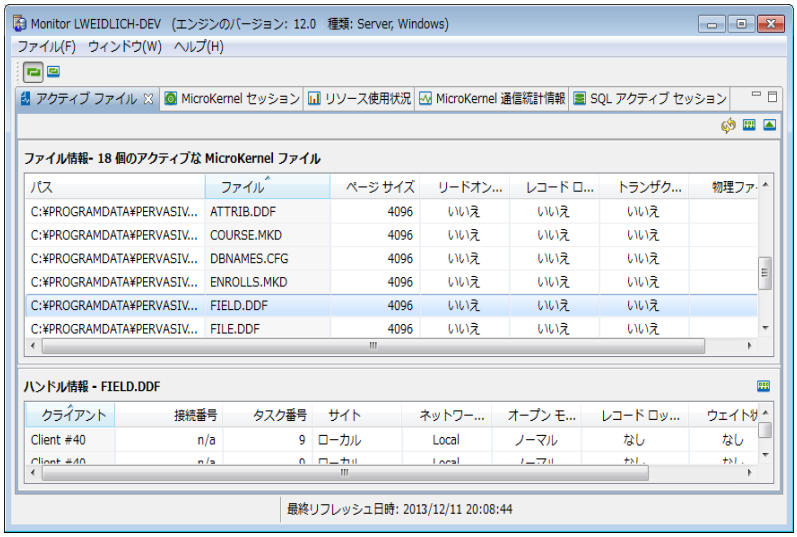
  ```
  zencc.exe -monitor_any
  ```

  監視するサーバーを選択できるダイアログが表示されます。

GUI の機能

下の表は次の図のユーザー インターフェイス コンポーネントの説明です。画像上のそれぞれの領域をクリックするとその詳細が表示されます。

図 5 Monitor のユーザー インターフェイス



GUI のオブジェクト	説明	関連情報
［ファイル］メニュー	リフレッシュ レートを設定できます。	「リフレッシュ オプションの設定」
［ウィンドウ］メニュー	ZenCC のデータベース ウィンドウに戻る、または Monitor の個々のタブを表示して初期設定を行うことができます。	「Monitor の初期設定」
［ヘルプ］メニュー	ドキュメントやログにアクセスすることができます。	「Zen イベント ログ（zen.log）」
タブ設定領域	その時点でデータベースに発生している動作についての情報を表示する、グリッド形式の 5 つのタブがあります。いずれかのタブを選択すると、タブ ラベル行の右端には、そのタブに適用されるアイコンが表示されます。	「タブの機能」

Monitor の初期設定

Monitor の初期設定は、Monitor 自体または ZenCC のどちらからでも設定できます。どちらの場合も、［**ウィンドウ**］>［**設定**］>［Zen ］>［Server Monitor］を選択します。

設定できる初期設定には 2 つのタイプがあります。1 つは、Monitor ユーザー インターフェイスのレイアウト設定です。Monitor を次回開くとき、今回のタブの配置を復元するよう設定できます。これはアクセスするサーバーごとに設定できます。もう 1 つは、特定のグリッドに対する機能の設定です。列幅、ソート順、列の順序などの機能について設定できます。別のサーバーでその特定のグリッドを開いた場合、そのグリッドにも同じ設定が適用されます。このため、異なるサーバーで同じグリッドを簡単に比較することができます。

リフレッシュ オプションの設定

Monitor の情報は、指定した間隔または必要に応じて随時リフレッシュすることができます。また、リフレッシュを行わないようにすることもできます。短い間隔で多くのウィンドウをリフレッシュすると、パフォーマンスが低下する可能性があるので注意してください。

自動リフレッシュ	1. **自動リフレッシュ設定**（[自動リフレッシュはオン / オフになっています]）アイコンを使用して自動リフレッシュをオンにします。 2. ［**ファイル**］>［**自動リフレッシュの設定**］の順に選択、または［**自動リフレッシュレートを設定します**］アイコンをクリックして［**リフレッシュレートの設定**］ダイアログ ボックスを開きます。 3. ［**リフレッシュ レートの設定**］ダイアログ ボックスで、リフレッシュ間隔を秒単位で入力し、［OK］をクリックします。
随時リフレッシュ	［**ファイル**］>［**自動リフレッシュ**］の順に選択するか、または［**データをリフレッシュ**］アイコンをクリックします。
リフレッシュしない	**自動リフレッシュ設定**（[自動リフレッシュはオン / オフになっています]）アイコンを使用して自動リフレッシュをオフにします。

タブの機能

タブは、必要に応じて配置を変更したり、分離させたり、また再び一体化したりすることができます。タブを移動させるには、そのタブ ラベル上にカーソルを置いてマウスの左ボタンを押したまま、目的の位置へ移動します。

タブによってデータの性質が異なるため、実行できる操作もタブごとに異なります。これらの操作は、タブ ラベル行の右端に表示されるアイコンによって実行されます。次の表ではこのアイコンについて説明しています。

表 29　GUI アイコン

アイコン	説明
自動リフレッシュの設定 	自動リフレッシュのオン / オフを切り替えます。「リフレッシュ オプションの設定」を参照してください。
リフレッシュ レートの設定 	Monitor ユーザー インターフェイスの情報を更新する間隔（秒数）を設定するダイアログ ボックスが表示されます。「リフレッシュ オプションの設定」を参照してください。
リフレッシュ 	ユーザー インターフェイスに最新情報を表示します。
表示する列の選択 	表示する列を選択できるダイアログ ボックスが表示されます。含まれる列はタブごとに異なるので、各ダイアログ ボックスには異なる列名が含まれます。すべてのダイアログ ボックスに、［**すべて選択**］と［**すべて選択解除**］ボタンがあります。
ハンドルを表示/非表示 	タブの下部にある［**ハンドル情報**］グリッドの表示 / 非表示を切り替えます。［**ハンドル情報**］グリッドに表示される情報は、タブによって異なります。 ［**ハンドル情報**］グリッドには［**表示する列の選択**］アイコンもあり、必要に応じて列を表示または非表示にすることができます。
選択したセッションの削除 	強調表示されたセッションをタブから削除します。このアイコンは、1 つのセッションが選択された場合にのみ有効になります。 **注意**：この手順はセッションを実質的に終了させるので、進行中の作業を中断させることになります。そのため、この操作を本当に行うかどうかを確認するメッセージが表示されます。

表 29　GUI アイコン

アイコン	説明
全セッションの削除 	すべてのセッションをタブから削除します。 **注意**：この手順はすべてのセッションを実質的に終了させるので、進行中の作業を中断させることになります。そのため、この操作を本当に行うかどうかを確認するメッセージが表示されます。
増加値をリセット 	統計情報を削除し、カウントをゼロから再スタートします。

アクティブ ファイルの監視

［**アクティブ ファイル**］では、現在開いている MicroKernel ファイルについての情報を提供します。監視する列を選択する方法については、「表示する列の選択」を参照してください。

表 30　MicroKernel アクティブ ファイルの監視対象として選択可能な列

パス	ファイルの場所のディレクトリとすべてのサブディレクトリを提供します。
ファイル	ファイル名（サフィックスを含む）を示します。
ページ サイズ	ファイルの各ページ サイズ（バイト単位）を表します。
リードオンリー	このファイルに、オペレーティング システムからリードオンリーのフラグが付けられているかを示します。
レコード ロック	選択したファイルのアクティブ ハンドルのいずれかが、レコード ロックを行っているかどうかを示します。どのアプリケーションもロックされたレコードを読み取ることが可能ですが、レコードの変更や削除ができるのは、ロックを行ったアプリケーションのみです。レコード ロックはファイルを開いたアプリケーションがレコードを更新している期間中にのみかかります。" はい " はファイルに対し 1 つ以上のレコード ロックが適用されていることを示し、" いいえ " はロックされたレコードがないことを示しています。
トランザクション ロック	選択したファイルのアクティブ ハンドルのいずれかが、トランザクション ロックを行っているかどうかを示します。トランザクション ロックはファイルを開いたアプリケーションがトランザクションの作業をしている期間中にのみかかります。
物理ファイル サイズ	ファイルのサイズをキロバイト（KB）で示します。この情報は特に、容量ベースのライセンス モデルでファイルごとの使用データ量を再調査する場合に役立ちます。『*Zen User's Guide*』の「容量ベース ライセンス モデル」も参照してください。 Monitor では、個々のファイルのサイズにはキロバイト（KB）、またリソースの使用状況（「リソース使用状況の監視」）にはメガバイト（MB）の単位を使用します。License Administrator ではサイズの単位にギガバイト（GB）を使用しています。これは、使用データがその単位でキーに関連付けられているからです。コンテキストごとに適切な単位を必要とします。 大量レコードの挿入直後にファイルを閉じた場合、Monitor ではそのファイルのサイズの変更が直ちに反映されません。たとえば、そのファイルに関する［物理ファイル サイズ KB］の統計情報は、次回そのファイルが読み取りまたは書き込みのために開かれるまで更新されません。

アクティブ ファイルのハンドル情報を見ることができます。「ハンドルを表示 / 非表示」を参照してください。アクティブ ファイルのハンドルには以下の情報が含まれます。

表 31　アクティブ ファイルのハンドルに対して選択可能な列

クライアント	名前（通常はユーザーのログイン ID）、またはデータベース サーバーのクライアント リストへのインデックスを示します。
接続番号	クライアントのネットワーク接続番号を表示します。クライアントがネットワーク接続を持たない場合、このフィールドには NA（適用外）と表示されます。
タスク番号	サーバーまたは Windows クライアントで発生するプロセスのタスク番号を表示します。
サイト	ユーザー プロセス（ローカルまたはリモート）のロケーションを示します。
ネットワーク アドレス	ネットワーク上の呼び出し元プロセスのロケーションを表示します。 呼び出し元プロセスが TCP/IP の場合、IP アドレスの先頭には、T4（IPv4 アドレスの場合）、T6（IPv6 アドレスの場合）および T（クライアント マシンの完全修飾ドメイン名の場合）が表示されます。たとえば、次のようになります。 T4: 180.150.1.24 T6: 1234:5678:0000:0000:0000:0000:9abc:def0 T: <*mymachine.mydomain.mycompany*>.com
オープン モード	アプリケーションでファイルの特定のハンドルを開くのに使用する方法を表示します。有効なオープン モードは以下のとおりです。 • ノーマル ― ファイルを開いたアプリケーションでノーマル共有読み取り / 書き込みアクセスが可能 • アクセラレイティド ― ファイルを開いたアプリケーションで共有読み取り / 書き込みアクセスが可能 • リードオンリー ― ファイルを開いたアプリケーションで読み取り専用モードでアクセスできるが、ファイルの変更は不可 • エクスクルーシブ ― ファイルを開いたアプリケーションで排他的なアクセス モードが使用可能。呼び出し元アプリケーションが閉じるまで、ほかのアプリケーションからファイルを開くことはできません。 Monitor では、すべてのオープン モードについて、**非トランザクション**または**共有ロック**が適用可能な場合は、その状態も示します。
レコード ロック タイプ	現在ハンドルで扱われているレコード ロックの種類を表示します。レコード ロックには、[シングル]、[マルチ]、[なし] の 3 種類があります。 シングル レコード ロックの場合、ユーザーは一度に 1 つのレコードをロックすることができます。マルチ レコード ロックの場合、ユーザーは一度に複数のレコードをロックすることができます。
ウェイト状態	このハンドルで、なんらかのロックによってユーザーが待機しているかどうかを次のように表示します。[レコード ロック ウェイト]、[ファイル ロック ウェイト]、または [なし]。
トランザクション タイプ	現在ハンドルで扱われているトランザクション ロックの状態を表示します。トランザクションには、[排他]、[並行]、および [なし] の 3 種類があります。

MicroKernel セッションの監視

[MicroKernel セッション] タブでは、MicroKernel エンジンに対する現在の接続情報を提供します。監視する列を選択する方法については、「表示する列の選択」を参照してください。

表 32　MicroKernel セッションの監視対象として選択可能な列

セッション	接続を特定する一意な識別子を提供します。「セッション」とは、MicroKernel エンジンによって使用されるクライアント ID、またはリレーショナル エンジンへの接続と定義しています。「クライアント ID」とは、データベース トランザクション コンテキストを一意に識別するために、アプリケーション、クライアント プラットフォームおよびデータベース エンジンによって提供される要素を組み合わせた 16 バイトの構造体と定義しています。 セッション情報は、MicroKernel エンジンとリレーショナル エンジンを介して確立されたセッションを示します。リレーショナル エンジンのみで確立されたセッションを表示させる場合は、「SQL アクティブ セッションの監視」を参照してください。 このタブでは、単独セッションの削除、または全セッションの一括削除が行えます。「選択したセッションの削除」および「全セッションの削除」を参照してください。
接続番号	セッションのネットワーク接続番号を表示します。セッションがネットワーク接続を持たない場合、このフィールドには NA（適用外）と表示されます。
タスク番号	サーバーで、または Windows クライアントから発生するプロセスのタスク番号を表示します。
サイト	セッション プロセス（ローカルまたはリモート）のロケーションを示します。
ネットワーク アドレス	ネットワーク上の呼び出し元プロセスのロケーションを表示します。 呼び出し元プロセスが TCP/IP の場合、IP アドレスの先頭には、T4（IPv4 アドレスの場合）、T6（IPv6 アドレスの場合）および T（クライアント マシンの完全修飾ドメイン名の場合）が表示されます。 たとえば、次のようになります。 T4: 180.150.1.24 T6: 1234:5678:0000:0000:0000:0000:9abc:def0 T: <*mymachine.mydomain.mycompany*>.com 単独マシンの複数のクライアントを異なる TCP/IP アドレスで接続する場合、各アドレスはそのクライアントに対して有効です。ただし、データベース エンジン内部的には、クライアントと関連付けられるアドレスは､そのクライアントによって使用される実際のアドレスではないかもしれません。これは、データベース エンジンが同じマシンからの複数のクライアントを特定および管理するからです。その結果、Monitor はエンジン情報を報告しているので、このツールでは実際のアドレスではなく、関連付けられたアドレスが表示される可能性があります。
使用中ロック数	セッションが現在使用中のロック数。
トランザクション タイプ	セッションが現在扱っているトランザクション ロックの種類。トランザクションには、［排他］、［並行］、および［なし］の 3 種類があります。
読み込みレコード数	セッションがファイルを最初に開いてから現在までに読み取ったレコードの数。
挿入レコード数	セッションが挿入したレコードの数。
削除レコード数	セッションが削除したレコードの数。
更新レコード数	セッションが更新したレコードの数。
ディスク アクセス数	セッションがディスク アクセスを要した回数。開いたばかりのファイルでは、ディスク アクセスについての情報は表示できません。
キャッシュ アクセス数	このクライアントが、リクエストに応えるために L1 または L2 キャッシュのデータを検索した回数を示します。

MicroKernel セッションのハンドル情報を見ることができます。「ハンドルを表示 / 非表示」を参照してください。MicroKernel セッションのハンドルには以下の情報が含まれます。

表 33　MicroKernel セッションのハンドル情報に対して選択可能な列

パス	ファイルの場所のディレクトリとすべてのサブディレクトリを提供します。
ファイル	ファイル名（サフィックスを含む）を示します。
オープン モード	アプリケーションでファイルの特定のハンドルを開くのに使用する方法を表示します。有効なオープン モードは以下のとおりです。 ノーマル ― ファイルを開いたアプリケーションでノーマル共有読み取り / 書き込みアクセスが可能 アクセラレイティド ― ファイルを開いたアプリケーションで共有読み取り / 書き込みアクセスが可能 リードオンリー ― ファイルを開いたアプリケーションで読み取り専用モードでアクセスできるが、ファイルの変更は不可 エクスクルーシブ ― ファイルを開いたアプリケーションで排他的なアクセス モードが使用可能。呼び出し元アプリケーションが閉じるまで、ほかのアプリケーションからファイルを開くことはできません。 Monitor では、すべてのオープン モードについて、**非トランザクション**または**共有ロック**が適用可能な場合は、その状態も示します。
レコード ロック タイプ	現在ハンドルで扱われているレコード ロックの種類を表示します。レコード ロックには、［シングル］、［マルチ］、［なし］の 3 種類があります。 シングル レコード ロックの場合、ユーザーは一度に 1 つのレコードをロックすることができます。マルチ レコード ロックの場合、ユーザーは一度に複数のレコードをロックすることができます。
ウェイト状態	このハンドルで、なんらかのロックによってユーザーが待機しているかどうかを次のように表示します。［レコード ロック ウェイト］、［ファイル ロック ウェイト］、または［なし］。
トランザクション タイプ	現在ハンドルで扱われているトランザクション ロックの状態を表示します。トランザクションには、［排他］、［並行］、および［なし］の 3 種類があります。

リソース使用状況の監視

［**リソース使用状況**］タブでは、エンジンが最後に起動してから現在まで MicroKernel によって使用されているリソースを表示します。監視する列を選択する方法については、「表示する列の選択」を参照してください。

データベース エンジンは、これらのリソースのうちいくつかの最大値を動的に制御します。ユーザー数、セッション数および使用中データの最大値は製品ライセンスによって決まります。『*Zen User's Guide*』の「ライセンス モデル」を参照してください。

リソースが監視対象の Zen 製品の種類に適用されない場合、そのリソースの各統計情報には "n/a"（適用外）が表示されます。たとえば、ユーザー数は Cloud Server には適用されません。このため、Cloud Server が監視される場合は、［ユーザー数］の［現在値］、［ピーク値］および［最大値］には "n/a" が表示されます。同様に、Zen サーバーが監視される場合は、［セッション数］と［使用中データ MB］の［最大値］には "n/a" が表示されます。

Cloud Server の使用を検討されている場合、［使用中データ MB］の現在値やピーク値の予測が必要です。そのため、それらの統計情報はサーバーでも表示されますが、実施はされません。これらの統計情報については、その値に関わらず通知は送信されません。

エンジンの稼働時間	MicroKernel エンジンが起動されている時間が、週、日、時、分、秒で示されます。 エンジン稼働時間は、リソース使用状況の表示対象として選択できる列ではありません。これはグリッド タイトルの一部です。

表 34　MicroKernel リソース使用状況の監視対象として選択可能な列

リソース	監視するリソースのタイプを示します。「リソース使用状況の監視の対象となるリソースの種類」を参照してください。
現在値	現在の使用状況をリソース別に示します。
ピーク値	MicroKernel が起動してから現在までにそのリソースが記録した最高値を示します。
最大値	各リソースに対して許容される最大値を示します。

次の表では、使用状況の監視の対象となるリソースの種類を示しています。

表 35　リソース使用状況の監視の対象となるリソースの種類

ファイル数	MicroKernel によって現在開かれているファイル数。このリソースの最大値に制限はありません。
ハンドル数	アクティブ ハンドルの数。MicroKernel は、ユーザーがファイルを開くたびにハンドルを作成します。1 つのセッションが同じファイルに対していくつかのハンドルを持つことができます。このリソースの最大値に制限はありません。
クライアント数	MicroKernel にアクセス中のクライアントの数。1 台のマシンで複数のクライアントがデータベース エンジンに同時にアクセスできます。データベース エンジンはクライアント一覧を動的に管理します。このリソースの最大値は無制限です（クライアント数はコンピューターのメモリによってのみ制限されます）。 「クライアント」とは、クライアント ID（トランザクショナル エンジン インターフェイス）によって、またはリレーショナル エンジン インターフェイスへの接続によって確立されたセッションをいいます。データベース エンジンは、Zen システム ファイル、メタデータ ファイル、dbnames.cfg およびデフォルトのシステム データベースへのアクセスなど、独自の内部処理用にさまざまなクライアント セッションを使用します。クライアント数は、内部的なクライアント セッションと非内部的なクライアント セッションの両方を示します（「MicroKernel セッションの監視」を参照）。
ワーカ スレッド数	MicroKernel の並行プロセスの数。
ユーザー数	同時に接続しているユーザー数。この最大値は、使用許諾契約書に従って許可される、最大ユーザー数です。
セッション数	データベース エンジンによって使用されているセッション数。「使用されているセッション数」は端的に「セッション数」ともいいます。この最大値（「セッション数の制限値」ともいいます）は、使用許諾契約書に従って許可される最大セッション数を示します。 セッション数は、MicroKernel エンジンまたはリレーショナル エンジンを介して確立されたすべてのセッションを示します。 セッション数に関するメッセージは、Zen のさまざまなログ リポジトリに記録されます。「メッセージ ログの見直し」を参照してください。 データベース エンジンは、Zen システム ファイル、メタデータ ファイル、dbnames.cfg およびデフォルトのシステム データベースへのアクセスなど、独自の内部処理用にさまざまなセッションを使用します。これらの内部セッションについてはセッション数を消費しません。

表 35　リソース使用状況の監視の対象となるリソースの種類

使用中データ MB	同時に開く全データ ファイルのサイズ（MB 単位）。この最大値は、使用許諾契約書に従って許可される、同時に開く全データ ファイルの総量です。この最大値は、「使用データの制限値」ともいいます。 使用データの値は、データ ファイルが初めて開かれたときに増加します。データ ファイルが既に開かれていれば、それ以降に開く際には総量に加算されません。使用データは、開くファイルのサイズが増加した場合にも増えます。既に開かれているファイルのサイズが大きくなって使用データの制限を超えても、そのファイルに対する操作は引き続き許可されます。 使用データの値は、データ ファイルを開いた最後のユーザーがそのデータ ファイルを閉じたとき減少します。複数のユーザーが同じデータ ファイルにアクセスできるので、そのデータ ファイルを開いた**すべての**ユーザーがそのファイルを閉じないと使用データは減少しません。 データに関するメッセージは、Zen のさまざまなログ リポジトリに記録されます。「メッセージ ログの見直し」を参照してください。 データベース エンジンは、Zen システム ファイル、メタデータ ファイル、dbnames.cfg およびデフォルトのシステム データベースなど、独自の内部処理用にさまざまなファイルを使用します。内部処理に使用するファイルによって使用データの値が増えることはありません。 大量レコードの挿入直後にファイルを閉じた場合、Monitor ではそのファイルのサイズの変更が直ちに反映されません。たとえば、そのファイルに関する［使用中データ MB］の統計情報は、次回そのファイルが読み取りまたは書き込みのために開かれるまで更新されません。
トランザクション数	トランザクションの数。このリソースの最大値に制限はありません。
ロック数	レコード ロックの数。このリソースの最大値に制限はありません。

MicroKernel 通信統計情報の監視

［MicroKernel **通信統計情報**］タブでは、MicroKernel エンジンとの通信に関する情報を表示します。これは、「通信統計情報」セクションと「リソース使用状況情報」セクションに分かれています。通信統計情報は、データベース エンジンの起動時から現在までに処理された回数の合計が計算されています。

監視する列を選択する方法については、「表示する列の選択」を参照してください。

エンジンの稼働時間	MicroKernel エンジンが起動されている時間が、週、日、時、分、秒で示されます。 エンジン稼働時間は、MicroKernel 通信統計情報の表示対象として選択できる列ではありません。これはグリッド タイトルの一部です。

通信統計情報

表 36　MicroKernel 通信統計情報の監視対象として選択可能な列

リソース	監視するリソースのタイプを示します。「MicroKernel 通信統計情報のリソースの種類」を参照してください。
合計値	データベース エンジンの起動時から現在までに発生した回数の合計を示します。
増加値	［MicroKernal 通信統計情報］タブへ最後にアクセスしてから現在までに発生した回数を示します。増加値のカウントをリセットして再スタートさせるには、［**増加値をリセット**］をクリックします。

表 37　MicroKernel 通信統計情報のリソースの種類

処理済リクエスト総数	データベース エンジンが処理している、ワークステーションやリモート、サーバーベース アプリケーションからのリクエストの数。
TCP/IP 処理済リクエスト数	データベース エンジンが処理しているクライアントやリモート、サーバーベース アプリケーションからの TCP/IP リクエストの数。
接続タイムアウト数	Auto Reconnect がクライアントに再接続を試行するときのタイムアウトの回数。「自動再接続タイムアウト」も参照してください。
接続復元数	Auto Reconnect 機能が接続タイムアウトから回復に成功した回数を示します。

リソース使用状況情報

リソース使用状況情報では、リソース発生回数の現在値、ピーク値、および最大値を提供します。

表 38　MicroKernel 通信統計情報に関するリソース使用状況の監視対象として選択可能な列

リソース	監視するリソースのタイプを示します。「MicroKernel 通信統計情報に関するリソース使用状況のリソースの種類」を参照してください。
現在値	現在の使用状況をリソース別に示します。
ピーク値	MicroKernel が起動してから現在までにそのリソースが記録した最高値を示します。
最大値	各リソースに対して許容される最大値を示します。

表 39　MicroKernel 通信統計情報に関するリソース使用状況のリソースの種類

通信スレッド数	MicroKernel で現在処理中のリモート リクエストの数。ここには、ローカルのリクエストについての情報は含まれません。処理中のリモートおよびローカル スレッドの合計については、「リソース使用状況の監視」を参照してください。 データベース エンジンは必要に応じて、通信スレッド数を許容される最大値まで動的に増加させます。Windows、Linux、および macOS では、最大値は 1024 です。 通信スレッドは Monitor のリクエストを処理するためにも使用されるので、現在の通信スレッド数にはこの数も含まれています。この状態は正常です。
リモート セッション総数	データベース エンジンに接続したリモート クライアントの数。最大数は動的で、ゼロが表示されます。
TCP/IP リモート セッション数	TCP/IP プロトコルを介してデータベース エンジンに接続したリモート クライアントの数。

SQL アクティブ セッションの監視

［SQL アクティブ セッション］タブでは、リレーショナル エンジンに対する現在の接続情報を提供します。このタブでは、SQL セッションの削除も行えます。「選択したセッションの削除」を参照してください。監視する列を選択する方法については、「表示する列の選択」を参照してください。

表 40　SQL アクティブ セッションの監視対象として選択可能な列

ユーザー名	ユーザーのログイン名を提供します。
クライアント ホスト名	選択されたユーザー名に対応するクライアント マシンの名前。使用不可の場合、［不明］に設定されます。

表 40　SQL アクティブ セッションの監視対象として選択可能な列

ネットワーク アドレス	選択されたユーザー名のクライアント マシンの IP アドレス。使用不可の場合、［不明］に設定されます。 表示される値には、"IP"、"Shared Memory" および " 不明 " があります。
クライアント　アプリケーション	接続したアプリケーションまたはモジュール。使用不可の場合、［不明］に設定されます。
データ ソース名	クライアント アプリケーションによって参照される、DSN の名前。
接続状態	選択されたユーザーの接続状態。接続状態のタイプは次のとおりです。 • ［アクティブ］ー セッションには開いているファイルがあります。 • ［アイドル］ー セッションには開いているファイルがありません。 • ［無効］ー アクティブ セッションは削除されていますが、SQL コードの処理が終了されていないことを示す一時的な状態。適切な終了時点で、そのセッションは［SQL アクティブ セッション］ダイアログに表示されなくなります。 • ［不明］ー 状態を入手できません。
アクティブ / アイドル時間	接続がアクティブまたはアイドルになってからの時間（ミリ秒単位）を表示します。
総接続時間	接続が確立されてからの時間（秒単位）を表示します。

Monitor のコマンド ライン インターフェイス

以下のトピックでは、Monitor のコマンド ライン バージョン bmon について説明します。

bmon へのアクセス

bmon コマンド ライン ツールは、Zen でサポートされる Windows、Linux、macOS、および Raspbian プラットフォームで実行します。この実行可能プログラムは、bmon.exe（Windows の場合）および bmon（Unix ベースのシステムの場合）です。Windows システムの場合、このツールは Zen インストール ロケーションの bin ディレクトリにあります。Unix システムの場合、このツールは /usr/local/actianzen/bin にあります。

設定ファイルの設定

bmon は構成ファイルに基づいてサーバー設定を管理します。Zen はデフォルトの設定ファイル monconfig.txt を Zen のインストール先の bin ディレクトリに用意しています。ファイル内の各設定の説明はコメントに記載されています。

bmon の出力

bmon からの出力をコンソールおよびログ ファイルのいずれか、または両方にリダイレクトすることができます。設定ファイルで出力先を指定することができます。たとえば、アプリケーションは、コンソールやログ ファイルで特定の状態を調べてから適切な対応をとることができます。Zen v15 では、-runonce オプションで JSON 出力がサポートされるようになりました。

コマンド構文

```
bmon -f [filepath]config_file [-runonce][-JSON]
```

オプション

-f	設定ファイルがツールに対して入力を提供することを指定するために必要なパラメーターです。
filepath	設定ファイルのパス。省略すると、bmon は設定ファイルをローカル ディスクで検索します。
config_file	設定ファイルの名前。ファイル名は、自由に選択できます。インストレーションには monconfig.txt がデフォルト ファイルとして含まれています。このファイルは Zen のインストール ディレクトリ内の bin ディレクトリにあります。ファイル内のコメント行には全オプションについての説明が記載されています。
-runonce	bmon を 1 度実行したら終了させるためのオプション パラメーターです。このパラメーターは、bmon がバッチ ファイルから呼び出される場合に特に役立ちます。「runonce パラメーターを使用しない bmon の実行」も参照してください。 **メモ**：リモートの PowerShell セッションなど stdin を使用しないコマンド ライン環境では、指定されていなくても bmon はこのオプションを使用します。bmon を複数回実行させるためには、bmon 設定ファイルで limitrefresh 設定を使用します。
-JSON	JSON 形式で出力を生成するためのオプション。-runonce パラメーターを指定した場合のみ使用できます。

runonce パラメーターを使用しない bmon の実行

runonce パラメーターは省略可能です。省略した場合、bmon は設定ファイルで指定された設定を使用します。サンプル ファイルの monconfig.txt では、リフレッシュ レート（refreshrate）の初期値は 5 秒に設定されています。bmon の実行中、いつでも手動で終了することができます。終了には Q（または q）+ Enter キーを使用します。

リフレッシュ レートが 0（ゼロ）の場合、bmon はキーボードのキー応答があるまで停止します。設定ファイルのリフレッシュ レート設定（refreshrate）により、停止する時間を指定します。

リフレッシュ オプション	注記
refreshrate=5 （5 秒間停止後、bmon を再実行） refreshrate=0 （キーボードの有効なキー応答を受け取るまで停止）	この値にはゼロまたは 5 以上の整数を指定します。 サンプル ファイル monconfig.txt でのデフォルト値は 5 です。
limitrefresh=0 （手動で bmon を停止させるには、Q または q + Enter キーを使用) limitrefresh=*n* （bmon を *n* 回実行した後に終了）	最大値は 2147483647 です。 サンプル ファイル monconfig.txt でのデフォルト値はゼロです。 ローカル実行を想定した stdin のセッションが、リモート接続であったことが原因で失われた場合、bmon は " 対話型入力でエラーが検出されました。" という警告を出して終了します。limitrefresh に 1 以上を設定すると、このメッセージは表示されなくなります。

データ ファイルの断片化の監視

データベースが頻繁に使用される状態が続くと、その間にレコードの作成、更新、または削除が繰り返し行われるので、データが断片化され、ファイルへのアクセスやトランザクションの応答に時間がかかるようになることがあります。この場合の断片化とはデータ ファイル内で起こるものであり、ハード ディスク上のファイル システムの断片化とは異なります。開発者またはデータベース管理者であれば、ファイルの集中使用から判断してファイルがいつ断片化されやすいかはご存知でしょうが、システムによってはその断片化のタイミングを推測している（断定できない）かもしれません。

Defragmenter は、データの断片化を検出し、それを修正することでこの問題を解決するツールです。最適化は、データ ファイル内でレコードの再配置やインデックスの再構築を行ったり、未使用領域を除去したりして、再びデータへ効率よくアクセスできるようにします。ファイルの最適化によってそのファイルのデータが変更されることはありません。また、ファイルの最適化中でもレコードの作成、読み取り、更新または削除が可能です。ほとんどの場合、Defragmenter の機能を使用するためにダウンタイムを設ける、または業務を停止する必要はなく、データベース エンジンの実行中にもこの機能を使用することができます。

Defragmenter は Zen Control Center から開くことができるグラフィカル ツールの 1 つです。このツールでは、使用中のデータ ファイルと、その読み取り / 書き込み回数が表示されるので、頻繁に使用されるデータ ファイルをすばやく見つけることができます。ファイルまたはテーブルを［ウォッチ リスト］タブに追加するには、その追加対象を選択してドラッグ アンド ドロップ、ボタンをクリック、または右クリックしてコマンドを選択します。別の場所に Btrieve ファイルがある場合は、その場所を参照し、監視対象として追加することもできます。

Defragmenter は dbdefrag ツールとしても実行します。

以下のセクションでは、Defragmenter の使用方法について説明します。

- 「最適化を行うタイミングの判断」
- 「Defragmenter を使用できない状況」
- 「Defragmenter へのアクセス」
- 「Defragmenter の GUI」
  - 「Defragmenter の初期設定」
  - 「自動リフレッシュ間隔の設定」
  - 「タブの配置」
- 「Defragmenter のタスク」
- 「Defragmenter のコマンド ライン インターフェイス」

Defragmenter を無人モードで使用する方法については、「自動最適化」を参照してください。

最適化を行うタイミングの判断

Defragmenter は、データ ファイルについて、パフォーマンス低下の原因を示す統計情報を分析するのに役立ちます。ファイルサイズ、断片化率、未使用率、および順序不同率の統計値が大きいと、データベースのパフォーマンス低下の原因になることがあります。ファイルまたはテーブルを最適化すれば、これら 4 つの数値をすべて下げることができます。詳しくは、「ウォッチ リスト」セクションで説明しています。一般に、コンパクト化され、インデックスを作成し直した新しいファイルは、効率性、容量およびパフォーマンスが復元し、トランザクションもより迅速に実行します。

データベースはそれぞれ異なるので、すべての人に該当するような推奨事項はありません。最適化を行うかどうかは、使用しているデータベースとそのアプリケーションに対する知識や経験値によって異なってきます。ただし、以下の一般的な事項は検討する価値があります。

- 読み取り専用データベースではパフォーマンスは一定ですが、読み取り / 書き込みが長期間にわたり発生するようなデータベースの場合は、監視対象のファイルの分析において統計情報の値が高くなります。データベース動作の変化（クエリやレポートの実行速度が遅くなるなど）は顕著に現れます。
- 多くの場合、一括削除操作によってファイル内の未使用領域が非常に大きくなりますが、これはそのファイルを最適化することで補正できます。

- 最適化の実施中でも読み取り / 書き込みが中断することはありません。ただし、以下のような理由から、最適化をトラフィックが低い時間に実施することも検討できます。
  - 最適化はリソースを使用しますが、通常、そのリソースは全面的にデータ ファイルに対する操作に使われています。このため、トラフィックが高い期間にはエンジンへの必要以上の要求がパフォーマンスに少なからず影響を与える可能性があります。
  - 最適化では、実行対象の各データ ファイルを短時間の間ロックする必要があります。トラフィックが低い間は、このロックの影響がないかもしれませんが、トラフィックが高い時間帯では、クライアントが応答を少し待つこともあります。

最適化を実施してもパフォーマンスの改善が見られない場合は、ほかの問題が原因である可能性があり、別の診断や解決策が必要です。

Defragmenter を使用できない状況

場合によっては、Defragmenter を使用できなかったり、優先順位の上位にランク付けされているデータベース活動によって操作をキャンセルされたりすることがあります。

- クライアントにおけるファイルの最適化は現在サポートされていません。このツールを実行できるローカル サーバーでのみファイルを最適化できます。
- ファイルを最適化するのに必要なディスク容量は、分析結果に表示されます。Defragmenter は、ファイルのサイズと同じだけの空きディスク容量を必要とするほかに、最適化操作のために少量の空き容量を必要とします。書き込みが頻繁に行われるファイルについては、さらに多くの空き容量が必要になる可能性があります。
  - ディスクの空き容量が少なすぎると、エンジンは最適化を開始する試みを拒否し、エラー コード 126 を返します。最適化を実行している最中にディスク ボリュームの空き容量が少なくなった場合には、エンジンはそのボリュームで実行しているファイルの最適化をキャンセルし、エラー コード 126 を返します。
  - 最適化を開始したものの、実行しているディスク ボリュームがいっぱいになった場合には、エンジンはそのボリュームでの一連の最適化をキャンセルし、エラー コード 18 を返します。
- ほかの ZenCC ツールを使用してインデックスやメタデータの変更を行います。これは以下のような場合です。
  - インデックスの作成、変更、または削除
  - 列の作成、変更または削除などのスキーマ編集
  - データベース、テーブル、またはデータ ファイルの削除
  - データベース、テーブル、またはファイル名の変更
  - オーナー ネームの設定またはクリア
  - バウンド データベースの作成、変更、または削除
- 最適化を実行中のファイルに対して、Backup Agent 操作を実行することはできません。最適化を実行するには、Backup Agent を先に終了しておく必要があります。
- 最適化を実行中のファイルに対して、データ バックアップを目的とする Continuous オペレーションを実行することはできません。最適化は、より優先度の高い操作によってキャンセルされます。最適化を実行するには、それらの操作が終了するのを待つ必要があります。
- 最適化は現在、VSS を使用してバックアップ操作を行っている環境にあるサーバー エンジンではサポートされません。
- データ ファイルのプロパティ（ページ サイズ、圧縮またはファイル バージョンなど）を変更する必要がある場合は、Defragmenter ではなく Rebuild（[ツール] > [Rebuild]）を使用してください。
- Defragmenter で破損レコードや間違ったレコードが見つかった場合、Defragmenter は自動的に停止し終了します。このような場合、Rebuild を使用してデータを回復させてください。最近リビルド（再構築）されたファイルは最適化する必要はありません。
- DataExchange ユーザー向け : DataExchange で使用するシステム データやキーが最適化によって変更されることはありません。最適化後に、テーブル同期やチェック ツール dxsynctables を実行する必要はありません。

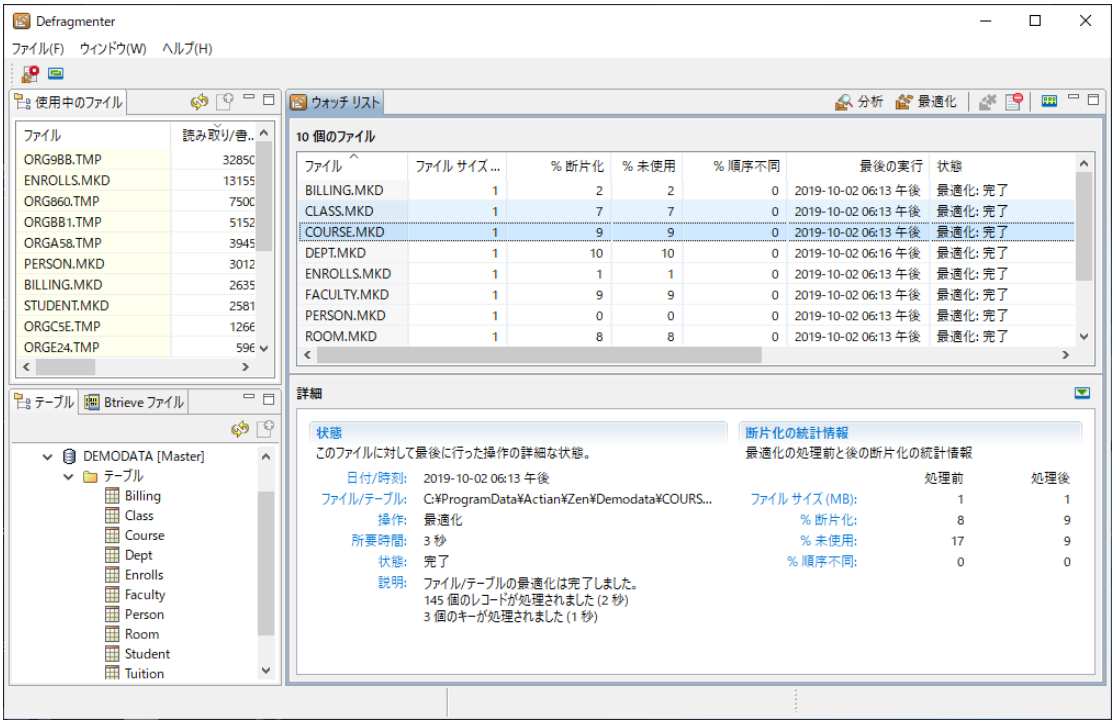
Defragmenter へのアクセス

Defragmenter は次の 3 つの方法で実行します。

- Zen Control Center で［**ツール**］>［**Defragmenter**］をクリックします。詳細については、「Defragmenter の GUI」を参照してください。
- コマンド プロンプトで、**dbdefrag** を実行します。これは Windows ファイル システムの最適化に用いられる Microsoft ツール **defrag** とは異なるので注意してください。詳細については、「Defragmenter のコマンド ライン インターフェイス」を参照してください。
- サーバー エンジンの［パフォーマンス チューニング］設定で、［自動最適化］オプションをオンにすることができます。詳細については、「自動最適化」を参照してください。

Defragmenter の GUI

次のスクリーンショットは Defragmenter の GUI を示しています。その下の表は Defragmenter GUI の各オブジェクトの説明です。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。



GUI のオブジェクト	説明
［ファイル］メニュー	［ウォッチ リスト］タブのリフレッシュ レートの設定、Defragmenter の終了、または ZenCC の終了を実行できます。
［ウィンドウ］メニュー	ZenCC ウィンドウに戻る、または各種設定の初期値を設定できます。
［ヘルプ］メニュー	ドキュメントやログへのアクセス、および Zen のバージョンの確認が行えます。

表 41　Defragmenter のアイコン

アイコン	説明
すべての最適化を取り消す 	すべての実行中の分析や最適化を手動で停止することができます。取り消すということは、現在分析されているどのファイルも新しい統計情報を表示しないということです。最適化対象のファイルのうち、まだ完了していないファイルは変更されません。分析待ちまたは最適化待ちのファイルは実行されません。 すべてのアクティブな分析と最適化の操作を取り消すには、ウィンドウの左上隅にあるこのアイコンをクリックします。このオプションを使用する場合には、ファイルを選択する必要はありません。 **メモ**: すべての操作は、コマンド プロンプトで開始したものであっても、Defragmenter ウィンドウから取り消すことができ、その逆も同様です。
自動リフレッシュ レートを設定します 	[ウォッチ リスト] タブを更新する間隔を設定するダイアログを開きます。値の単位は秒です。デフォルト値は 5 です。
リフレッシュ 	[使用中のファイル]、[テーブル]、または [Btrieve ファイル] タブ内の項目一覧を手動で更新することができます。自動リフレッシュ レートの設定が作用するのは [ウォッチ リスト] タブのみであることに注意してください。
ウォッチ リストに追加 	選択した項目を [ウォッチ リスト] タブに入れます。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。
ファイルの分析 	[ウォッチ リスト] タブで選択した項目について断片化の統計情報を収集して表示します。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。ファイルを分析することによって、そのファイルの内容が変更されることはありません。
ファイルの最適化 	[ウォッチ リスト] タブで選択した項目に対し最適化を開始します。Ctrl キーを使用して項目をクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。
最適化を取り消す 	実行中の分析や最適化を任意に選択して停止することができます。取り消すということは、現在分析されているどのファイルも新しい統計情報を表示しないということです。最適化対象のファイルのうち、まだ完了していないファイルは変更されません。 [ウォッチ リスト] の右上にあるこのアイコンは、[ウォッチ リスト] 内のファイルを分析または最適化しているときに有効になります。1 つ以上のファイルを選択してこのアイコンをクリックすると、選択したファイルに対する操作が取り消されます。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。
ファイルの削除 	[ウォッチ リスト] タブから項目を削除します。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。
表示する列の選択 	[ウォッチ リスト] に表示する列をカスタマイズ（チェック ボックスをオン / オフ）するためのダイアログを開きます。デフォルト設定では、すべての列が表示されるようになっています。
詳細を表示する / 非表示にする 	[詳細] パネルの表示を切り替えます。

Defragmenter のタブ表示

Defragmenter では、データベース テーブル、ファイル、およびその他のオブジェクトをそれぞれのタブに表示します。

- 「使用中のファイル」
- 「テーブル」
- 「Btrieve ファイル」
- 「ウォッチ リスト」

Defragmenter ウィンドウ内のタブの配置は変更することができます。「タブの配置」を参照してください。

使用中のファイル

［使用中のファイル］タブでは、Zen エンジンで現在開いている、または最近開いたファイルを一覧表示します。このタブに表示されるファイルは［ウォッチ リスト］タブに追加して、監視、分析、および最適化を行うことができます。パフォーマンスの低下の一因となる可能性が高いので、データベース作業のルーチン実行に関わるファイルのみを選択することをお勧めします。

［使用中のファイル］タブでは、［読み取り / 書き込み］列の見出しをクリックすることで、リストを並べ替えることができます。リストを並べ替えることで、最も頻繁に使用されており、断片化されている可能性が高いファイルを特定することができます。

テーブル

［テーブル］タブは、Zen エクスプローラーと似ており、Defragmenter が実行しているサーバーのノードを表示します。このタブに表示されるファイルは［ウォッチ リスト］タブに追加することができます。

Btrieve ファイル

［Btrieve ファイル］タブは、ファイル システム上のドライブやディレクトリのファイル エクスプローラーです。このタブでは、データ ファイルに加えルーチン オペレーションに含まれない項目（dbnames.cfg や .ddf ファイルなど）も一覧に表示されます。これらのファイルは［ウォッチ リスト］タブに追加して分析や最適化を行うことができますが、データ ファイルとは異なり、それらのファイルを変更しようとしたときにファイルが使用中であった場合はエラーが返ります。これらのメタデータ オブジェクトは断片化のリスクが低いです。これは一般に実稼働環境では変更しないからです。まれなケースとしてこれらの最適化が必要となった場合は、ダウンタイムのメンテナンス中に行うことができます。

ウォッチ リスト

［ウォッチ リスト］タブ内の項目は、分析または最適化が適用された後の統計情報を表しています。

統計情報	評価基準	解説
ファイル サイズ（MB）	ファイルのサイズ（メガバイト単位）	ファイルが使用されている時間が長くなるほど、またファイルに含まれる未使用領域が大きくなるほど、ファイル サイズは大きくなります。 通常、ファイルのサイズが小さいほど、（そのファイルに多くのインデックスがある場合を除き）最適化にかかる時間も短くなります。 Cloud Server ユーザーの場合、ファイルが断片化されることでサイズが大きくなることが懸念されます。そのような場合、最適化を行うことでライセンス制限内に収められる可能性があります。
% 断片化	データが未使用領域によって分離されてブロック化している割合	長期間にわたりトランザクションを実行すると、データを含まないページを作成する可能性があります。このようなページがより多く実際のデータ ページと混在していると、断片化の度合いが大きくなります。 割合が低ければ断片化も少なく、より大きなデータ ブロックが互いに近い場所で格納されるため、読み取りや書き込みの時間も速くなります。 **メモ**：この統計情報は Btrieve 6.x および 7.x ファイルには対応していません。これらの古いファイル バージョンを分析し、問題なく最適化することはできますが、統計情報に測定値は表示されません。
% 未使用	未使用領域の割合	未使用領域は多くの場合、挿入操作、更新操作、削除操作によってファイルが変更されたときに生じます。 この割合が低ければ、ファイルはよりコンパクトになり、読み取りや書き込みの時間も速くなります。 **メモ**：この統計情報は Btrieve 6.x および 7.x ファイルには対応していません。これらの古いファイル バージョンを分析し、問題なく最適化することはできますが、統計情報に測定値は表示されません。
% 順序不同	レコードが連続して格納されていない割合	長期間にわたり挿入を行うと、ページの論理的な順序と物理的な位置の間の不整合が拡大し、データの場所を探すのに要する時間が長くなります。 一般に、割合が低ければ大量ファイルに対するテーブル スキャンなどの動作のパフォーマンスが向上します。 **メモ**：ファイルによっては、順序不同の想定される最低の割合がゼロよりも高いことがあります。そのファイル内のレコードは既に可能な限り効率よく格納されているので、さらに最適化を実施してもこの統計値は低くなりません。
最後の実行	最後に操作を実行した日付と時刻	アクションは分析または最適化です。
状態	最後に行った操作の報告	初めて項目を追加した場合は空白になっています。通常は、完了した分析または最適化の報告です。
テーブル	論理的な場所	データベースおよびテーブル / ファイル名
パス	物理的な場所	ファイル システムのパス

詳細

［ウォッチ リスト］タブの下にある［詳細］ペインでは、［ウォッチ リスト］タブ内の列の統計情報をまとめたものですが、さらに次の情報が追加されています。

- 最適化の実施に要した時間
- 最適化処理前後の、ファイル サイズの比較、また断片化、未使用、順序不同の割合の比較

［ウォッチ リスト］タブで 1 つの項目を選択すると、その項目の詳細が表示されます。複数の項目を選択した場合は、強調表示（選択）されている項目のうち一番上にある項目の詳細が表示されます。

Defragmenter の初期設定

最適化の初期設定は、ツール内または Zen Control Center のどちらからでも設定できます。どちらの GUI でも、［**ウィンドウ**］>［**設定**］>［Zen］>［Defragmenter］の順に選択すれば、［設定］ダイアログ ボックスの［Defragmenter］ページが開きます。

次の表では、設定できる初期設定とその内容について示しています。

初期設定	選択時の動作
ウィンドウ レイアウトを保存	Defragmenter ツールを終了するときに、その時点のウィンドウ全体のサイズ、高さおよび位置、さらにタブの配置も保存します。
複数選択の操作中に、互換性のない操作の警告を表示しない	異なるタイプの項目に対して操作が不規則に適用される場合に、警告ダイアログ ボックスが表示されないようにします。
今後、最適化の取り消しに対する警告を表示しない	分析や最適化の操作を取り消すときに、警告を示すダイアログ ボックスが表示されないようにします。 取り消しを確認するダイアログ ボックスのチェック ボックスをオンにした場合、このチェック ボックスもオンになります。確認のダイアログ ボックスが再び表示されるようにするには、このチェック ボックスをオフにします。
今後、すべての最適化の取り消しに対する警告を表示しない	すべての分析や最適化の操作を取り消すときに、警告を示すダイアログ ボックスが表示されないようにします。 取り消しを確認するダイアログ ボックスのチェック ボックスをオンにした場合、このチェック ボックスもオンになります。確認のダイアログ ボックスが再び表示されるようにするには、このチェック ボックスをオフにします。

自動リフレッシュ間隔の設定

Defragmenter の情報は、指定した間隔で自動的にリフレッシュすることができます。デフォルトでは 5 秒間隔でリフレッシュされます。頻繁に多くのウィンドウをリフレッシュすると、パフォーマンスが低下する可能性があるので注意してください。

タブの配置

Defragmenter のタブは、必要に応じて配置を変更したり、分離させたりすることができます。タブを移動させるには、タブのラベル部分を希望の移動先までドラッグ アンド ドロップします。Defragmenter の初期設定で、［ウィンドウ レイアウトを保存］オプションを選択した場合は、変更したタブの配置が維持されます。

Defragmenter のタスク

ここでは、Defragmenter GUI におけるざまなタスク向け機能を使用する手順を提供します。

▶▶ 自動リフレッシュの間隔（自動リフレッシュ レート）を設定するには

自動リフレッシュ レートは［ウォッチ リスト］タブの統計情報に適用されます。それ以外のタブ一覧をリフレッシュするには、各タブで［更新］アイコンをクリックするか右クリックで［更新］を選択し、手動で行います。

1 ［ファイル］>［自動リフレッシュ レートの設定］の順にクリックするか、または［自動リフレッシュ レートを設定します］アイコン をクリックして［リフレッシュ レートの設定］ダイアログ ボックスを開きます。

2 このダイアログ ボックスで、リフレッシュ間隔を秒単位で入力します。この値には 1 以上の整数を指定する必要があります。

3 ［OK］をクリックします。

▶▶［ウォッチ リスト］タブへファイルまたはテーブルを追加するには

［ウォッチ リスト］タブへのファイルやテーブルの追加は、次のようにさまざまな方法で行うことができます。

- ［使用中のファイル］タブ、［テーブル］タブ、または［Btrieve ファイル］タブ内で項目を 1 つ以上選択し、タブの右上にある［ウォッチ リストへ追加］アイコン をクリックします。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。
- ［使用中のファイル］タブ、［テーブル］タブ、または［Btrieve　ファイル］タブ内で項目を右クリックし、［ウォッチ リストへ追加］を選択します。
- 各タブ内で項目を選択し、［ウォッチ リスト］タブへドラッグ アンド ドロップします。

［ウォッチ リスト］タブへ追加される項目は、既に追加されている項目の一覧の最後部に表示されます。

▶▶ ファイルまたはテーブルを分析するには

分析では統計情報を報告します。これにより、最適化する必要がある項目を決定できます。

1 ［ウォッチ リスト］タブで、項目をクリックして選択します。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。

2 次のいずれかを実行します。

- ［ウォッチ リスト］タブで、右上にある （**分析**）アイコンをクリックします。
- 選択した項目を右クリックし、［**ファイルの分析**］をクリックします。

［ウォッチ リスト］タブでは、対象項目の各列を更新することで分析結果を報告します。最初に選択された項目の場合、［詳細］タブにはファイルを最適化するために必要な空きディスク容量など、その項目の詳細な状態が表示されます。

▶▶ ファイルまたはテーブルを最適化するには

1 ［ウォッチ リスト］タブで、項目をクリックして選択します。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。

2 次のいずれかを実行します。

- ［ウォッチ リスト］タブで、右上にある （**最適化**）アイコンをクリックします。
- 選択した項目を右クリックし、［**ファイルの最適化**］をクリックします。

［ウォッチ リスト］タブでは、対象項目の各列を更新することで最適化の結果を報告します。1 つの項目を選択していた場合は、下にある［詳細］タブにその項目の詳細な状態が表示されます。複数の項目を選択していた場合は、最初に選択した項目についての状態が表示されます。

▶▶ 分析や最適化を取り消すには

分析または最適化の実行中に、［ウォッチ リスト］内のファイルを選択し、右側にある［最適化の取り消し］アイコン をクリックします。表示される確認のダイアログで、［はい］をクリックします。Defragmenter は［はい］がクリックされるまで操作を続けます。

もう 1 つの方法として、左側にある［すべての最適化の取り消し］アイコン をクリックすることもできます。このアイコンでは、すべての分析および最適化の活動を停止でき、右側のアイコンでは、ウォッチ リストにある、分析または最適化が行われている個々のファイルについて活動を停止できます。

最適化の取り消し後、［ウォッチ リスト］タブ内の項目のうち、状態が " 分析 : 完了 " または " 最適化 : 完了 " で新しい統計情報が表示されるものもあれば、" 分析 : キャンセル " または " 最適化 : キャンセル " で空の統計情報が表示されるものもあるかもしれません。最適化を取り消した項目の場合、［詳細］タブでは取り消されるまでの所要時間が表示されます。

取り消しが成功したということは、分析が停止したか、ファイルまたはテーブルの断片化が変わっていないことを意味します。

メモ すべての操作は、コマンド プロンプトで開始したものであっても、Defragmenter ウィンドウから取り消すことができ、その逆も同様です。

▶▶［ウォッチ リスト］タブからファイルまたはテーブルを削除するには

1 ［ウォッチ リスト］タブで、項目をクリックして選択します。Ctrl キーを使用して項目を個別にクリックするか、Shift キーを使用して範囲（隣接する複数項目）を選択します。Ctrl + A キーを使用して全項目を選択することもできます。

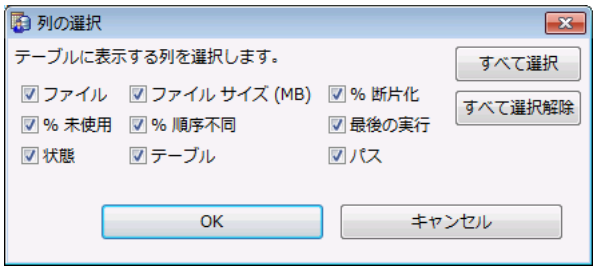
2 次のいずれかを実行します。

- ［ウォッチ リスト］タブで、右上にある （**ファイルの削除**）アイコンをクリックします。
- 選択した項目を右クリックし、［**ファイルの削除**］アイコンをクリックします。

▶▶ 表示する列を選択するには

［ウォッチ リスト］タブの各列の表示 / 非表示を設定することができます。

1 ［ウォッチ リスト］タブで、右上にある （表示する列の選択）アイコンをクリックします。



2 デフォルトでは、すべての列が選択されています。必要に応じて選択を変更し［OK］をクリックするか、あるいは［**キャンセル**］をクリックします。

▶▶ 分析または最適化の詳細を表示する、または非表示にするには

［クリックして詳細を非表示にする］アイコン をクリックすると、［詳細］タブの表示 / 非表示を切り替えることができます。

Defragmenter のコマンド ライン インターフェイス

CLI ツールの dbdefrag では、ウォッチ リストがないことを除き、GUI バージョンと同じ機能を提供します。どちらのインターフェイスからでも最適化の開始、状況の確認、または取り消しが可能です。

次の表では、コマンド ライン オプションについて説明します。

コマンド	説明
dbdefrag ファイル	1 つ以上のファイルの最適化を開始します。アスタリスク（*）をワイルドカード文字として使用することができます。処理されたレコードとキーの数を表示します。Ctrl + C キーで処理を取り消すことができます。ファイルは、パス名以外にも構文 brtv:// **ユーザー** @ **ホスト** / **データベース名** ? **パラメーター** を用いた URI として指定することもできます。これについては、「データベース URI」で説明しています。
dbdefrag -background ファイル	1 つ以上のファイルの最適化をバックグラウンド処理として開始します。アスタリスク（*）をワイルドカード文字として使用することができます。
dbdefrag -cancel ファイル	現在実行中の 1 つ以上のファイルの分析または最適化を取り消します。アスタリスク（*）をワイルドカード文字として使用することができます。その最適化をバックグラウンドで実行している場合を除き、別のプロンプトで実行する必要があります。
dbdefrag -cancelall	すべてのアクティブな分析または最適化操作を取り消します。その最適化をバックグラウンドで実行している場合を除き、別のプロンプトで実行する必要があります。 **メモ** : すべての操作は、Defragmenter GUI ウィンドウで開始したものであっても、Defragmenter のコマンド プロンプトから取り消すことができ、その逆も同様です。
dbdefrag -status ファイル	1 つ以上のファイルの最適化の状態を表示します。アスタリスク（*）をワイルドカード文字として使用することができます。 最後に完了した最適化について、以下の情報が表示されます。 • 状態 : 完了 • 開始 : *yyyy-mm-dd hh:mm:ss* • 終了 : *yyyy-mm-dd hh:mm:ss* • 所要時間 : *n*h:*nn*m:*nn*s 現在実行している最適化については、以下の情報が表示されます。 • 最適化の状態 : 進行中 • *n*/*n* 個のレコードを処理しました（*n*%） • *n*/*n* 個のキーを処理しました（*n*%） 取り消された最適化については、以下の情報が表示されます。 • 状態 : 最適化の取り消し • 所要時間 : *n*h:*nn*m:*nn*s エンジンが再起動されるまで、ファイルの状態は引き続き使用できます。エンジンを再起動後、状態の情報は Zen イベント ログ（zen.log）で確認できます。
dbdefrag -analyze ファイル	1 つ以上のファイルの断片化の統計情報を表示します。アスタリスク（*）をワイルドカード文字として使用することができます。以下の情報が返ります。 • % 断片化 • % 未使用 • % 順序不同 • ファイル サイズ（MB） • 最適化に必要な推定ディスク領域（MB） ファイルを分析することによって、そのファイルの内容が変更されることはありません。

コマンド	説明
dbdefrag -ignore	ファイル処理中のエラーを無視します。このオプションはどのコマンドにも追加することができます。これは、複数のファイルを分析または断片化する場合に役に立ちます。-ignore オプションを使用しない場合、分析または最適化されるファイルのうちの 1 つがエラーを返すと、エラーが記録され、残りのファイルの分析と最適化の操作が停止します。-ignore オプションを使用すると、エラーがログに記録され、残りのファイルについて引き続き分析および最適化が行われます。
dbdefrag -help	コマンド オプションの一覧を表示します。

dbdefrag の使用

次の例では、サイズが大きい .mkd ファイルで dbdefrag を使用する場合の一連の手順を示します。

1 分析

```
C:¥ProgramData¥Actian¥Zen¥Examples>dbdefrag -analyze mybtrievefile.mkd
     断片化 : 9%
     未使用 : 4%
   順序不同 : 100%
ファイル サイズ : 713 MB
```

2 最適化

```
C:¥ProgramData¥Actian¥Zen¥Examples>dbdefrag mybtrievefile.mkd
1125 / 1195242 個のレコードを処理しました (0%)
87021 / 1195242 個のレコードを処理しました (7%)
170910 / 1195242 個のレコードを処理しました (14%)
255393 / 1195242 個のレコードを処理しました (21%)
339805 / 1195242 個のレコードを処理しました (28%)
404202 / 1195242 個のレコードを処理しました (33%)
487928 / 1195242 個のレコードを処理しました (40%)
572585 / 1195242 個のレコードを処理しました (47%)
655802 / 1195242 個のレコードを処理しました (54%)
716472 / 1195242 個のレコードを処理しました (59%)
800406 / 1195242 個のレコードを処理しました (66%)
883729 / 1195242 個のレコードを処理しました (73%)
947475 / 1195242 個のレコードを処理しました (79%)
1032061 / 1195242 個のレコードを処理しました (86%)
1116015 / 1195242 個のレコードを処理しました (93%)
1185831 / 1195242 個のレコードを処理しました (99%)
1195242 / 1195242 個のレコードを処理しました (100%)
1 / 2 個のキーを処理しました (50%)
2 / 2 個のキーを処理しました (100%)
最適化は完了しました。
所要時間 : 0h:24m:57s
```

3 別のプロンプトからの状態の確認、またはバックグラウンドで実行した場合の状態の確認

```
C:¥ProgramData¥Actian¥Zen¥Examples>dbdefrag -status mybtrievefile.mkd
最適化の状態 : 進行中
1053373 / 1195242 個のレコードを処理しました (88%)
0 / 2 個のキーを処理しました (0%)
```

4　最適化後の状態の確認

```
C:¥ProgramData¥Actian¥Zen¥Examples>dbdefrag -status mybtrievefile.mkd
    状態： 完了
    開始： 2014-08-08 11:08:28
    終了： 2014-08-08 11:33:25
所要時間： 0h:24m:57s
```

5　分析の再実行

```
C:¥ProgramData¥Actian¥Zen¥Examples>dbdefrag -analyze mybtrievefile.mkd
    断片化： 0%
    未使用： 0%
   順序不同： 27%
ファイル サイズ： 682 MB
```

パフォーマンス カウンターの監視

Zen サーバーの Windows 版では、Windows パフォーマンス モニターで使用するパフォーマンス カウンターを提供します。(Zen のパフォーマンス カウンターがサポートされるのは、Windows Vista 以降のオペレーティング システムのみです。) パフォーマンス カウンターはデータベース エンジンの状態や動作を測定します。これによりアプリケーションのパフォーマンスを分析することができます。Windows パフォーマンス モニターは、指定の時間間隔でパフォーマンス カウンターの現在の値を要求します。

Zen は、パフォーマンス モニターで表示するためだけにデータを提供しており、カウンターのプロパティは変更できません。パフォーマンス モニターは以下のアイテムを制御します。

- **データの表示形式**。折れ線グラフ (デフォルト)、ヒストグラムおよびテキスト レポートの 3 種類の形式で表示されます。
- **表示フィールド**。[最新]、[平均]、[最小] および [最大] というラベルが付けられています。
- **値の表示用のスケール**。Zen は各カウンターに対してデフォルトのスケールを提供します。パフォーマンス モニターを使用すれば、個々のカウンターのスケールを変更することができます。「カウンターのスケールを変更するには」を参照してください。

カウンターの値は、呼び出し元を問わず、データベース エンジンへのすべての呼び出しを反映します。つまり、MicroKernel エンジン、リレーショナル エンジン、ネイティブな Btrieve アプリケーション、ユーティリティなど、すべてがカウンターの値の一因となります。カウンターの値はすべてのファイルを対象に収集されます。ファイルごとのカウンターは現在サポートされていません。

パフォーマンス カウンターの使用は、主にアプリケーション開発者およびその他の技術スタッフ向けの高度な機能です。Windows パフォーマンス モニターおよび一般的なカウンターの使用の詳細については、Microsoft の関連ドキュメントを参照してください。

インストール時の登録

デフォルトで、Zen のインストールでは Zen パフォーマンス カウンターをパフォーマンス モニターに登録します。このカウンターはインストール完了後に使用できるようになります。

お客様のニーズへの対応のため、本章で説明されていない Zen コレクター セットまたはカウンターを追加インストールすることになる可能性もあります。そのような場合は、Windows パフォーマンス モニターで提供されるコレクター セットまたはカウンターの説明を参照してください。「カウンター セットまたは個別のカウンターをモニターへ追加する」を参照してください。

データ コレクター セット

データ コレクター セットは、複数のカウンターを 1 つのコンポーネントにまとめています。このコンポーネントを使用してパフォーマンスを再確認したり記録したりすることができます。Zen では以下のデータ コレクター セットを提供します。

- 「Zen MicroKernel Btrieve Operations」
- 「Zen MicroKernel Cache」
- 「Zen MicroKernel I/O」
- 「Zen MicroKernel Locks and Waits」
- 「Zen MicroKernel Transactions」
- 「Zen ページ サーバーの活動状況」

Zen MicroKernel Btrieve Operations

これらのカウンターは、Btrieve API に関するクライアント アプリケーションの動作の特徴を示すために役立ちます。カウンターは、指定した時点にデータベース エンジンで処理されたオペレーションの種類を報告します。

『*Btrieve API Guide*』の 「Btrieve API オペレーション」も参照してください。

表 42　MicroKernel Btrieve オペレーション用のカウンター

カウンター	説明	典型的な使用法
Btrieve Close Operations/sec	Btrieve Close オペレーションの 1 秒あたりの数。	クライアント アプリケーションの動作を調査します。問題を解決する最初のステップとして、Btrieve オペレーションに関する動作の分析に役立つ可能性があります。
Btrieve Get/Step Operations/sec	Btrieve Get および Step オペレーションの 1 秒あたりの数。 各種 Get および Step オペレーションについては、『*Btrieve API Guide*』の「Btrieve API オペレーション」を参照してください。	
Btrieve Open Operations/sec	Btrieve Open オペレーションの 1 秒あたりの数。	
Btrieve Records Deleted/sec	1 秒あたりに削除された Btrieve レコード数。	
Btrieve Records Inserted/sec	1 秒あたりに挿入された Btrieve レコード数。	
Btrieve Records Updated/sec	1 秒あたりに更新された Btrieve レコード数。	
Change Operations/sec	データ ファイルを変更する Btrieve オペレーションの 1 秒あたりの数。	
Operations/sec	1 秒あたりのオペレーション数。	

Zen MicroKernel Cache

データベース エンジンは 2 つのレベルのメモリ キャッシュ システムを使用してデータ操作のパフォーマンスを向上させます。2 つのキャッシュは「L1（レベル 1）キャッシュ」と「L2（レベル 2）キャッシュ」と呼ばれます。

ユーザーの要求に応じエンジンがディスクからページを読み取る頻度が高くなるほど、パフォーマンスは低下します。これらのカウンターを使用すると、データベース エンジンによるディスク読み取りの回避状況をまとめて表示し、キャッシュ サイズの設定に対してなんらかの変更が必要かどうかを判断することができます。

すべてのデータが L1 キャッシュに格納された場合は最高のパフォーマンスが得られます。ただし、データ ファイルのサイズによっては、L1 キャッシュにすべてのデータ ページを保持できないこともあります。このため、データベース エンジンは第 2 レベルのキャッシュ（L2 キャッシュ）も使用します。L2 キャッシュのページは圧縮形式で格納されるため、より多くのページをメモリに入れることができます。そのため、L1 キャッシュよりもパフォーマンスは低下しますが、データベース エンジンがディスクからページを読み取る状況と比べればパフォーマンスは高くなります。

「データベースのメモリ キャッシュの理想的なサイズを計算するには」も参照してください。

表 43　MicroKernel キャッシュ用のカウンター

カウンター	説明	典型的な使用法
Cache Hit Ratio	L1 または L2 キャッシュからの、最近のキャッシュ アクセスのヒット率（%）。	
L1 Cache Discards/sec	新しいページ用の領域を確保するために、L1 キャッシュから 1 秒あたりに破棄されたページ数。これは、L1 キャッシュがいっぱいになった場合にのみ発生します。L1 キャッシュから L2 キャッシュへ移動されたページは含まれません。	

表 43　MicroKernel キャッシュ用のカウンター

カウンター	説明	典型的な使用法
Level 1 Cache Dirty Percentage	ダーティ ページを含む使用中の L1 キャッシュの割合（%）。	頻繁にアクセスされるページが継続的にキャッシュの外へ出されるかどうかを判断するのに役立ちます。そのようなページが継続的にキャッシュの外へ出されるとパフォーマンスが低下する可能性があります。 ディスクへ書き込まれていない変更を含むダーティ ページは、L1 キャッシュに存在しているだけかもしれません。大量に書き込みが行われる状況下では、L1 キャッシュにダーティ ページが含まれる可能性が高くなります。これによってページが強制的に L1 キャッシュから出され、L2 キャッシュを設定していれば L2 キャッシュへ入れられます。L2 キャッシュを設定していなければ L1 キャッシュの外へ完全に出されてしまいます。 データベース エンジンは、指定した間隔で、または L1 キャッシュがほぼいっぱいになったときにダーティ ページをディスクへ書き込みます。ページが頻繁にディスクへ書き込まれると、パフォーマンスに悪影響を及ぼす可能性があります。 L1 キャッシュのサイズを調整してダーティ ページの割合が高くならないようにすれば、パフォーマンスを向上させることができます。「キャッシュ割当サイズ」も参照してください。
Level 1 Cache Hits/sec	1 秒あたりの L1 キャッシュ ヒット数。	データベース エンジンが、要求されたページを L1 キャッシュで見つけることができたかどうかを判断するのに役立ちます。ヒット率が高くなると、データベース エンジンが L2 キャッシュまたは物理記憶域へアクセスすることなく L1 キャッシュでページを見つけていることを示します。
Level 1 Cache Hit Ratio	最近の L1 キャッシュ アクセスのヒット率（%）。	
Level 1 Cache Misses/sec	1 秒あたりの L1 キャッシュ ミス数。	
Level 1 Cache Usage Percent	現在使用中の L1 キャッシュの割合（%）。	L1 キャッシュのサイズをアプリケーションに応じて調整する場合に役立ちます。たとえば、サイズが小さなデータ ファイルまたは主に読み取り専用のデータ ファイルを使用するアプリケーションの場合、デフォルトで設定された L1 キャッシュはそのデータ ファイルだけではいっぱいにならないかもしれません。L1 でメモリが使用されなくても、オペレーティング システムまたはその他のアプリケーションがその未使用分のメモリを使用することはできません。 L1 キャッシュ サイズを適宜変更すれば、メモリをオペレーティング システムへ解放することができます。逆に、データベース全体をメモリに入れておきたい場合は、L1 キャッシュ サイズの設定が妥当かどうかを知るためにこの値を監視することができます。
Level 2 Cache Hits/sec	1 秒あたりの L2 キャッシュ ヒット数。	データベース エンジンが、要求されたページを L2 キャッシュで見つけることができたかどうかを判断するのに役立ちます。ヒット率が高くなると、データベース エンジンが物理記憶域へアクセスすることなく L2 キャッシュでページを見つけていることを示します。
Level 2 Cache Hit Ratio	最近の L2 キャッシュ アクセスのヒット率（%）。	
Level 2 Cache Misses/sec	1 秒あたりの L2 キャッシュ ミス数。	

表 43　MicroKernel キャッシュ用のカウンター

カウンター	説明	典型的な使用法
Level 2 Cache Raw Size	L2 キャッシュの現在のサイズ（バイト）。	任意指定の L2 キャッシュ サイズを決定するために役立ちます。 L2 キャッシュは「MicroKernel の最大メモリ使用量」用に設定される要素の 1 つです。その設定は、総物理メモリに対してデータベース エンジンが消費できるメモリの割合を指定します。これには、データベース エンジンによる、L1 キャッシュ、L2 キャッシュおよびその他すべてのメモリの使用量が含まれます。 「MicroKernel の最大メモリ使用量」の設定が 0 以外の場合、L2 キャッシュのサイズはその設定によるメモリ制限内に収まります。L2 キャッシュはシステムのメモリ消費を監視し、必要に応じて自身のサイズを変更します。L2 キャッシュによって使用されるメモリは、オペレーティング システムによってスワップ アウトされることもあります。
Level 2 Cache Usage	現在使用中の L2 キャッシュの量（バイト）。	
Level 2 Cache Size Relative to Memory	物理メモリに対する L2 キャッシュ サイズの割合（%）。	システム メモリに対する L2 キャッシュの使用割合を示します。
Level 2 Cache Usage Percent	現在使用中の L2 キャッシュの割合（%）。	現時点における L2 キャッシュの使用割合を示します。

Zen MicroKernel I/O

このセットのカウンターは、データベース エンジンによる、物理記憶域へのデータの読み取りおよび書き込み状況を把握するのに役立ちます。カウンターによって報告されるページとはデータ ファイル ページです。これらのカウンターでは、アーカイブ ログまたはトランザクション ログ向けに使用されるファイルのページ用のデータは報告しません。

『*Zen Programmer's Guide*』の「ページ」も参照してください。

表 44　MicroKernel 入力 / 出力用のカウンター

カウンター	説明	典型的な使用法
Pages Read/sec	1 秒あたりにディスクから読み取られたページ数。	データベース エンジンによる、物理記憶域へのデータの読み取りおよび書き込み状況を測定します。
Pages Written/sec	1 秒あたりにディスクに書き込まれたページ数。	

Zen MicroKernel Locks and Waits

クライアントの要求は、リソースが使用可能になるのを待つため遅延することがあります。これらのカウンターを使用すれば、リソースが使用可能になるまでクライアント要求を待つ必要があるデータベース リソースのタイプを特定することができます。同様に、これらのカウンターでは複数のクライアントがリソースへアクセスしたときのデータベース エンジンの動作を観察することができます。値がクライアント数に近いまたは等しい場合は、同じリソースに対して競合している可能性があることを示しています。これらの競合を低減するための是正措置を行えば、反応性を改善することができます。

「Waits on Page Buffers」および「Waits on Page Reads」カウンターはグローバル リソースを待機します。その他のカウンターはすべて複数のクライアントに適用しますが、各クライアントはそれぞれ異なるリソースを待機している場合があります。

『*Zen Programmer's Guide*』の「データ整合性」および「複数のクライアントのサポート」を参照してください。

表 45　MicroKernel ロックおよび待機用のカウンター

カウンター	説明	典型的な使用法
Client Record Locks	クライアントによって明示的にロックされたレコード数。	クライアント アプリケーションの作業負荷を調査します。
Waits on Active Reader Lock	アクティブなリーダー ロックで待機中のクライアント数。複数のクライアントが同時にアクティブなリーダー ロックを保持できます。ただし、アクティブなリーダー ロックとアクティブなライター ロックは排他的に使用されます。そのため、1 つのクライアントがアクティブなリーダー ロックを保持すると、どのクライアントもアクティブなライター ロックを取得することはできません。1 つのクライアントがアクティブなライター ロックを保持すると、複数のクライアントがアクティブなリーダー ロックを取得することはできません。各ファイルがそれ自身のリーダー（およびライター）ロックを持っています。 「Waits on Active Writer Lock」カウンターも参照してください。	
Waits on Active Writer Lock	アクティブなライター ロックで待機中のクライアント数。一度に 1 つのクライアントだけがアクティブなライター ロックを保持できます。各ファイルがそれ自身のライター（およびリーダー）ロックを保持します。 「Waits on Active Reader Lock」カウンターも参照してください。	
Waits on File Locks	現在、ファイル ロックで待機中のクライアント数。	

表 45　MicroKernel ロックおよび待機用のカウンター

カウンター	説明	典型的な使用法
Waits on Page Buffers	ページ バッファーが利用可能になるのを待機中のクライアント数。要求に応えるために利用できるページがない場合は、MicroKernel がページを利用できるようになるまで、その要求はブロックされます。	データベース エンジンが、キャッシュのページ バッファーを使用できるかどうかを示します。この値をメモリ キャッシュ関連のカウンターと一緒に使用して、キャッシュのサイズが作業負荷に対して適切かどうかを確認します。キャッシュ サイズを大きくすると使用可能な総ページ数も増えます。これにより、ページ バッファーへの待機を減らすことができます。 キャッシュにページがない場合は、以下の 3 つ状況によってこの値が急増する可能性があります。 • データ ファイルが最近開かれた • データ ページへ初めてアクセスする、またはそのデータ ページへアクセスすることがほとんどない • メモリ キャッシュが、頻繁にアクセスおよび変更されるすべてのページを含めることができないほど小さい可能性がある 1 番目と 2 番目の項目によって値が急増するのは、ファイルへのアクセスが初めてであるため回避することはできません。3 番目の状況はキャッシュ サイズを大きくすることで回避できます。キャッシュがいっぱいでキャッシュ ミス数が多くなると、メモリ キャッシュが小さすぎて、頻繁にアクセスおよび変更されるすべてのページを含めることができない可能性があります。 「MicroKernel キャッシュ用のカウンター」も参照してください。
Waits on Page Locks	現在、ページ ロックで待機中のクライアント数。	クライアント アプリケーションの作業負荷を調査します。
Waits on Page Reads	ディスクからのページの読み取りを待機中のクライアント数。1 つのクライアントが既にページの読み取り処理を行っている場合、ほかのクライアントは処理中の読み取りが完了するまで待機しなければなりません。	同時に同じファイルの同じページへの読み取りを試みるクライアント数を測定します。
Waits on Record Locks	現在、レコード ロックで待機中のクライアント数。	クライアント アプリケーションの作業負荷を調査します。

Zen MicroKernel Transactions

これらのカウンターは、トランザクションに関するクライアント アプリケーションの動作を理解するために役立ちます。たとえば、変更が多く処理に長時間かかる少数のトランザクションと、短時間で処理が完了する多くのトランザクションとでは動作が異なります。

『*Btrieve API Guide*』の「Begin Transaction（19 または 1019）」、「End Transaction（20）」、および「Abort Transaction（21）」、また『*Zen Programmer's Guide*』の「MicroKernel エンジンの基礎」も参照してください。

表 46　MicroKernel トランザクション用のカウンター

カウンター	説明	典型的な使用法
System Transactions in Progress	進行中のシステム トランザクション数。システム トランザクションは特別類のトランザクションで、データ ファイルの変更は準備された後、ファイルに保存されます。	システム トランザクションの発生頻度が高すぎるまたは低すぎるかどうかを判断するために役立ちます。 データベース エンジンは、システム トランザクション中に変更をデータ ファイルへ書き込みます。システム トランザクションが発生する頻度は、「起動時間制限」および「オペレーション バンドル制限」という 2 つのサーバー データの整合性設定プロパティによって決められます。また、L1 キャッシュの空き領域が少ない場合には、サーバーが自動的に頻度を調整します。 通常、システム トランザクションの実行頻度が高すぎたり低すぎたりするとパフォーマンスに悪影響を及ぼします。一般的には、1 秒あたりのページ書き込み数が増加し、レコードを変更する Btrieve オペレーション数は減少し、またアクティブなライタ ロックを待機するクライアント数は増加する可能性があります。これは特定の作業負荷に対する理想的な間隔を測定するための実験が必要となることもあります。
Transaction Commits /sec	1 秒あたりに実行されたコミット数。	アプリケーション トランザクションのコミット数を測定します。『*Btrieve API Guide*』の「End Transaction (20)」も参照してください。

Zen ページ サーバーの活動状況

このセットのカウンターは、ページ サーバーに関する統計情報を提供します。

表 47　ページ サーバーの活動状況のカウンター

カウンター	説明
Page Server Pass-Thru Operations/sec	ページ サーバーがキャッシュ エンジンから受け取ったパススルー操作の 1 秒あたりの数。
Page Server Page Requests/sec	ページ サーバーがキャッシュ エンジンから受け取った、データ ファイル ページに対する要求の 1 秒あたりの数。
Page Server Invalid Page List Requests/sec	キャッシュ エンジンから受け取った、無効ページ リストに対する要求の 1 秒あたりの数。
Page Server Invalid Page List Piggy-backs/sec	ページ サーバーが他のキャッシュ エンジンの要求に対する応答でピギーバックした無効ページ リストの 1 秒あたりの数。
Page Server Invalid Page List Length	ページ サーバーの無効ページ リストの現在の長さ。
Page Server Invalid Page List Adds/sec	ページ サーバーの無効ページ リストに追加されたコミット済みページの 1 秒あたりの数。重複もカウントします。
Page Server Invalid Page List Removals/sec	ページ サーバーの無効ページ リストから削除されたエントリの 1 秒あたりの数。
Page Server Level 1 Cache Hits/sec	ページ サーバーによる L1 キャッシュ ヒットの 1 秒あたりの数。
Page Server Level 1 Cache Misses/sec	ページ サーバーによる L1 キャッシュ ミスの 1 秒あたりの数。
Page Server Level 1 Cache Hit Ratio	ページ サーバーによる、最近の L1 キャッシュ アクセスのヒット率（%）。
Page Server Level 2 Cache Hits/sec	ページ サーバーによる L2 キャッシュ ヒットの 1 秒あたりの数。
Page Server Level 2 Cache Misses/sec	ページ サーバーによる L2 キャッシュ ミスの 1 秒あたりの数。

表 47　ページ サーバーの活動状況のカウンター

カウンター	説明
Page Server Level 2 Cache Hit Ratio	ページ サーバーによる、最近の L2 キャッシュ アクセスのヒット率（%）。
Page Server Cache Hit Ratio	ページ サーバーによる、L1 または L2 キャッシュからの最近のキャッシュ アクセスのヒット率（%）。
Page Server Level 1 Cache Usage Percent	現在使用中の L1 キャッシュのうち、ページ サーバーが占める割合（%）。
Page Server Level 2 Cache Usage Percent	現在使用中の L2 キャッシュのうち、ページ サーバーが占める割合（%）。

Zen キャッシュ エンジンの活動状況

このセットのカウンターは、Zen キャッシュ エンジンに関する統計情報を提供します。

表 48　キャッシュ エンジンの活動状況用カウンター

カウンター	説明
Cache Engine Pass-Through Ops/sec	キャッシュ エンジンから要求されたパススルー操作の 1 秒あたりの数。
Cache Engine Local Ops/sec	キャッシュ エンジンによって実行されたローカル操作の 1 秒あたりの数。
Cache Engine Page Requests/sec	キャッシュ エンジンから要求されたデータ ファイル ページの 1 秒あたりの数。
Cache Engine Page Invalidations/sec	このキャッシュ エンジンによって無効になったページの 1 秒あたりの数。
Cache Engine FCR Invalidations/sec	このキャッシュ エンジンによって無効になった FCR の 1 秒あたりの数。
Cache Engine Level 1 Cache Usage Percent	現在使用中の L1 キャッシュのうち、キャッシュ エンジンが占める割合（%）。
Cache Engine Level 1 Cache Hits/sec	キャッシュ エンジンによる L1 キャッシュ ヒットの 1 秒あたりの数。
Cache Engine Level 1 Cache Misses/sec	キャッシュ エンジンによる L1 キャッシュ ミスの 1 秒あたりの数。
Cache Engine Level 1 Cache Hit Ratio	キャッシュ エンジンによる、最近の L1 キャッシュ アクセスのヒット率（%）。
Cache Engine Level 2 Cache Usage Percent	現在使用中の L2 キャッシュのうち、キャッシュ エンジンが占める割合（%）。
Cache Engine Level 2 Cache Hits/sec	キャッシュ エンジンによる L2 キャッシュ ヒットの 1 秒あたりの数。
Cache Engine Level 2 Cache Misses/sec	キャッシュ エンジンによる L2 キャッシュ ミスの 1 秒あたりの数。
Cache Engine Level 2 Cache Hit Ratio	キャッシュ エンジンによる、最近の L2 キャッシュ アクセスのヒット率（%）。
Cache Engine Cache Hit Ratio	キャッシュ サーバーによる、L1 または L2 キャッシュからの最近のキャッシュ アクセスのヒット率（%）。

Windows パフォーマンス モニターの使用

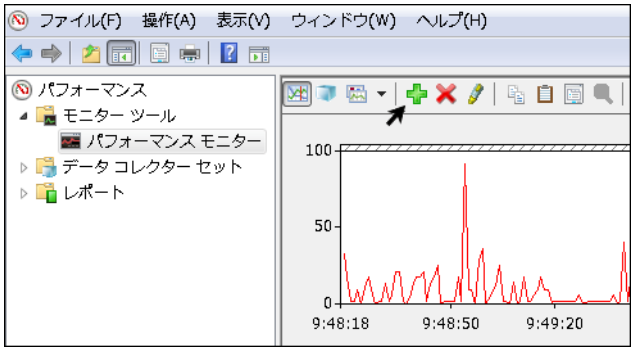
このトピックでは、Windows パフォーマンス モニターの基本的な使用手順について説明し、Zen パフォーマンス カウンターを使用できるようにします。Windows パフォーマンス モニターの使用に関する詳細については、Microsoft のドキュメントを参照してください。

以下に示す手順は、Zen のインストールによって Zen パフォーマンス カウンターが Windows パフォーマンス モニターに登録されていることを前提としています。

▶▶ Zen データ コレクター セットを表示する

1 Windows パフォーマンス モニターを起動します。このツールの起動手順はオペレーティング システムによって異なりますが、一般的には［**コントロール パネル**］>［**管理ツール**］から起動することができます。また、［ファイル名を指定して実行］ウィンドウ（［**スタート**］>［**ファイル名を指定して実行**］を選択）で、「perfmon」と入力し［OK］をクリックして起動することもできます。

2 左ペインのツリーから［パフォーマンス モニター］をクリックし、右ペイン上部のツールバーにあるプラス記号（＋）をクリックします。

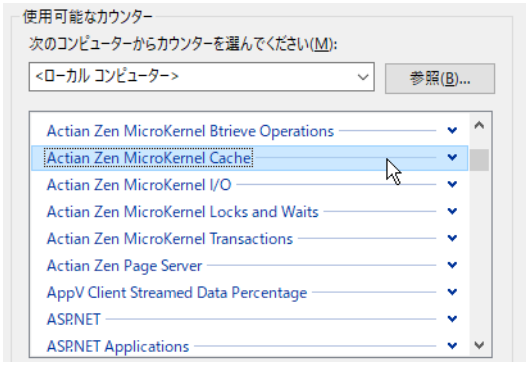


3 ［使用可能なカウンター］セクションのカウンター リスト内で、Zen データ コレクター セットへスクロールします。

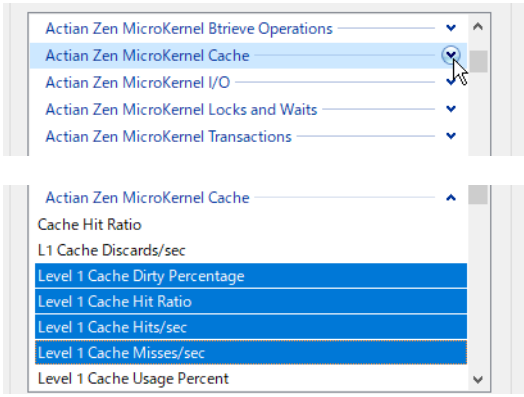
▶▶ カウンター セットまたは個別のカウンターをモニターへ追加する

1 次のいずれかを実行します。

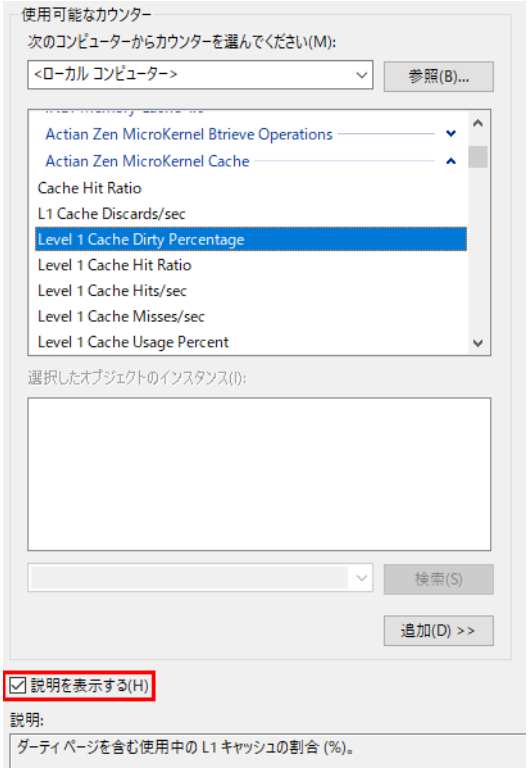
- カウンター セットとしてまとめて追加するには、［使用可能なカウンター］リストから希望のカウンター セット項目をクリックします。



- カウンターを個別に追加するには、そのカウンターが属するカウンター　セットを展開し（カウンター セットの項目をダブルクリック、または右端の下矢印をクリック）、その中から対象のカウンターをクリックします。



カウンターの説明を見るには、［**説明を表示する**］オプションを選択してください。

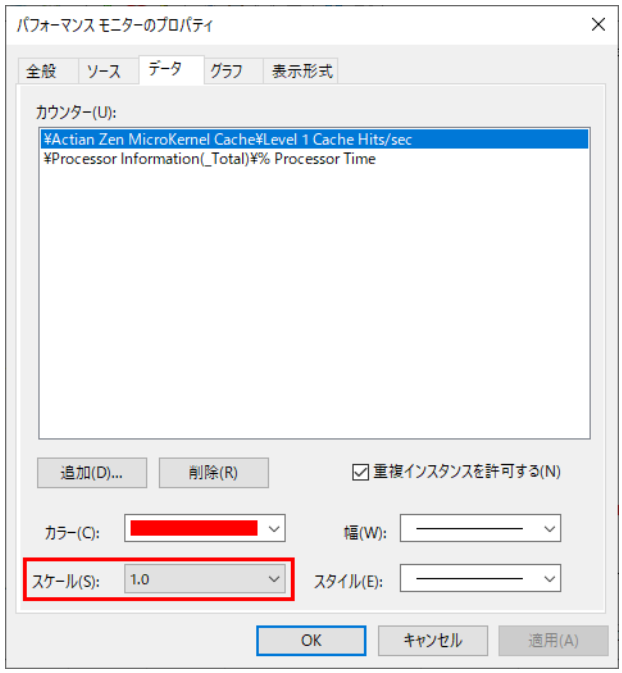


2　［**追加**］をクリックしてから［OK］をクリックします。

▶▶ カウンターのスケールを変更するには

1　メイン ウィンドウの下部にあるリストにカウンターが追加されたら、それを右クリックして［**プロパティ**］を選択できます。

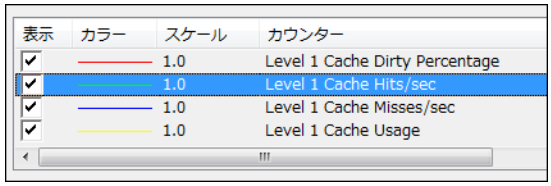
2　［データ］タブで、［スケール］リストから値を選択します。

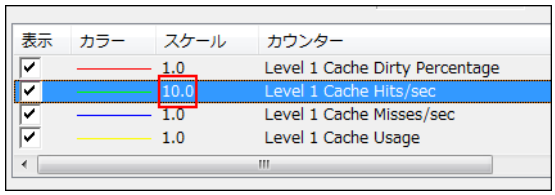


同じグラフ上に 2 つ以上のカウンターを表示する場合に、それらカウンターの出力数値の差が非常に大きい場合は、適切に表示されるようスケールを調整する必要があるかもしれません。データがグラフ化される前に、カウンターの出力値にスケール値を掛けます。たとえば、1 番目のカウンターの出力値が 53 と 99、2 番目のカウンターの出力値が 578 と 784 であるとします。1 番目のカウンターのスケールに 10 を設定することで、530 および 990 と出力されるようにするこができます。これにより、両カウンターのデータをより同等（530、990、578 および 784）に見ることができます。

3 ［OK］をクリックします。

［スケール］フィールドの値は、元の値（この例では "1"）から新しい値（この例では "10"）に変更されています。





4 ［グラフ］タブで、［垂直スケール］セクションの［**最大**］および［**最小**］フィールドに必要な値を設定します。

グラフ化される対象が非常に小さい（または非常に大きい）場合は垂直スケールを変更することができるので、グラフが見やすくなります。たとえば、カウンターの出力値が常に 20 未満の場合は、垂直スケールの最大を 20、最小を 0 に変更することができます。

5 ［OK］をクリックします。

ライセンスの使用状況の監視

Zen 製品では、エディションに応じて異なるライセンス モデルを使用します。Zen Enterprise Server および Workgroup は同時ユーザー カウント ライセンスに基づいてライセンスされます。Cloud Server は容量ベース ライセンスに基づいてライセンスされます。

ユーザー カウント ライセンス	ユーザー カウント ライセンス モデルは、多くのユーザーまたはデバイスが別々のデスクトップから頻繁にレコードを追加、更新および削除するような従来型のクライアント / サーバー アプリケーションに適しています。各製品キーにはそれぞれライセンスされたユーザー数が定められています。ユーザー数により、指定された接続数が、Zen データベース エンジンへ同時接続できます。ユーザー数は、ネットワーク アドレスでカウントされます。
容量ベースのライセンス	容量ベース モデルは、ユーザー数ではなくデータベース サーバー処理の作業量に重点を置くようになりました。このライセンス モデルは容量に基づくもので、サービス管理、SaaS（サービスとしてのソフトウェア）、またはその他の多重環境におけるライセンス執行に対応しています。たとえば、Cloud Server のインスタンスごとに、使用セッション数および使用データの両方に対して容量制限が設けられています。

ライセンスの使用状況に関する監視の大部分が、ユーザー数、セッション数、または使用中データを対象とします。たとえば、これらの現在の値を調べる、現在の値を増やす、または必要な容量をさらに調べることができます。

ライセンスの使用状況の監視に使用する 4 つの主要なユーティリティは、Monitor（「リソース使用状況の監視」を参照）、「Capacity Usage ビューアー」、「License Administrator」、および 「Notification Viewer」です。

Capacity Usage ビューアー

ZenCC では、すべてのデータベース エンジンの同時セッション数とデータ使用量を監視する Capacity Usage ビューアーを提供します。このビューアーは特に、Zen Enterprise Server から Zen Cloud Server への移行を検討する際に役立ちます。なぜなら、これら 2 つのエディションにはライセンス方法に違いがあるからです。

Capacity Usage ビューアーには 2 つのグラフが含まれています。1 つは同時セッション数を表し、もう 1 つはデータの使用量を表します。各グラフには**使用量のレベル バー**というグラフを横断する横線を表示させることができます。これは業務上の正常とする使用量と異常とする使用量を判断するのに役立ちます。Capacity Usage ビューアーは使用量のピークに関する統計情報も表示します。

このグラフは毎日記録されるピーク値を使用します。エンジンが使用されない日については、ゼロの値が使用されます。グラフの生成には最低 2 日間のデータが必要です。そのデータがなければ、Capacity Usage ビューアーはエラー メッセージを表示します。

ここでは、以下の項目について説明します。

- 「Capacity Usage ビューアーにアクセスするには」
- 「Capacity Usage ビューアーの GUI」
- 「拡大」

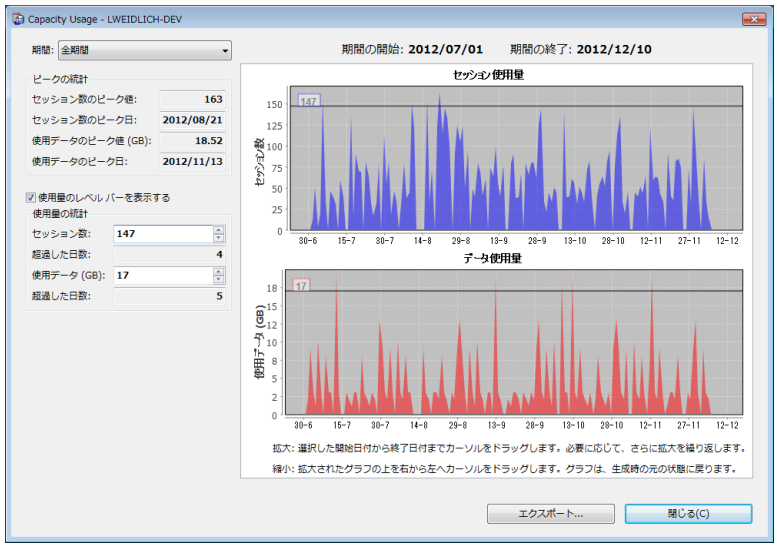
⏩ Capacity Usage ビューアーにアクセスするには

ZenCC で、検査対象のエンジンを右クリックし、［Capacity Usage ビューアー］を選択します。

Capacity Usage ビューアーの GUI

次の図は Capacity Usage ビューアーを表しています。図の下の表は各機能の説明です。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

図 6 Capacity Usage ビューアーのユーザー インターフェイス



GUI のオブジェクト	説明
タイトル バー	選択したエンジンを示します。
統計情報インジケーター	有用な統計情報を表示したり、グラフに表示させるものを選択したりすることができます。「統計情報インジケーター」を参照してください。
期間指定	グラフに表示するデータの対象となる期間の開始日と終了日を表示します。
セッション使用量のグラフ	選択した期間中に同時発生したセッション数をグラフで表示します。
データ使用量のグラフ	選択した期間中に使用されたデータ量をグラフで表示します。
使用量のレベル バー	使用量が、選択したレベルを超える頻度を調べることができます。レベル バーの表示オプションが選択されると、各グラフのデフォルト レベル（使用量のピークの 90%）の位置にレベル バーが表示されます。使用量のレベル バーの左側に表示される数値は、そのレベル（セッション数またはデータ量）を示します。この使用量のレベル バーは必要に応じて移動させることができます。レベル バーを移動させるには、スピン ボックスを使用するか、またはカーソルでドラッグします。2 つの使用量レベル バーは互いに独立しています。
拡大縮小手順	グラフの拡大と縮小についての一般的な手順を説明します。手順の詳細については、「拡大」を参照してください。
［エクスポート］ボタン	分析をさらに行うためにデータを保存しておくことが有用と思われた場合、そのデータを .csv ファイルへエクスポートできます。［**エクスポート**］ボタンをクリックすると［**フォルダーの参照**］ダイアログを開きます。ここでデータを保存する場所を選択することができます。

図 7 統計情報インジケーター

GUI のオブジェクト	説明
期間	グラフに表示させるデータの対象期間を選択することができます。ウィンドウを開いたときにデフォルトで表示される期間は、前回そのウィンドウを閉じる前に選択されていた期間です。選択できる期間は次のとおりです。 • 全期間 • 先週 • 過去 30 日以内 • 過去 90 日以内 • 過去 180 日以内 グラフを拡大または縮小して期間を選択することもできます。グラフを拡大または縮小した場合、[**期間**] フィールドには、選択した期間として "**カスタム**" が表示されます。
[ピークの統計] グループボックス	グラフに表示する期間における最大データ使用量と最大同時セッション数の統計情報を表示するフィールドが含まれています。
セッション数のピーク値	**セッション使用量**グラフに表示される期間中に発生した最大同時セッション数を表示します。
セッション数のピーク日	発生した同時セッション数が最大となった日を表示します。最大同時セッション数の発生が複数日あった場合は、最後に発生した日を表示します。
使用データのピーク値 (GB)	**データ使用量**グラフに表示される期間中、一度に使用された最大データ量（GB 単位）を表示します。
使用データのピーク日	使用されたデータ量が最大となった日を表示します。使用データ量が最大となった日が複数日あった場合は、最後に発生した日を表示します。
使用量のレベル バーを表示する	このチェックボックスのオン / オフによって、グラフを横断する使用量レベルバーの表示 / 非表示を切り替えます。
[使用量の統計] グループボックス	使用量のレベル バーを上下に移動させるためのスピン ボックス、および使用量のレベル バーの移動により得られる統計情報を表示するフィードが含まれています。

GUI のオブジェクト	説明
セッション数	**セッション使用量**グラフで、使用量のレベル バーを置く位置に対応する値を設定します。
超過した日数	同時セッション数が、使用量のレベル バーが設定されたレベルを超えた日数を表示します。
使用データ (GB)	**データ使用量**グラフで、使用量のレベル バーを置く位置に対応する値を設定します。
超過した日数	使用データ量が、使用量のレベル バーが設定されたレベルを超えた日数を表示します。

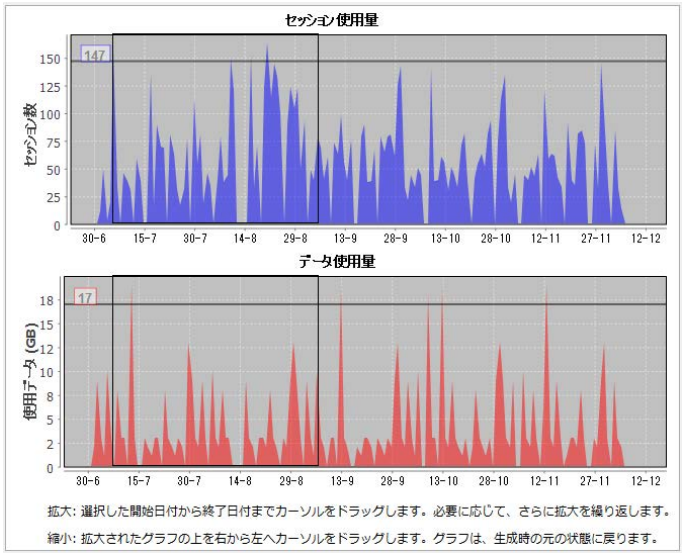
拡大

［**期間**］プルダウン メニューで提供されている期間の選択肢以外の期間について確認する必要がある場合は、グラフのセグメントを選択して拡大することができます。2 つのグラフの一方を拡大すると、もう一方のグラフも同時に拡大されます。2 つのグラフは常に同じ期間が設定されます。

▶▶ グラフを拡大するには

1 表示対象期間の開始位置にカーソルを置きます。

2 マウスの左ボタンを押したまま、表示させたい期間の終了位置までカーソルをドラッグします。選択した部分を囲む四角形の黒い枠線がグラフ内に表示されます。カーソルを移動させると、それに応じてその四角形も拡張されます。



3 マウス ボタンを離します。選択した期間を表示するためにグラフが更新されます。［**期間**］フィールドの設定表示は " **カスタム** " になります。

4 さらに拡大するには上記の手順を繰り返します。

5 縮小するには、カーソルをグラフ内の任意の位置に置き、マウスの左ボタンを押したままカーソルを右から左へドラッグします。これでグラフは元の期間（ウィンドウを開いたときに表示されていた期間）に戻ります。

License Administrator

License Administrator はライセンスのキーの管理に使用するツールです。これについては、『*Zen User's Guide*』の「ライセンス管理」で詳しく説明しています。次の表では、このツールで実施できる一般的な監視作業について簡単に説明し、関連情報の参照先として『*Zen User's Guide*』へのリンクも提供しています。

表 49　License Administrator におけるライセンス使用状況の監視の概要

監視する項目	グラフィカル ユーザー インターフェイスの使用	コマンド ライン インターフェイスの使用
ライセンス情報 (製品、製品キー、キーの状態、ライセンスが適用されるプラットフォーム、ライセンス タイプ、ユーザー数、セッション数、使用データ、ライセンスの失効日、ライセンスをインストールしたベンダー ソフトウェア、およびライセンスが適用されるアプリケーションなど、認証されているキーに対するすべてのライセンス情報)	「キーの情報を表示するには」	「キーの情報を表示するには」
ユーザー数 (製品キーによって許可される、Zen データベース エンジンへの同時接続数)	「合計ユーザー数を調べるには」	
セッション数の制限 (使用許諾契約書に従って許可される最大同時セッション数)	「セッション数の制限を調べるには」	
使用データの制限 (使用許諾契書に従って許可される、同時に開く全データ ファイルの総量)	「使用データの制限を調べるには」	
キーの認証残り回数	「認証残り回数を表示するには」	「認証残り回数を表示するには」

データベース アクセスの監視

Actian Corporation では、Audit for PSQL という、Zen に付属するデータベース レベルでのアクセスを監視するための製品を提供しています。Audit for PSQL を使用すると、次のような監査目的のために、データへの変更を監視することができます。

- だれがレコードにアクセスしたか、または変更を行ったか
- どのようなアクセスまたは変更が発生したか、それはいつ、どこから行われたか
- どのようにして変更が行われたか

Audit for PSQL は詳細な監査証跡を提供するほか、クエリおよび警告機能を提供します。

製品の詳細については、Audit for PSQL のドキュメントを参照してください。

メッセージ ログの見直し

Zen ではメッセージ用のさまざまなログ リポジトリを提供して、トラブルシューティングを支援します。ログは大きく分けて 2 つのカテゴリに分類されます。

- **すべてのメッセージ**：これらのメッセージにはステータス、エラー、警告および情報メッセージが含まれます。これらは、ライセンス管理コンポーネントを含め、あらゆる Zen コンポーネントから発生します。
- **ライセンス　メッセージ**：これらのメッセージは、ライセンスに関する問題を警告し、そのトラブルシューティング情報を提供します。これらはライセンス管理コンポーネントから発生します。

次の表にログ リポジトリの概要を示します。

表 50　Zen メッセージ ログ リポジトリ

リポジトリ	記録元
「Notification Viewer」	ライセンス管理コンポーネント
「オペレーティング システムのイベント ログ」	ライセンス管理コンポーネント（Windows） Zen イベント ログ（Windows）にエラー メッセージを書き込む Zen コンポーネント Zen の全コンポーネント（Linux、macOS、および Raspbian）
「Zen イベント ログ（zen.log）」	Zen の全コンポーネント（Windows）

ライセンス メッセージ

ログ リポジトリのいくつかはライセンス メッセージに重点を置いています。キーが無効と判断された場合、そのキーの状態は「アクティブ」から「検証失敗」に変わります。この状態でも一定の期間内はデータベース エンジンが正常に機能するので、検証失敗を修正するための時間は十分にあります。

その修正猶予期間が終わるまでにエラーが修正されないと、キーの状態は再び無効になります。キーが無効になると、データベース エンジンはデータ ファイルにアクセスできなくなります。

検証失敗への対応を適時に行う必要があるため、キーの状態の変更はできるだけ早く通知されるようになっています。たとえば、1 つのメッセージがすべてのメッセージ リポジトリに記録されます。リポジトリの中で最もわかりやすいのは Zen Notification Viewer です。License Administrator でもキーの状態が表示されます。このため、いつでも検証動作を実行して状態を確認することができます。『*Zen User's Guide*』の「ライセンス管理」を参照してください。

キーの状態の変更

次の表では、キーの状態の変更に応じて返されるメッセージのタイプについて説明します。

表 51　キーの状態の変更に応じたメッセージのタイプ

メッセージのタイプ	キーの状態の変更		シナリオ
	変更前	変更後	
警告	アクティブ	検証失敗	検証動作によってキーの検証失敗が検出されました。たとえば、マシン署名を判別できないためキーと一致しなくなった、または 1 つまたは複数のライセンス コンポーネントをロードできないなどが原因です。 この状態でも一定の期間内はデータベース エンジンが正常に機能するので、検証失敗を修正するための時間は十分にあります。
警告	無効	検証失敗	キーが無効になった原因のいくつかは修正されていますが、まだ解決する必要がある問題が少なくとも 1 つあります。 キーの状態は検証失敗に変わったため、一定の期間内はデータベース エンジンが正常に機能するので、検証失敗の原因を修正することができます。
エラー	検証失敗	無効	検証失敗の原因を修正するために提供された期間が過ぎました。検証失敗の原因となる状態が期限内に修正されませんでした。キーが無効になると、データベース エンジンはデータ ファイルにアクセスできなくなります。
情報	検証失敗	アクティブ	検証失敗となった原因が修正されています。キーは有効になり、データベース エンジンはすべて機能します。
情報	無効	アクティブ	キーが無効になった原因が修正されました。キーは有効になり、データベース エンジンはすべて機能します。

期限切れ状態（一時キーの場合のみ適用）、または非アクティブ状態（前バージョンの Zen で登録されたままのキーに適用）となっているキーに関するメッセージは記録されないので注意してください。

ログ記録の頻度

次の表は、ライセンス メッセージが記録される頻度を特定の動作別に示しています。

表 52　メッセージのログ記録の頻度（起因動作別）

起因動作	ログ記録の頻度	ログ リポジトリ [1]
表 51 で示したキーの変更状況	即時	• Notification Viewer • オペレーティング システムのイベント ログ • Zen のイベント ログ
キーが検証失敗状態のまま	1 日 1 回通知	• Notification Viewer • オペレーティング システムのイベント ログ • Zen のイベント ログ
API 呼び出しによってプログラムから起動した検証動作 『*Zen User's Guide*』の「認証残り回数を表示するには」、『*Distributed Tuning Interface Guide*』の「PvValidateLicenses()」、および『*Distributed Tuning Objects Guide*』の「ValidateLicenses メソッド」を参照してください。	即時	• オペレーティング システムのイベント ログ • Zen のイベント ログ

表 52　メッセージのログ記録の頻度（起因動作別）

起因動作	ログ記録の頻度	ログ リポジトリ [1]
Zen ライセンス サーバーから発生した警告またはエラー メッセージ	即時	• オペレーティング システムのイベント ログ • Zen のイベント ログ
[1] メッセージ ログは一方向の階層に従って記録されます。Notification Viewer へ記録されるライセンス メッセージはオペレーティング システムのイベント ログへ、さらに Zen イベント ログへと記録されます。オペレーティング システムのイベント ログに記録されるライセンス メッセージは、Zen イベント ログにも記録されます。		

Notification Viewer

Notification Viewer は、ライセンス コンポーネントによって記録されるメッセージを表示するツールです。このツールは、注目すべきライセンス メッセージ（表 52 を参照）をわかりやすく通知することを目的としています。

Notification Viewer は、デフォルトで Zen Enterprise Server および Cloud Server（Windows、Linux、macOS の 32 ビットおよび 64 ビット版）および Zen Workgroup（Windows 版）のインストール時に一緒にインストールされます。また、Windows プラットフォームではデフォルトで、Windows を再起動すると Notification Viewer も再起動します。

Windows プラットフォームの場合、この実行可能ファイルの名前は notifyviewer.exe です。これはユーザーに対し、単独で実行するインスタンスを提供します。Notification Viewer が既に実行されている状態で新たに Notification Viewer を起動しようとすると、実行済みの GUI が画面の最前面に表示されます。

Linux、macOS、および Raspbian の場合、このツールは notifyviewer という名前のシェル スクリプトです。このシェル スクリプトは実行するたびに、新たな Notification Viewer インストタンスを起動します。オペレーティング システムを再起動した場合は、Notification Viewer を手動で再起動する必要があります。再起動時、このシェル スクリプトは自動的には実行されません。

コマンド ライン オプション

次のコマンド ライン オプションを使用すると、このツールの実行方法を指定できます。

オプション	説明
（オプションなし）	オプションを指定しないでこのツールを実行する場合は、GUI が表示され、オペレーティング システムがシステム トレイをサポートしている場合はトレイ アイコンが現れます。
-tray	このオプションを指定してツールを実行すると、GUI が表示されずトレイ アイコンを表示します。オペレーティング システムでシステム トレイがサポートされない場合は、GUI が表示されます。

Notification Viewer では、システム トレイ アイコンとグラフィカル ユーザー インターフェイスの 2 つのインターフェイスを提供します。

システム トレイ アイコン インターフェイス

Windows のデフォルトでは、Notification Viewer は GUI を表示しないで起動し、システム トレイ アイコンを表示します。Linux および macOS の場合、Notification Viewer は GUI として起動し、そのディストリビューションがシステム トレイをサポートしている場合は、システム トレイ アイコンを表示します。ツールが実行されると、ライセンス メッセージの監視を開始します。

Notification Viewer が未読のメッセージを検出すると、トレイ アイコンは未読メッセージを示す表示に変わります。「トレイ アイコン」を参照してください。

Notification Viewer は 2 種類のツールヒントも表示します。ツールヒントをマウスでポイントすると、重要な未読メッセージがある場合はそのメッセージ数、未読メッセージの総数が表示されます。メッセージがすべて既読の場合はツールの名前が表示されます。通知する必要があるメッセージを Notification Viewer が検出した場合は、バルーン形式のツールヒントが表示されます。Windows の場合、このツールヒントは直接閉じるか、キーボード ま

たはマウス操作を行わない限り表示され続けます。Linux および macOS の場合、このツール ヒント上をクリックして閉じる必要があります。

ポップアップ メニュー

トレイ アイコンのポップアップ メニューには、[Zen Notification Viewer を開く]（GUI を開く）および [終了]（ツールを終了する）という 2 つの項目があります。このメニューはトレイ アイコンを右クリックすると表示されます。

トレイ アイコン

次の表ではトレイ アイコンの意味を説明します。

アイコン	説明
	Notification Viewer は実行中でライセンス メッセージを監視しています。このアイコンは、すべてのメッセージが開封済み（既読）である正常な状態を示しています。
	Notification Viewer には未読メッセージがあります。 このアイコンは、未読メッセージがすべて開封されない限り表示され続けます。「左パネル」を参照してください。

グラフィカル ユーザー インターフェイス

トレイ アイコンをダブルクリックするか、トレイ アイコンを右クリックして [Zen Notification Viewer を開く] をクリックすると、Notification Viewer GUI を開くことができます。Linux および macOS のデフォルトでは、Notification Viewer は GUI として起動し、システム トレイ アイコンを表示します。起動動作を変更する場合は、notifyviewer シェル スクリプトに -tray オプションを渡します。Linux または macOS ディストリビューションがシステム トレイをサポートしていない場合、Notification Viewer は GUI を表示し、システム トレイ アイコンは表示しません。その場合は、シェル スクリプトを実行して Notification Viewer を起動してください。

この GUI が表示されると、未読メッセージが直ちに GUI へ追加されます。さらに、トレイ アイコンにツールヒントが表示され、アイコンは未読メッセージを示す表示に変わります。

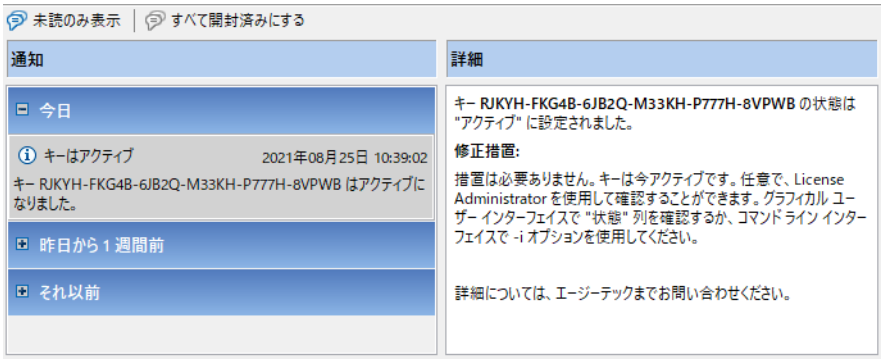
Zen は、通知ファイルにある記録の既読または未読状態をユーザーごとに監視します。つまり、GUI を表示している各ユーザーからはすべてのメッセージが見えますが、メッセージの既読または未読状態はユーザーごと異なるということです。

オペレーティング システムでシステム トレイがサポートされない場合は、Notification Viewer に対して Close コマンドを実行することでこのツールは終了します。システム トレイがサポートされる場合は、画面を閉じることで GUI が非表示になり、ツールが実行中であることを示すトレイ アイコンが表示されます。

ツールバーとパネル

図 8 で示すように、Notification Viewer GUI にはツールバーと 2 つのメイン パネルがあります。

図 8 Notification Viewer の GUI



GUI 要素	説明
ツールバー	2 つのオプションを提供します。 • ［未読のみ表示］（オンとオフで表示を切り替える） • ［すべて開封済みにする］
左パネル	スクロール可能なメッセージの一覧が含まれます。メッセージは一番上に最新のメッセージが来るようソートされ、" 今日 "、" 昨日から 1 週間前 "、" それ以前 " の 3 つのグループに分けて配置されます。各グループ ノードは展開したり閉じたりすることができます。 各メッセージには、情報のタイプを表すイメージ アイコン、タイトル文字列、日付および簡単な説明が表示されます。そのメッセージが未読の場合、テキストはすべて太字で表記されます。 メッセージを既読にするには、その対象のメッセージをクリックして選択します。
右パネル	メッセージの詳細を示します。ここではそのメッセージについて詳しい説明を記載し、場合によっては問題を解決する提案事項も提供されます。

オペレーティング システムのイベント ログ

Zen は、オペレーティング システムのイベント ログに以下のエントリを書き込みます。

- Zen ライセンス管理メッセージ
- Zen イベント ログ（zen.log）に書き込まれる Zen エラーメッセージ
- （上記のメッセージも含む）Linux、macOS、および Raspbian 上の Zen コンポーネントの全メッセージ

詳細については、以下のトピックを参照してください。

- 「Windows プラットフォームのイベント ログ」
- 「Linux、macOS、および Raspbian ディストリビューションのイベント ログ」

Windows プラットフォームのイベント ログ

Windows オペレーティング システムでは、「アプリケーション」、「セキュリティ」、または「システム」として分類されるイベントの記録方法を提供しています。Zen では、一般的なエラー メッセージと選択したライセンス メッセージが zen.log に書き込まれると、Windows のアプリケーション イベント ログにもこれらのメッセージを記録します。

ライセンス メッセージの場合、「エラー」または「警告」として分類されるイベントが記録されます。これには、キーの状態の変更に起因するメッセージのほか、警告、エラー メッセージが含まれます（表 52 を参照）。さらに、表 51 で挙げているような特定の情報メッセージも記録されます。

イベント ログの表示

Windows オペレーティング システムでは、イベント ログを表示および操作するための「イベント ビューアー」という GUI を提供しています。この GUI には、Windows の PC 設定またはコントロール パネルから、あるいはコマンド インターフェイスで eventvwr.msc コマンドを実行してアクセスすることができます。

Zen は 1 つのイベントに対し以下の内容を表示します。

- 日付と時刻 - イベントの日付と時刻
- ソース - Actian Zen
- タスク カテゴリ - Zen
- 種類 / レベル - イベントのレベル：情報、警告またはエラー
- イベント ID - 1000
- ユーザー - N/A
- コンピューター - コンピューターの名前

さらに、［キーワード］列に " クラシック "、［ログ］列に " アプリケーション " を表示します。イベント ビューアーではさらに列が表示できるようになっていますが、Zen はそれらに対するデータを提供しません。

Linux、macOS、および Raspbian ディストリビューションのイベント ログ

Linux、macOS、および Raspbian ディストリビューションの場合は、Zen の全コンポーネントが標準ログ システムの syslog にメッセージを書き込みます。syslog はデフォルトで、Linux と Raspbian では /var/log/messages に、macOS では /var/log/system.log に書き込みます。SQL 接続マネージャーに限り、任意でメッセージを event.log ファイルに記録することもできます。

event.log ファイルと bti.ini

bti.ini は、Lnux、macOS、および Raspbian ディストリビューションで使用される Zen 設定ファイルです。デフォルトでは、このファイルは /usr/local/actianzen/etc にあります。

このファイルを使用すると、SQL　接続マネージャー用の構成を行うことができます（INI　ファイルの［SQLManager］セクション）。設定の 1 つである LogEvent は、イベント ログ ファイル（event.log）に記録するイベント メッセージのタイプを決定します。デフォルトでは、event.log は /usr/local/actianzen/bin にあります。

bti.ini の SQLManager セクション用の各種パラメーター	説明
MgrPort	SQL 接続マネージャーで使用されるポート番号を設定します。デフォルト値は 1583 です。
MgrUseTransport	SQL 接続マネージャーで使用されるプロトコルの種類を設定します。これは TCP に設定する必要があります
LogEvent=*msg_type*	*msg_type* には以下の値のいずれかを指定します（デフォルトは 1 です）。この値はイベント ログ（event.log）に記録するメッセージのタイプを示します。 • 0 － 記録しない • 1 － エラーのみ • 2 － エラーと警告 • 3 － エラー、警告および情報メッセージ • 4 － エラー、警告、情報メッセージおよび connect.log
InstallDirectory=/usr/local/actianzen	接続ログ /usr/local/actianzen/connect.log をアクティブ化します。

Zen イベント ログ（zen.log）

Windows プラットフォームの場合、Zen の全コンポーネントが Zen のイベント ログへステータス、エラー、警告、および情報メッセージを書き込みます。Linux、macOS、および Raspbian ディストリビューションの場合、Zen は専用のイベント ログを使用しません。その代り、Zen の全コンポーネントが標準ログ システムの syslog にメッセージを書き込みます。「Linux、macOS、および Raspbian ディストリビューションのイベント ログ」を参照してください。

Zen のイベント ログには zen.log という名前が付けられています。デフォルトで、このログは<**アプリケーション データ ディレクトリ**> ¥actianzen¥logs ディレクトリにあります。Windows プラットフォームでは Zen の全コンポーネントがこのログ ファイルに書き込みます。Zen データベース エンジンを使用した複数のアプリケーションが同じマシン上で動作している場合、これらのアプリケーションは zen.log を共有します。

zen.log のフィールド

zen.log の内容は、次の表 53 で説明されている形式のテキスト メッセージで構成されます。

表 53　zen.log のフィールド

フィールド	内容
Date	*yyyy/mm/dd/* 形式で表される自動日付スタンプ。
Time	*hh:mm:ss* 形式で表される自動時刻スタンプ。これらの表示形式は地域の設定に従います。
Component	エラーを返すコンポーネントのファイル名（拡張子なしのプレフィックスのみ）。
Process	コンポーネントのインスタンス ID。これは、コンポーネントのプロセス ID です。
Process Name	コンポーネントのパスおよび名前（最後の 15 文字に短縮）。
Computer Name	プロセスをホストするマシンに割り当てられた名前（最初の 15 文字に短縮）。
Type	1 文字 : I は情報、W は警告、E はエラーです。
Message	呼び出し元のコンポーネントに関連するリソースから取得される文字列、または呼び出し元のコンポーネントから直接渡されるテキスト文字列から成るメッセージ テキスト。 メッセージ テキストには数値が含まれることもあります。この数値の形式は 10 進数または 16 進数です。16 進数と 10 進数を区別するため、16 進数値の前には "0x" 文字が付けられます。 メッセージ テキストの中には、OEM アプリケーション固有の情報（ベンダーの Web サイトへのリンクやトラブルシューティング情報など）が含まれている場合もあります。

エントリの後に、標準の 16 進形式のバイナリ データが続くことがあります。バイナリ データの長さには、制限がありません。

zen.log のエントリ例

以下は、zen.log に含まれるデータの種類の例です。

Date	Time	Component	Process	Process Name
5/10/2019	09:53:06	LicenseMgr	9048	zenengnsvc.exe

Computer Name	Type Category	Message
USRegion2Svr	W	ライセンスは検証に失敗しました。修復期限は 14 日間です。

メッセージの電子メール通知を受け取る

Zen にはイベント メッセージの電子メール通知が含まれていません。そのような機能を提供するベンダーの製品が簡単に入手できるからです。このトピックでは、それらの製品のいくつかを挙げるとともに、Zen ライセンスおよび製品キーに関する操作を監視するためのイベント ログの使用についても説明します。

監視されるイベントの電子メール通知を提供する製品

次の表は、オペレーティング システムのイベント ログに含まれるイベントの電子メール通知を提供できる製品の一部を挙げています。製品はアルファベット順に並んでいます。弊社では、特定の製品について推奨することはしません。一覧に記載されている製品は、ほかのベンダーから提供されるものであるため、製品とそれに関する解説がここで述べたものと異なることがあります。

通常、そのような製品は、エージェントをインストールする、WMI（Windows Management Instrumentation）のようなリモート アクセス メカニズム、あるいはセキュア シェル（SSH）を有効にする必要があります。

表 54　監視されるイベントの電子メール通知を提供する製品

製品	コスト	プラットフォーム	説明
Hyperic www.hyperic.com	Hyperic 社までお問い合わせください。	Windows	監視されるマシン上にエージェントをインストールし、ファイアウォールでそのエージェント用にポートを開く必要があります。
		Linux、macOS、および Raspbian	監視されるマシン上で SSH を有効にし、ファイアウォールでその SSH 用にポートを開く必要があります。
Nagios www.nagios.org/	無料	Windows および Linux	監視されるマシン上にエージェントをインストールし、ファイアウォールでそのエージェント用にポートを開く必要があります。
Spiceworks www.spiceworks.com	無料	Windows	監視されるマシン上でリモート WMI を有効にし、ファイアウォールでそのリモート WMI 用にポートを開く必要があります。
		Linux	監視されるマシン上で SSH を有効にし、ファイアウォールでその SSH 用にポートを開く必要があります。
System Center Configuration Manager（SCCM） www.microsoft.com	Microsoft 社までお問い合わせください。	Windows	監査されるマシン上で、エージェントをインストールするか、またはリモート WMI を有効にする必要があります。WMI が有効になると、ファイアウォールは自動的に調整されます。エージェントを使用する場合は、ファイアウォールでそのエージェント用のポートも開く必要があります。この製品は Windows オペレーティング システムでのみ使用可能です。
ZenOSS www.zenoss.com	ZenOSS 社までお問い合わせください。	Windows	監視されるマシン上にエージェントをインストールし、ファイアウォールでそのエージェント用にポートを開く必要があります。
		Linux、macOS、および Raspbian	監視されるマシン上で SSH を有効にし、ファイアウォールでその SSH 用にポートを開く必要があります。

ライセンスのイベント メッセージを監視するイベント ログの内容

表 54 で挙げた製品はすべて、オペレーティング システムのイベント ログの内容を識別することができます。識別方法は、Windows プラットフォームと、Linux、macOS、および Raspbian ディストリビューションで異なりま

す。ここでは、各製品の具体的な監視方法について説明しません。詳細については、ベンダーのドキュメントを参照してください。

Windows プラットフォーム

次に表では、オペレーティング システムのイベント ログで、Zen ライセンスのプロパティを示しています。監視する製品の設定で、それらのプロパティの値に基づいて電子メール通知が送信されるようにします。

オペレーティング システムのイベント ログのプロパティ	監視する値
ソース	Actian Zen
タスク カテゴリ	Product Keys
種類	警告、エラー

例

Zen ライセンスに関するイベント メッセージのうち、警告またはエラーのメッセージについて電子メール通知を受け取りたいとします。製品を次のような条件で監視するよう設定します。

- ソース = "Actian Zen"
- タスク カテゴリ = "Product Keys"
- 種類 = " 警告 " | 種類 = " エラー "

Linux、macOS、および Raspbian ディストリビューション

Linux、macOS、および Raspbian の syslog には、Windows オペレーティング システムのイベント ログと同じプロパティは含まれていません。syslog 内のイベントと文字列を特定するよう通知製品を設定する必要があります。イベントと文字列の比較に基づいて、製品は電子メール通知など、希望の動作を開始することができます。

Zen データベース エンジンによって syslog に書き込まれるすべてのメッセージ（ライセンス関連かどうかに関わらずすべてのメッセージ）には `mkded` という文字列が含まれています。次の一覧には、syslog で Zen ライセンスのみに該当する文字列をいくつか示しています。syslog を監視する方法の 1 つは、`mkded` と以下の文字列の 1 つを含む内容に基づいて文字列比較を設定することです。

- `disabled`
- `failed validation`
- `for key`
- `validation of key`
- `capacity for session count`
- `capacity for data in use`
- `user count`

Btrieve オペレーションのテスト

14

Function Executor で Btrieve オペレーション実行する方法

以下のトピックでは Function Executor ツールの使用に関して説明します。

- 「Function Executor の概念」
- 「Function Executor のグラフィカル ユーザー インターフェイス」
- 「Function Executor での作業」

Function Executor の概念

ここでは、以下の項目について説明します。

概要

Function Executor は Windows 上で動作します。この対話型ツールを使用すると、Btrieve オペレーションがどのように動作するかを確認することができます。Btrieve オペレーションは、MicroKernel エンジン オペレーションと同じです。

また、Btrieve オペレーションを一度に 1 つずつ実行できるため、アプリケーション開発者は Function Executor を使って、Btrieve アプリケーションのオペレーションをシミュレートすることができます。このシミュレーションは、アプリケーションの残りの部分のデータベース呼び出しから分離しているので、プログラムのテストおよびデバッグに役立ちます。

Function Executor は、本来アプリケーション開発者向けのツールです。この章では、Btrieve オペレーションの基礎知識があることを前提に説明を進めます。Btrieve オペレーションの詳細については、開発者リファレンスの『*Btrieve API Guide*』を参照してください。

Function Executor でできること

- Btrieve オペレーションを実行しながらメモリ構造の内容を監視することができます。
- 一連の Btrieve オペレーションをキャプチャして履歴ファイルとして保存し、後で再生することができます。
- Btrieve クライアント、ローカル エンジンおよびリモート エンジンのバージョンを表示します。
- データ ファイルの Btrieve 特性を表示し、これらをテンプレート（ディスクリプション ファイル）として保存したり、これらの特性に基づいて新しいファイルを作成したりすることができます。詳細については、「ファイル統計情報」を参照してください。

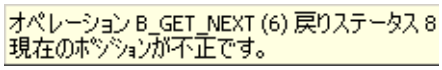
Function Executor の機能

Function Executor の機能は以下のとおりです。

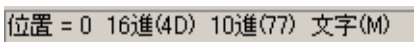
エディター ステータス バー

ステータス バーには以下の要素があります。

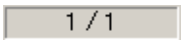
開いているファイルのウィンドウの下部にあるエディター ウィンドウのステータス バーに最新 / 現在のステータス コードが表示され、最新のステータスがゼロ以外の場合には赤色で表示されます。



ステータス コードにマウス カーソルを移動すると、そのステータスについての説明と、原因となったオペレーションが表示されます。これらのステータス コードのヘルプ全体を表示するには、赤色で表示されている部分をクリックします。詳細については、「ステータス コードのヘルプを見るには」を参照してください。



カーソルが Function Executor メイン ウィンドウの入力領域にあるときには、ステータス バーにはバッファー内でのオフセットが表示されます。つまり、現在の位置のバイトを 16 進数、10 進数、ASCII 値で表した値が表示されます。

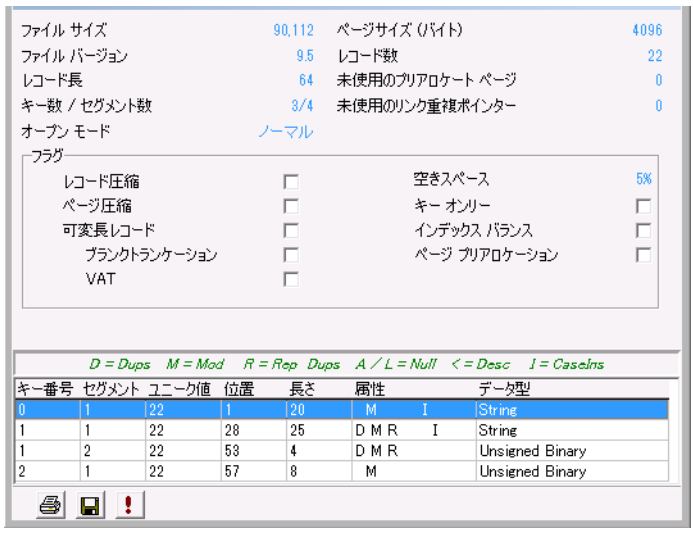


キューに複数の項目がある場合には、オペレーションがいくつ実行されたのかもステータス バーに表示されます。通常、1 / 1 と表示されますが、複数のオペレーションを実行している場合には実行されるオペレーションの数が表示されます。

トランザクションに入ると、その状態もステータス バーに表示されます。

統計情報

[**ファイル統計情報**] アイコンをクリックすると、現在開いているファイルについての統計情報を一覧表示したダイアログ ボックスが表示されます。これらの統計情報をテキスト ファイルに出力したり、`butil -create` コマンドで使用できるディスクリプション ファイルに保存することができます。また、同じ特性を持つ空のファイルを新規作成することもできます。

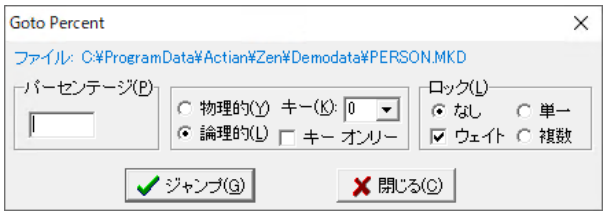


Get および GetExt

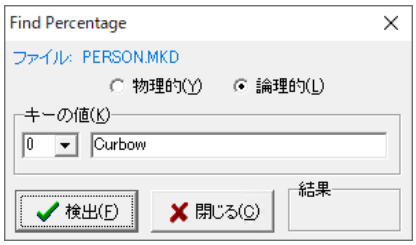
[Get] メニューから、テーブル内にある先頭レコード、次レコード、前レコード、末尾レコードを取得することができます。[GetExt] メニューには、[Goto Percent（**パーセンテージによるレコードの取得**）]、[Get Position（**物理位置の取得**）]、[Find Percent（**レコード位置のパーセンテージを取得**）] の各コマンドがあります。

［Get］コマンドと［GetExt］コマンドは、メニュー バーとツールバーから使用できます。ツールバーには、［Step］（物理的）と［Get］（論理的）があり、ファイルの自然の順序で（物理的）、または特定の順序で（論理的）移動できます。

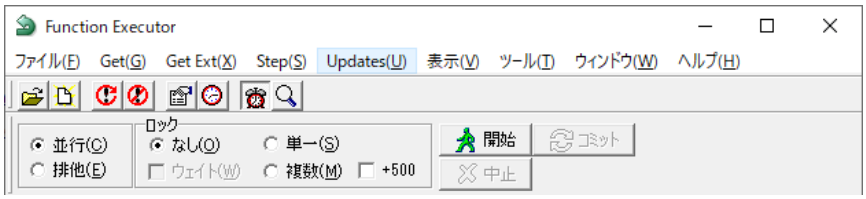
［Goto Percent（パーセンテージによるレコードの取得）］では、ファイルの物理レイアウト内での特定の地点に移動するか、あるいはそのファイル内に定義されているキーを定めて、任意のキー パスをたどっていくかどうかの選択ができます。［ロック］グループ ボックスのオプション ボタンを使用して、ロック バイアスを設定することもできます。



［Find Percent（レコード位置のパーセンテージを取得）］は、［Goto Percent（パーセンテージによるレコードの取得）］の逆です。ファイル内を論理的に移動しているか、物理的に移動しているかによって、データ内での位置を示します。



トランザクション ツールバー



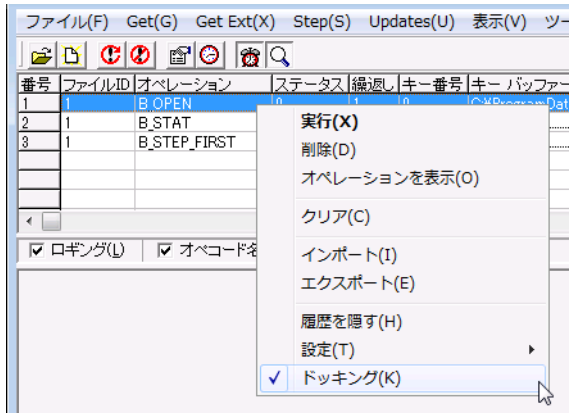
トランザクション ツールバーを使用すると、トランザクションの開始、終了、中止を行うことができます。Transaction API のすべてをこのツールバーで設定することができ、オペレーションはすぐに実行されます。トランザクションのステータスは、そのクライアント ID 用に開いているすべてのファイルに影響するため、メイン ウィンドウのステータス バーにも表示されます。

ログイン ダイアログ

ログイン ダイアログ ボックスを使用すると、GUI インターフェイスを介して Btrieve ログイン操作を実行することができます。詳細については、データベース セキュリティとその設定に関するトピックを参照してください。

履歴ログ

Function Executor ツールを使用してオペレーションを実行すると、これらは履歴ログに記録されます。このログを使用して、ログに含まれるオペレーションを実行したり、履歴をファイルとして保存し後で再読み込みして同じ手順を行うことができます。



履歴ログの詳細については以下のトピックを参照してください。

- 「履歴」
- 「履歴の作業」

任意のデータ型での表示

ファイルが開いているとき、バッファー内の任意の場所を右クリックし、[**表示形式**] をクリックします。ダイアログ ボックスが表示され、選択したバッファー位置のバイト数を任意のデータ型で表示できます。

表 55 に Function Executor で使用できるコントロールを示します。

表 55　Function Executor のコントロール

コントロール	説明
履歴	コマンドの繰り返しができます。
作成	新規ファイルを作成できます。
ファイル統計情報	BSTAT 関数の情報を示します。この統計情報は印刷できます。
MDI	Multiple Document Interface によって、複数のファイルを開くことができます。
リセット	クライアント ID をリセット

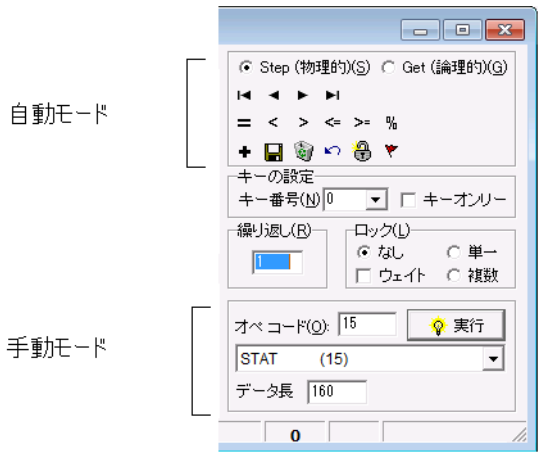
表 55　Function Executor のコントロール

コントロール	説明
ストップ	Btrieve の停止
バージョン	Btrieve バージョン情報

Function Executor の自動モード

開いているファイルごとに（「アプリケーション ウィンドウ」を参照）Btrieve オペレーションを実行する際に Function Executor を使用するかどうかを選択することができます。Function Executor を使用する方法としない方法で設定を変更する必要はありません。自動モードと手動モードのどちらを使用するかは、どの GUI コントロールを使用するかによって決まります。

図 9 自動モードと手動モードのコントロール



メモ メニュー（「アプリケーション ウィンドウ」を参照）での選択も自動モードの一部として行われます。

自動モード領域のボタンをクリックすると、ツールは以下のような支援を行います。

- データ バッファーとデータ長は、自動的にステータス コード 22 を防ぐように設定されます。
- 情報は操作に応じて適切に要求されます。

詳細情報を入手するには

Function Executor はプログラム開発者にとって有益なツールですが、Btrieve の実際的な基礎知識があることを前提にしています。このツールのすべての機能を理解するには、以下のトピックを参照してください。

- 『*Zen Programmer's Guide*』の「MicroKernel エンジンの基礎」
- 『*Btrieve API Guide*』
- 本ドキュメントのセキュリティに関するさまざまなトピック

Function Executor のグラフィカル ユーザー インターフェイス

このトピックでは、Function Executor のグラフィカル ユーザー インターフェイス（GUI）のオブジェクトについて説明します。

- 「アプリケーション ウィンドウ」
- 「メイン ウィンドウ」
- 「ログインおよびログアウト」
- 「[ファイルのオープン] ダイアログ」
- 「Btrieve ファイルの新規作成用ダイアログ」
- 「トランザクション ツールバー」
- 「ファイル統計情報」
- 「履歴」

アプリケーション ウィンドウ

下の表は次の図のウィンドウ コンポーネントの説明です。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

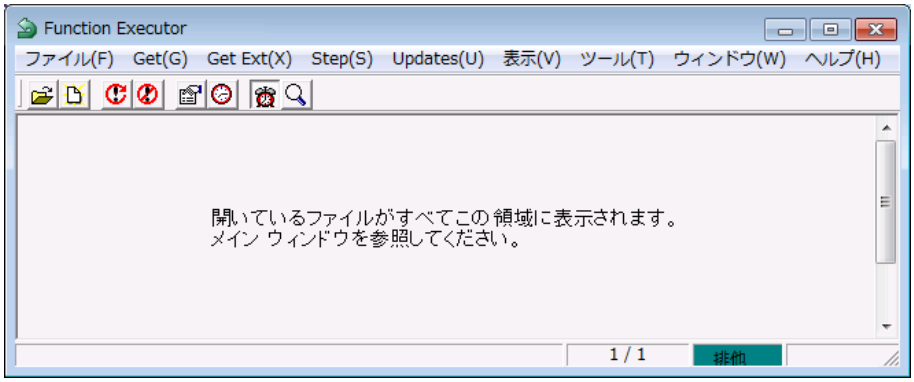


表 56　Function Executor アプリケーション ウィンドウ

GUI のオブジェクト	説明	関連情報
［ファイル］メニュー	以下のコマンドおよび Btrieve オペレーションが実行できます。 • Login および Logout • Open および Close • New • Print Setup • Set Owner Name および Clear Owner Name • Continuous オペレーションの開始または終了 • Reset、Stop および Exit	「オペレーション作業の実行」 「ファイルを開く作業」
［Get］メニュー	以下の Btrieve オペレーションが実行できます。 • Get First および Get Next • Get Previous および Get Last • Get Greater Than および Get Greater or Equal • Get Less および Get Less or Equal	「オペレーション作業の実行」
［GetExt］メニュー	以下の Btrieve オペレーションが実行できます。 • Goto Percent • Get Position • Find Percent	「オペレーション作業の実行」
［Step］メニュー	以下の Btrieve オペレーションが実行できます。 • Step First および Step Next • Step Previous および Step Last	「オペレーション作業の実行」
［Update］メニュー	以下の Btrieve オペレーションが実行できます。 • Insert、Update および Delete このメニューを使用してロックを解除することもできます。	「オペレーション作業の実行」

表 56　Function Executor アプリケーション ウィンドウ

GUI のオブジェクト	説明	関連情報
［表示］メニュー	GUI 要素を表示することができます。 • ツール バー（メインおよびトランザクション） • 履歴ウィンドウ • ファイル統計情報ウィンドウ • Btrieve Version オペレーションを使用したエンジンのバージョン	「履歴の作業」
［ツール］メニュー	以下の Btrieve オペレーションが実行できます。 • Get Directory および Set Directory	
［ウィンドウ］メニュー	ウィンドウ操作が実行できます。 • ［ウィンドウを重ねて表示］および［ウィンドウを並べて表示］ • 開いているウィンドウの一覧から選択	
［ヘルプ］メニュー	このツールのヘルプ リソースを一覧から選択することができます。	「ステータス コードのヘルプを見るには」
［開く］ボタン 	ダイアログ ボックスを表示し、開く Btrieve ファイルを選択します。	「ファイルを開く作業」
［新規作成］ボタン 	ダイアログ ボックスを表示し、新規 Btrieve ファイルを作成することができます。	「Btrieve ファイルを作成する作業」
［リセット］ボタン 	現在のクライアント接続をリセットし（Btrieve オペレーション 28）、そのクライアント ID で開いているすべてのファイルを閉じます。	
［停止］ボタン 	トランザクション オペレーション（Btrieve オペレーション 25）を終了させ、開いているすべてのファイルを閉じます。	
統計情報 	現在開いているファイルの統計情報を一覧表示するダイアログ ボックスを表示します。これらの統計情報をテキスト ファイルに出力したり、BUTIL -CREATE で使用できるディスクリプション ファイルに保存することができます。	
［バージョン］ボタン 	実行中の Zen のバージョン情報を（Btrieve オペレーション 26 を使用して）表示します。開かれているファイルがない場合には、リクエスター DLL とローカル MicroKernel エンジンについての情報が表示されます。リモート サーバーでファイルを開いている場合、サーバーとリクエスター DLL についての情報が表示されます。	
ヘルプの表示 	ゼロ以外のステータス コードを受け取ったときにポップアップ ダイアログ ボックスを表示するかしないかを切り替えます。	
履歴の表示 	履歴ウィンドウを表示するかしないかを切り替えます。	「履歴ウィンドウを表示するには」 「履歴」 「履歴ログ」

表 56　Function Executor アプリケーション ウィンドウ

GUI のオブジェクト	説明	関連情報
オペレーション カウント	現在のオペレーション数を示します。 繰り返しの値に 2 以上を設定したときに使用されます。	「オペレーション作業の実行」
トランザクションの状態 	トランザクションの現在の状態を示します。以下のいずれかになります。 • 空白（トランザクションが 1 つも有効でない場合） • 並行（Concurrent） • 排他（Exclusive）	「オペレーション作業の実行」 「トランザクション ツールバー」

メイン ウィンドウ

すべての開いているファイルについて、メイン ウィンドウが表示されます。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

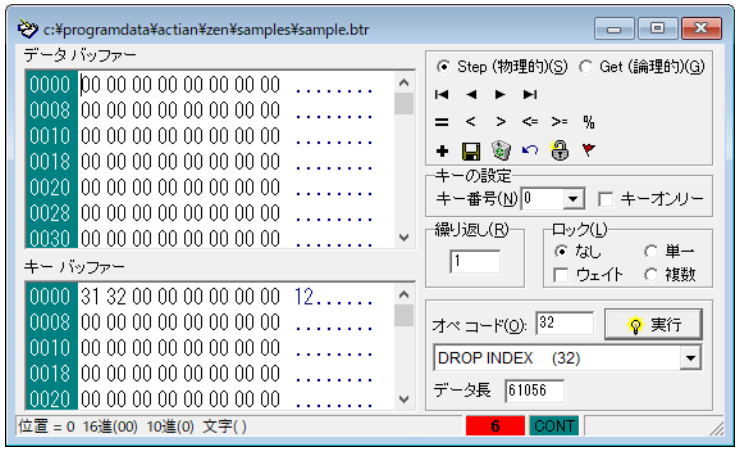


表 57　Function Executor メイン ウィンドウ（オープン ファイル用）

GUI のオブジェクト	説明	関連情報
タイトル バー	開いているデータ ファイルの絶対パスを一覧表示します。	「Function Executor でデータ ファイルを開くには」
データ バッファー	データの値を指定します。読み取りや書き込みオペレーションでは、Data Buffer にレコードが含まれています。ほかのオペレーションには、データベース エンジンでオペレーションの処理に必要なファイル スペック、フィルター条件、その他の情報が Data Buffer に含まれています。このコントロールは Data Buffer パラメーターに対応するものです。	
キー バッファー	Btrieve オペレーションを実行するデータ ファイルのパスを指定します。	
Step と Get	Step および Get オペレーションを切り替えます。	
Get/Step First	Get Next または Step Next オペレーションを実行します。	「オペレーション作業の実行」
Get/Step Prev	Get Previous または Step Previous オペレーションを実行します。	「オペレーション作業の実行」
Get/Step Next	Get Next または Step Next オペレーションを実行します。	「オペレーション作業の実行」
Get/Step Last	Get Last または Step Last オペレーションを実行します。	「オペレーション作業の実行」
Get Equal	Get Equal オペレーションを実行します。	「オペレーション作業の実行」
Get Less Than	Get Less Than オペレーションを実行します。	「オペレーション作業の実行」
Get Greater Than	Get Greater Than オペレーションを実行します。	「オペレーション作業の実行」
Get Less Than or Equal	Get Less Than または Get Equal Than オペレーションを実行します。	「オペレーション作業の実行」
Get Greater Than or Equal	Get Greater Than または Get Equal Than オペレーションを実行します。	「オペレーション作業の実行」

表 57　Function Executor メイン ウィンドウ（オープン ファイル用）

GUI のオブジェクト	説明	関連情報
Get/Find Percent	Get Percent または Find Percent オペレーションを実行します。	「オペレーション作業の実行」
Insert	Insert オペレーションを実行します。	「オペレーション作業の実行」
Update	Update オペレーションを実行します。	「オペレーション作業の実行」
Delete	Delete オペレーションを実行します。	「オペレーション作業の実行」
Cancel	最後に行った変更をキャンセルします。	
Unlock	すべてのロックを解除します。	
Set Bookmark または Goto Bookmark	前もって定義しておいたブックマークに設定または位置付けます。	
キー番号	ほとんどの Get オペレーションで、現在のオペレーションが従うことになるキー番号またはインデックス パスを指定します。ほかのオペレーションには、ファイル オープン モード、暗号化、論理ディスク ドライブなどの情報を指定するものもあります。このコントロールは Key Number パラメーターに対応するものです。	
キー オンリー	データではなくキーのみを取得することを指定します。	
繰り返し	オペレーションを指定した回数だけ繰り返します。	
ロック	現在のオペレーションで行いたいロック動作を指定します。	
オペレーション コード	現在のオペレーション コードとそのバイアス（ある場合）を示します。デフォルト設定は、0 です。Btrieve オペレーション コードに精通している場合は、目的のコードを入力することができます。慣れていなければ、［**リスト**］ボックスを使用してオペレーションを指定します。このコントロールは、**オペレーション コード** パラメーターに対応するものです。	「オペレーション作業の実行」
［実行］ボタン	現在指定されているオペレーションを実行します。	「オペレーション作業の実行」
オペレーション リスト	Btrieve オペレーションとそのコードをリストします。デフォルトは Open（0）です。実行するオペレーションの頭文字を入力すれば、リスト内ですぐに移動ができます。	「オペレーション作業の実行」
データ長	データ バッファーの長さ（バイト単位）を指定します。デフォルト値は 1024 です。データ バッファーを必要とする各オペレーションでは、バッファー長を必ず指定してください。多くのオペレーションでは、データベース エンジンにより Data Length に値が返されます。一般には、オペレーションの実行前に、Data Length を必ず指定してください。このコントロールは Data Buffer Length パラメーターに対応するものです。	「オペレーション作業の実行」
ステータス コード インジケーター	データベース エンジンで返された数値のステータス コードと、Btrieve オペレーションの結果を示す短いメッセージが表示されます。 これらのステータス コードやメッセージについての詳細は、『*Status Codes and Messages*』マニュアルを参照してください。	「ステータス コードのヘルプを見るには」
Continuous オペレーション インジケーター	ファイルが Continuous オペレーション モード（オペレーション 42）にある場合、ファイル ウィンドウの最下行に以下を表示します。	「Continuous オペレーションの使用」

ログインおよびログアウト

ログイン ダイアログ ボックスとログアウト ダイアログ ボックスでは、データベース エンジンに対する Btrieve のログインおよびログアウト オペレーションを実行することができます。画像の領域をクリックするとその詳細が表示されます。

図 10 ログイン ダイアログ

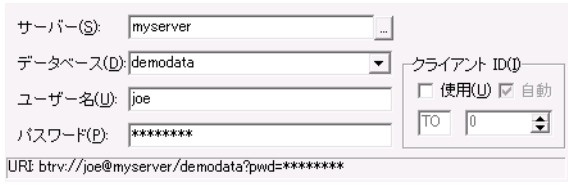


図 11 ログアウト ダイアログ

表 58 Function Executor ログインおよびログアウト ダイアログ

GUI のオブジェクト	説明
サーバー	ログインまたはログアウトしたいデータベースが存在するサーバーを指定します。Linux および macOS オペレーティング システム上のデータベースにアクセスするにはサーバー名が必要なので注意してください。『*Zen Programmer's Guide*』の「データベース URI」も参照してください。
データベース	認証を求めるサーバー上のデータベースを指定します。
ユーザー名	データベースに対して認証を求めるユーザーの名前です。
パスワード	ユーザー名に対するパスワードです。
クライアント ID	このログインまたはログアウトを特定のクライアント ID にのみ適用する場合は、［使用］をクリックして範囲を指定します。それ以外の場合は、そのままにしておきます。 『*Btrieve API Guide*』の「クライアント ID」を参照してください。
URI 文字列	情報をフォームに入力すると、選択した結果の URI がこの領域に表示されます。

［ファイルのオープン］ダイアログ

このダイアログを使用すると、ファイルを開くことができます。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

表 59　Function Executor ファイルのオープン ダイアログ

GUI のオブジェクト	説明	関連情報
ファイル名	開くファイルの場所と名前を設定します。	「ファイルを開く作業」
オーナー ネーム	Btrieve ファイルのパスワードとして機能します。このパスワードは Btrieve ファイルへのアクセスに必要です。 短いオーナー ネームは半角 8 文字までの範囲で指定できます。長いオーナー ネームの長さは、ファイル形式によって異なります。詳細については、「オーナー ネーム」を参照してください。	「ファイルを開く作業」
モード	ファイルを開くときの状態を設定します。この状態に基づき、データベース エンジンは開くファイルに適用する条件を識別します。たとえば、ファイルの読み取りは可能だが更新はできない（リードオンリー）という条件が可能です。	「ファイルを開く作業」 モードの説明については、『*Btrieve API Guide*』を参照してください。
クライアント ID	このログインを特定のクライアント ID にのみ適用する場合は、［使用］をクリックして範囲を設定します。それ以外の場合は、そのままにしておきます。	「ファイルを開く作業」
共有	データベース エンジンは、［共有］オプションを無視します。［共有］オプションは、このエンジンのレガシー バージョンである Btrieve 6.15 のみに適用されます。	「ファイルを開く作業」

Btrieve ファイルの作成

このダイアログ ボックスを使用すると、既に開いているファイルを基に、Btrieve ファイルを作成することができます。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

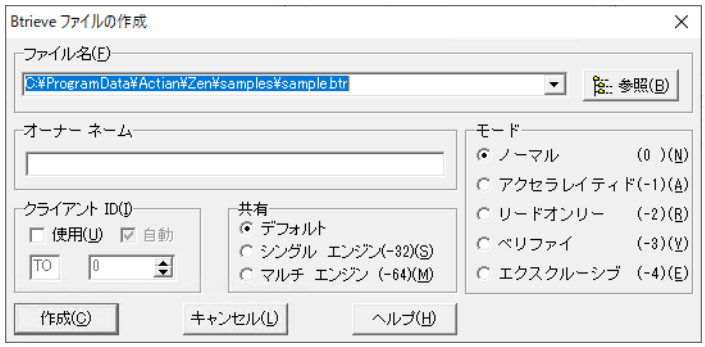


表 60　Function Executor ファイルの作成ダイアログ

GUI のオブジェクト	説明	関連情報
ファイル名	作成するファイルの場所と名前を設定します。	「Btrieve ファイルを作成する作業」
オーナー ネーム	Btrieve ファイルのパスワードとして機能します。このパスワードは Btrieve ファイルへのアクセスに必要です。 短いオーナー ネームは半角 8 文字までの範囲で指定できます。長いオーナー ネームの長さは、ファイル形式によって異なります。詳細については、「オーナー ネーム」を参照してください。	「Btrieve ファイルを作成する作業」
モード	ファイルを開くときの状態を設定します。この状態に基づき、データベース エンジンは開くファイルに適用する条件を識別します。たとえば、ファイルの読み取りは可能だが更新はできない（リードオンリー）という条件が可能です。	「Btrieve ファイルを作成する作業」 モードの説明については、『*Btrieve API Guide*』の「オープン モード」を参照してください。
クライアント ID	このログインを特定のクライアント ID にのみ適用する場合は、[使用] をクリックして範囲を設定します。それ以外の場合は、そのままにしておきます。	「Btrieve ファイルを作成する作業」
共有	データベース エンジンは、[共有] オプションを無視します。[共有] オプションは、データベース エンジンのレガシー バージョンである Btrieve 6.15 のみに適用されます。	

Btrieve ファイルの新規作成用ダイアログ

このダイアログ ボックスを使用すると、作成されているファイルに、さらに別の属性を指定することができます。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

表 61　Function Executor ファイルの作成ダイアログ（高度）

GUI のオブジェクト	説明	関連情報
[キー]および[セグメント]コマンド	キーの追加と削除や、キー セグメントの追加、挿入、削除が行えます。	「Btrieve ファイルを作成する作業」
ファイル仕様	ファイルの仕様の表示、およびその編集が行えます。	「Btrieve ファイルを作成する作業」 「レコードおよびページ圧縮」
統計情報	ファイルに関する情報を提供します。読み取り専用です。	「Btrieve ファイルを作成する作業」
キーとセグメントの行列	ファイルのキーとセグメントの開始位置、長さ、属性およびデータ型を表示します。行をクリックすると、下方のキーとセグメントの各ブロックで情報が表示されます。必要であれば、この情報は変更することもできます。	「Btrieve ファイルを作成する作業」
キー	キーの仕様の表示、およびその編集が行えます。	「Btrieve ファイルを作成する作業」
セグメント	セグメントの仕様の表示、およびその編集が行えます。	「Btrieve ファイルを作成する作業」
各種コマンド ボタン	データ ファイルやディスクリプションの作成、あるいはダイアログ ボックスで行った編集をキャンセルすることができます。	「Btrieve ファイルを作成する作業」

トランザクション ツールバー

ファイルが開いているとき、[表示] > [ツールバー] > [トランザクション] を選択すると、トランザクション ツールバーを表示できます。このツールバーを使用して、トランザクションを手動で制御することができます。次の画像の領域をクリックすると、このツールバーで行えることが表示されます。

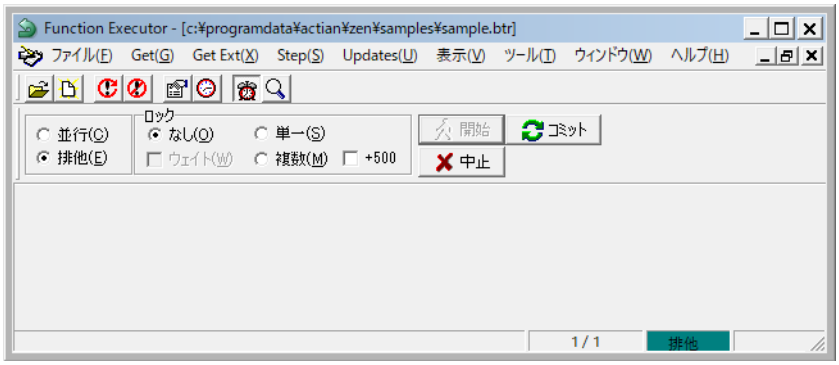


表 62　Function Executor トランザクション ダイアログ

GUI のオブジェクト	説明	関連情報
メイン アプリケーション ウィンドウ	プログラムの主な機能です。	「アプリケーション ウィンドウ」
トランザクションの種類	排他的トランザクションまたは並行トランザクションのどちらを使用するかを指定します。	
ロック情報	トランザクションで使用するロックの種類を指定します。	
[トランザクションの開始] ボタン	トランザクションを開始します。	
[トランザクションのコミット] ボタン	アクティブなトランザクションをコミットします。	
[トランザクションの中止] ボタン	アクティブなトランザクションを中止し、行われた変更をロールバックします。	
ファイル領域	この領域には開いているファイルが 1 つ以上含まれます。	「メイン ウィンドウ」
トランザクション インジケーター	現在トランザクション内にあるかどうかを示します。ここには " 並行 " または " 排他 " と表示されます。	

ファイル統計情報

ファイルが開いているとき、［表示］>［ファイル統計情報］を選択する、ツール バーの［ファイル統計情報］アイコンをクリックする、または Ctrl+S キーを押すと、ファイル統計情報ウィンドウを表示できます。このウィンドウでは、ファイルの統計情報を見ることができます。次の画像の領域をクリックすると、このウィンドウで行えることが表示されます。

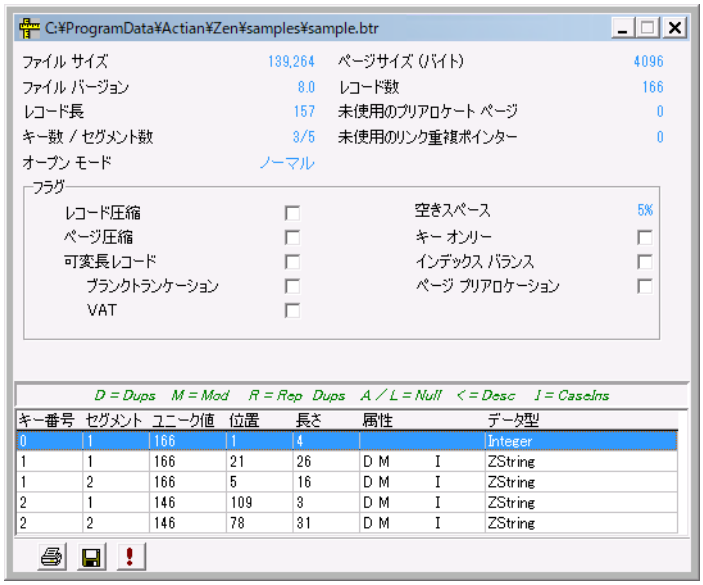


表 63　Function Executor ファイル統計情報ダイアログ

GUI のオブジェクト	説明
ファイル情報	ファイルの統計情報を表示します。
フラグ	このファイルに設定されているフラグを表示することができます。ダイアログ ボックス内の一覧で、設定されているすべてのフラグの隣にはチェックマークが付きます。
キー	このファイルに定義されているキーを表示することができます。
キーの凡例	1 文字の値でこのファイルのキー属性を示します。 • D = DUP（重複可能） • M = MOD（変更可能） • R = REPEAT_DUPS_KEY（繰り返し重複キー） • A= NUL（全セグメント ヌル キー） • L = MANUAL_KEY（一部セグメント ヌル キー） • < = DESC_KEY（キー値を表します） • I = NOCASE_KEY（大文字小文字無視キー） 『*Zen Programmer's Guide*』の「キー属性」も参照してください。
［印刷］ボタン	ファイル統計情報を印刷することができます。［ファイル］メニューを使用してプリンターを設定します。
［保存］ボタン	現在のファイル統計情報をディスクリプション ファイルに保存することができます。このディスクリプション ファイルを使用して、後で同じ属性を持つファイルを新規作成することができます。
［ファイルの新規作成］ボタン	これらのファイル統計情報に基づいて新規ファイルを作成することができます。

履歴

このダイアログ ボックスを使用すると、これまでに実行したすべてのオペレーションを見ることができます。スクリーンショット上のそれぞれの領域をクリックするとその詳細が表示されます。

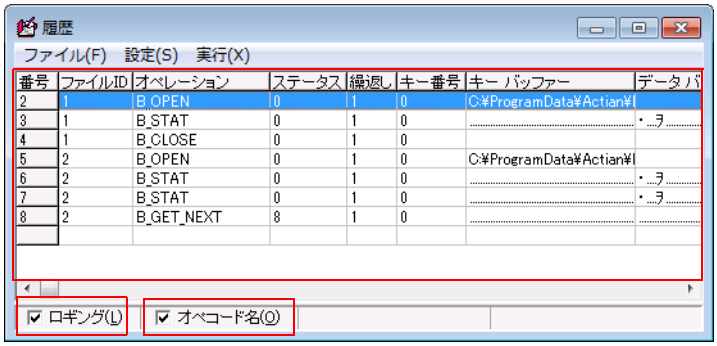


表 64　Function Executor 履歴ダイアログ

GUI のオブジェクト	説明	関連情報
［ファイル］メニュー	以下の操作が実行できます。 • 履歴ファイルのインポート • 履歴ファイルのエクスポート • 履歴ウィンドウを閉じる	
［設定］メニュー	履歴ウィンドウの表示属性を指定することができます。 • 終了時に保存 - 履歴ウィンドウで行ったカスタマイズをほかのセッションでも使用できるように保存するかどうかを指定します。 • デフォルト - 履歴ウィンドウをデフォルトの設定にリセットします。指定した設定はすべて取り除かれます。 • ドッキング - 履歴ウィンドウの状態で、分離したウィンドウまたは「アプリケーション ウィンドウ」と一体化した状態とを切り替えます。 • 手前に表示 - 履歴ウィンドウの状態を、フォーカスを失わない状態と失う状態の間で切り替えます。	• 「履歴ウィンドウのドッキング状態を切り替えるには」 • 「履歴ウィンドウの常に前面に表示状態を切り替えるには」 • 「履歴ウィンドウをデフォルトの設定にリセットするには」
［実行］コマンド	［履歴の再生設定］ウィンドウをロードします。	
オペレーションの一覧	最近実行したオペレーションを一覧表示します。各オペレーションと共に以下の情報がログに記録されます。 • ファイル ID • ［オペ コード名］チェック ボックスの状態により、オペレーション名またはオペレーション番号。 • そのオペレーションが実行されたときのステータス コード。 • そのオペレーションが実行された回数。 • そのオペレーションで設定されたキー番号。 • キー バッファーの内容。 • データ バッファーの内容。	• 「履歴の作業」 • 「履歴」

表 64　Function Executor 履歴ダイアログ

GUI のオブジェクト	説明	関連情報
ロギング	履歴一覧に将来のオペレーションを含めるか含めないかを切り替えます。	
オペ コード名	オペレーション列に表示するオペレーション　コード名（B_OPEN など）とオペレーション コード番号（B_OPEN に対する 0）を切り替えます。	

Function Executor での作業

Function Executor での作業は以下のカテゴリにグループ化されています。

- 「Function Executor での作業の開始」
- 「オペレーション作業の実行」
- 「ファイルを開く作業」
- 「Btrieve ファイルを作成する作業」
- 「履歴の作業」

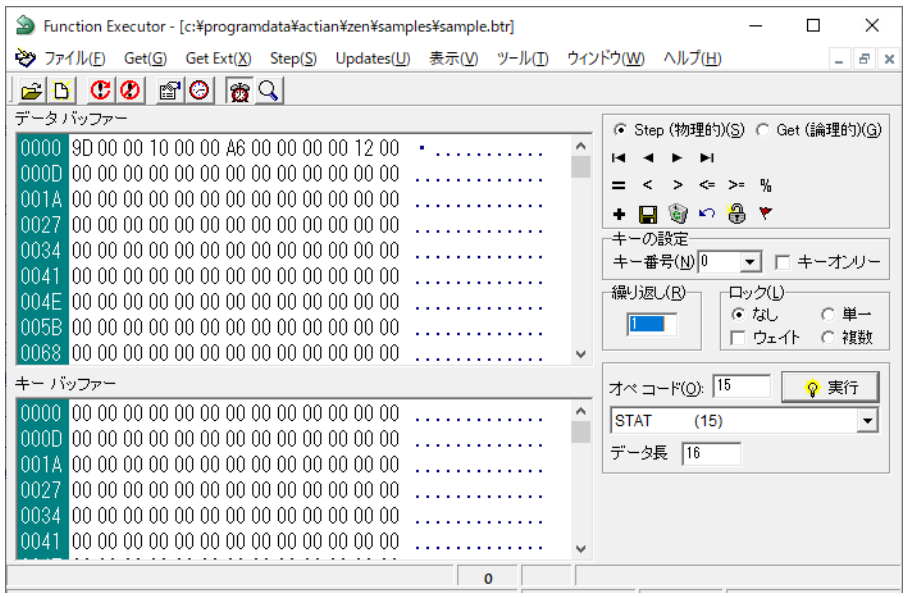
Function Executor での作業の開始

▶▶ Function Executor を起動するには

1 オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から、あるいは Zen Control Center の［**ツール**］メニューから Function Executor にアクセスします。

2 メイン ウィンドウが表示されます。

図 12 Function Executor メイン ウィンドウ



オペレーション作業の実行

Btrieve には数多くのオペレーションがあるため、この章ですべてを説明することはできません。次のセクションでは、一般的ないくつかのオペレーションと、Function Executor を使ったそれらの新しい実行方法についても説明します。

メモ どのメニューからでもオプションを選択すれば、目的のオペレーションがすぐに実行されます。以前のバージョンの動作のように、グリッドに自動入力したりコマンドの実行待ちになることはありません。また、フォームを閉じると、開いているファイルは閉じられます。Close オペレーションを手動で実行する必要はありません。

全般的なオペレーション - 関連する作業

- 「ステータス コードのヘルプを見るには」

ほかの作業については、以下のセクションを参照してください。

- 「ファイルを開く作業」
- 「Btrieve ファイルを作成する作業」
- 「履歴の作業」

▶▶ ステータス コードのヘルプを見るには

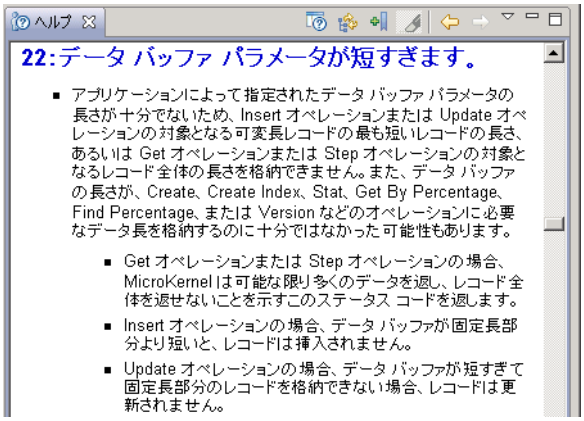
1 Function Executor を使用してステータス コードを受け取った場合は、開いているファイルのステータス バーに表示されます。

マウスを動かして赤色で表示されているステータス コード上に移動します。

図 13 ステータス コードを受け取る

2 ステータス コード インジケーターをクリックすると、図 14 のように、そのステータス コードの完全な説明が表示されます。

図 14 ステータス コードの説明

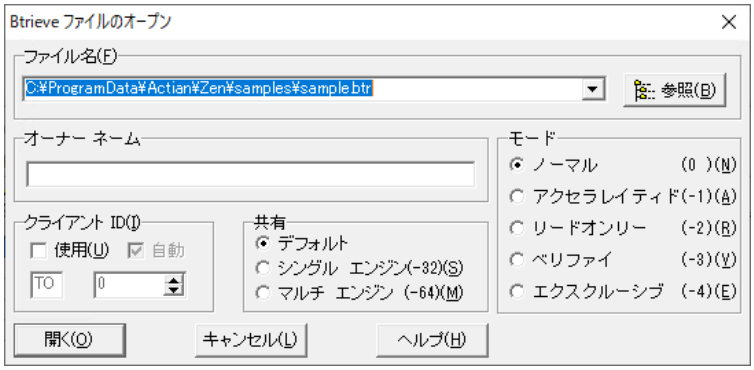


ファイルを開く作業

▶▶ Function Executor でデータ ファイルを開くには

1 ［ファイル］メニューの［開く］を選択します。次のダイアログ ボックスが表示されます。

図 15 ［Btrieve ファイルのオープン］ダイアログ ボックス



メモ クライアント ID：［使用］を無効にしている場合、Function Executor では、BTRV() 関数呼び出しが使用されます。［使用］を有効にしている場合、そのファイルで実行される各オペレーションについて BTRVID() 関数呼び出しが使用されます。［自動］を有効にしている場合、Function Executor では、クライアント ID が自動的に作成されます。［自動］を無効にしている場合には、手動で値を入力できます。

2 ［**参照**］をクリックします。

3 目的のファイル名をダブルクリックします。

Function Executor でファイルを開くその他の方法

- Windows エクスプローラーから［Btrieve ファイルのオープン］ダイアログ ボックス内にファイルをドラッグすることができます。ドラッグしたファイル名が［ファイル名］ボックスに入ります。
- メイン ウィンドウには複数のファイルをドラッグすることができます。
- 1 つのファイルを開いてエディター ウィンドウを表示しておけば、オペ コード（オペレーション コード）0 を使用して別のファイルを開くことができます。そのファイルは新しいウィンドウに表示されます。
- コマンド プロンプトから Function Executor を起動し、ファイル名の一覧を指定して開くことができます。たとえば、次のように指定します。

  ```
  WBExec32 person.mkd billing.mkd
  ```

  次のように、ワイルドカードを使用することもできます。

  ```
  WBExec32 *.mkd
  ```

  また、.btr、.dat、.mkd やその他の拡張子を持つファイルを Function Executor に関連付けることができます。その後、関連付けされたファイルを Windows エクスプローラー内でダブルクリックすると、Function Executor でそのファイルが開きます。

Btrieve ファイルを作成する作業

Function Executor で Btrieve ファイルを作成する方法には 2 つの方法があります。ファイルが既に開いている場合は、それを複製することができます。そうでなければ、1 から作成します。

注意 Btrieve エンジンは、拡張子を無視してファイル名を使用します。このため、同じディレクトリに、ファイル名が同一で拡張子のみが異なるようなファイルを置かないでください。たとえば、データ ファイルの 1 つに invoice.btr、もう 1 つに invoice.mkd という名前を付けてはいけません。

方法 1：現在のファイルをテンプレートとして使用する

1　［**ファイル**］メニューの［**新規作成**］を選択します。次のダイアログ ボックスが表示されます。

図 16　［ファイル定義の修正］ダイアログ ボックス

2　このダイアログ ボックスでキーの操作もできます。［**キー**］メニューから、［追加］、［削除］、［**セグメント**］メニューから［追加］、［挿入］、［削除］のいずれかをクリックします。新規ファイルを BUTIL-create での使用の目的でディスクリプションとして保存することもできます。［**ディスクリプションとして保存**］をクリックし、ファイルの保存先の名前と場所を示します。

3　ファイルを作成するには、［**作成**］をクリックします。すると、ファイルが開き、作成が完了したことを示すメッセージが表示されます。

方法 2：新しいファイルを始めから作成する

1　メイン ツール バーの［**Btrieve ファイルの作成**］アイコンをクリックするか、ファイルがまだ開いていない場合には、前項と同様に［**ファイル**］メニューの［**新規作成**］をクリックします。

2　画面上にファイルが既に開かれている場合には、選択ボックスが表示されます。［**新規ファイルの作成**］を選択します。

3　前と同じダイアログ ボックスが値が空白の状態で表示され、これに新しい値を入力できます。

4　まず、［**キー**］メニューを使用するか、Ctrl キーと A キーを同時に押して新規キーを追加します。

5　ダイアログ ボックスの下部にキーの属性を入力します。

6　必要に応じてキーやセグメントの追加や削除を続けます。これにはメニューを使用するか、一覧表示されたキーを右クリックします。

7　ここで、［**作成**］ボタンをクリックすると、B_Create（14）オペレーションが実行されます。すると、画面上にそのファイルも自動的に開きます。

履歴の作業

以下の作業は履歴の機能に関連しています。

■　「履歴ウィンドウを表示するには」

- 「履歴ウィンドウのドッキング状態を切り替えるには」
- 「履歴ウィンドウの常に前面に表示状態を切り替えるには」
- 「履歴ウィンドウをデフォルトの設定にリセットするには」

▶▶ 履歴ウィンドウを表示するには

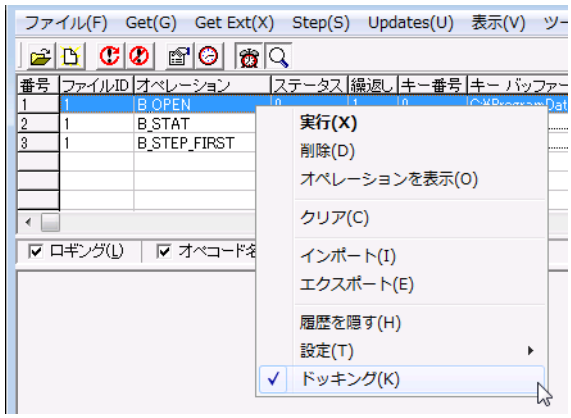
1 ［**表示**］>［**履歴**］を選択するか、［**履歴**］ボタンをクリックします。

▶▶ 履歴ウィンドウのドッキング状態を切り替えるには

1 履歴ウィンドウが表示されていない場合は、表示させます（「履歴ウィンドウを表示するには」を参照してください）。

2 履歴項目ウィンドウ内で右クリックし、次の図に示すように［ドッキング］オプションのチェックをオンまたはオフにしてドッキング状態を切り替えます。

図 17 履歴ウィンドウのドッキングを解除するには



ドッキングされた場合、履歴ウィンドウはここで示すようにアプリケーション ウィンドウに連結されます。ドッキングされない場合、履歴ウィンドウは別個のウィンドウになります。

▶▶ 履歴ウィンドウの常に前面に表示状態を切り替えるには

1 履歴ウィンドウが表示されていない場合は、表示させます（「履歴ウィンドウを表示するには」を参照してください）。

2 この機能は、履歴ウィンドウがドッキングされていない状態でのみ使用できます（「履歴ウィンドウのドッキング状態を切り替えるには」を参照してください）。

3 履歴項目ウィンドウで表示を右クリックし、［**手前に表示**］にチェックを付けるかまたははずします。

▶▶ 履歴ウィンドウをデフォルトの設定にリセットするには

1 履歴ウィンドウが表示されていない場合は、表示させます（「履歴ウィンドウを表示するには」を参照してください）。

2 履歴項目ウィンドウで表示を右クリックし、［**設定**］>［**デフォルト**］を選択します。

Maintenance を使用した Btrieve データ ファイルの操作

15

Maintenance ツールを使用した Btrieve データ ファイルの操作

この章では、以下の項目について説明します。

Maintenance ユーティリティの概要

Zen には、対話型の Maintenance GUI およびコマンド ラインの Maintenance ツールが用意されています。どちらのバージョンも以下のファイル操作およびデータ操作を行います。

- 定義したファイルおよびキー仕様に基づいた新規データ ファイルの作成
- 既存データ ファイルのファイルおよびキー仕様の設定
- データ ファイルのオーナー ネームの設定とクリア
- データ ファイルのインデックスの作成と削除
- ASCII シーケンシャル データのインポートとエクスポート
- Zen データ ファイル間のデータのコピー
- 最後のバックアップからシステム エラーが発生する間に行ったファイルへの変更の回復

2 つのユーティリティは基本的に同じですが、機能が異なる部分が多少あります。たとえば、対話型 Maintenance ツールでは、定義したファイルおよびキー仕様に基づいてディスクリプション ファイルを作成することができますが、コマンド ライン Maintenance ツールでは、サーバー上でファイルの Continuous オペレーションを直接開始または停止することができます。

Maintenance ツールを使用する前に、ファイル、レコード、キー、セグメントなどの Btrieve の基本概念について熟知しておく必要があります。これらのトピックスの詳細については、『*Zen Programmer's Guide*』を参照してください。

メモ Zen 製品では次の 2 種類の Maintenance ツールを提供します。Btrieve および SQL の 2 つです。SQL Maintenance ツールは ODBC 経由でのリレーショナル アクセスに使用するデータ ソース名（DSN）をサポートします。

対話型 Maintenance ツール

対話型 Maintenance ツールは、Windows 32 ビットおよび 64 ビット プラットフォームで実行される Windows アプリケーションです。このツールは、グラフィカル インターフユーザェイスを使用した場合、またはディスクリプション ファイルを作成したい場合に使用します。このセクションには次の主なトピックがあります。

- 「ファイル情報エディター」
- 「オーナー ネームの管理」
- 「情報レポート」
- 「インデックス」
- 「データ」

それぞれの主なトピックにはそのトピックに固有のタスクがあります。

拡張ファイルのサポート

MicroKernel データ ファイルは、オペレーティング システムのファイル サイズ上限を超えた容量を持つことが可能です。データを、MicroKernel の拡張ファイルからシーケンシャル ファイルにエクスポートした際、実際の形式の違いから、そのシーケンシャル ファイルの容量がデータベース エンジンのファイル サイズ上限を超える場合があります。

大きなサイズのファイルをエクスポートする際、対話型 Maintenance ツールでは、シーケンシャル ファイルがファイル サイズの上限（2 GB）を超えていることを検出すると、エクステンション ファイルの作成が開始されます。この処理は自動的に行われます。エクステンション ファイルおよび元のシーケンシャル ファイルは、同じボリューム内に存在する必要があります。ファイルのサイズの制限は、オペレーティング システムやファイル システムによって異なるので注意してください。2 GB というサイズはデータベース エンジンでサポートされる制限サイズです。

エクステンション ファイルの名前には、ベース ファイルと似た名前を付ける方式が使用されます。エクステンション ファイルを示すのに、ネイティブな MicroKernel エンジン エクステンション ファイルがキャレット（"^"）を使用するのに対し、シーケンシャル ファイルのエクステンション ファイルはチルダ（"~"）を使用するので、同じベース ファイル名を持つ既存のエンジン拡張ファイルを上書きするのを防ぎます。最初のエクスポート エクステンション ファイルには、ベース ファイル名に ".~01" という拡張子が付きます。次のエクステンション ファイルには、ベース ファイル名に ".~02" というように拡張子が付けられます。これらの拡張子は 16 進形式で追加されます。

名前付け規則は 255 までのエクステンション ファイルをサポートします。したがって、最大ファイル サイズは 256 GB です。

また、シーケンシャル ファイルからデータをインポートする場合は、ファイルが拡張されているかどうかがチェックされ、エクステンション ファイルからデータがロードされます。

長いファイル名と埋め込みスペースのサポート

スペースを含む長いファイル名は、サポートされるすべてのオペレーティング システムで使用できます。参照対象となるすべてのファイル名には、埋め込みスペースを含むことが可能であり、また 8 バイトを超える名前も使用可能です。

Btrieve の古いバージョンでは、Open や Create など、パスに基づくオペレーションの際に、ファイル名の最後にスペースを追加することができました。このバージョンでも、デフォルトとしてこれが設定されており、操作に支障をきたしませんが、バージョン埋め込みスペースを含むファイル名やディレクトリ名も正しく認識する機能を利用したい場合は、クライアント リクエスターの［**スペースを含むファイル / ディレクトリ名**］設定を**オン**に設定します。デフォルトの設定はオンです。

このオプションをオフに設定した場合でも、埋め込みスペースを含む名前を持つファイルにアクセスする際に、アプリケーションがファイルを開く、または作成する BTRV/BTRVID/BTRCALL/BTRCALLID コールを行うとき、その名前を二重引用符で囲めば、使用することができます。

レコードおよびページ圧縮

Zen ではレコードおよびページによる 2 種類のデータ圧縮を提供します。これら 2 種類のデータ圧縮は、別々に使われることもあれば、一緒に使われることもあります。2 つの圧縮タイプの主な目的はいずれも、データ ファイルのサイズを減らし、データのタイプとデータ操作のタイプに応じてより速いパフォーマンスを提供することです。

レコード圧縮

レコード圧縮には 6.0 以上のファイル形式が必要です。レコード圧縮により、多数の繰り返し文字を含むレコードの格納に必要なスペースを大幅に削減できます。データベース エンジンは、5 つ以上連続する同一文字を 3 バイトに圧縮します。

ファイルの作成時、指定されたレコード長に余裕を持たせるため、データベース エンジンは自動的に指定されたより大きいページ サイズを使用します。圧縮されていないレコードが、使用できる最大ページに収まりきらないほど大きい場合、データベース エンジンは自動的にレコード圧縮をオンにします。

圧縮レコードの最終的な長さは、ファイルに書き込まれるまで決定されないため、データベース エンジンはレコード圧縮されたファイルを可変長レコード ファイルとして作成します。レコードの圧縮されたイメージは可変長レコードとして格納されます。アプリケーションがレコードの追加、更新、削除を頻繁に行うと、個々のレコードはいくつかのファイル ページに断片化されます。データベース エンジンは 1 つのレコードを取得するために複数のファイル ページを読み取らなければならない場合があるので、この断片化によってアクセス時間が遅くなるおそれがあります。

『*Zen Programmer's Guide*』の「ページ サイズの選択」、「ファイル サイズの予測」および「レコード圧縮」を参照してください。

ページ圧縮

ページ圧縮には 9.5 以上のファイル形式が必要です。内部的には、Zen データ ファイルはさまざまな種類のページの連続です。ページ圧縮は、ファイル内のデータ ページの圧縮と復元を制御します。

ファイルが物理ストレージから読み取られると、データ ページは復元されてメモリ キャッシュに保持されます。レコードの読み取りと更新は、メモリ キャッシュ内の圧縮されていないデータに対して行われます。書き込み操作が行われると、データ ページは圧縮されて物理ストレージに書き込まれます。圧縮されたページが、次回アクセスされるまで保持されるかどうかは、キャッシュ管理によって異なります。

データの種類によって圧縮しても意味がない場合、データベース エンジンはそのデータを圧縮しないで物理ストレージに書き込みます。

圧縮の使用時期の判断

レコード圧縮、ページ圧縮、あるいはその両方を使用することで得られる利点は、圧縮されるデータのタイプによって異なります。次の表では、データ圧縮を使用するかどうかを判断するための**一般的な**要因を説明します。

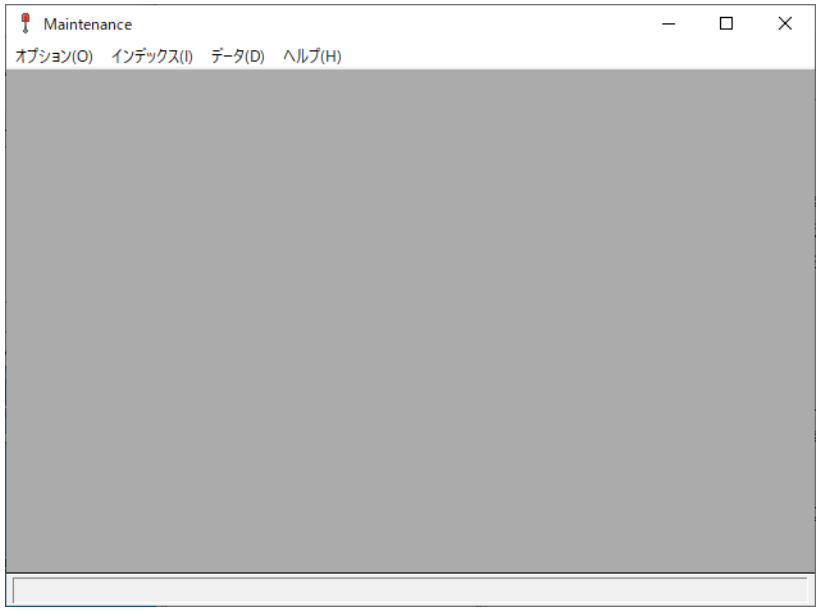
表 65　データ圧縮がふさわしいか考慮するための要因

圧縮		考慮する要因
レコード	ページ	
X		レコード圧縮は、以下の条件の場合に最も効果的です。 • 各レコードに文字の繰り返しが多数含まれる可能性がある。たとえば、レコードにいくつかのフィールドが含まれており、レコードをファイルに挿入するときにそれらのフィールドはすべてタスクによって空白に初期化される可能性があります。レコード圧縮は、これらのフィールドがほかの値を含むフィールドで分割される場合でなく、レコード内で 1 つにグループ化される場合に有効です。 • データベース エンジンを実行するコンピューターが、圧縮バッファーで必要となる追加メモリを提供できる。 • レコードは、変更されることに比べ、より頻繁に読み取られる。 レコードの固定長部分が「ページ サイズ - オーバーヘッド」より長い場合、自動的に圧縮が使用されます。 レコード圧縮は、キー オンリー ファイルやブランク トランケーションを使用するファイルには使用できません。
	X	ページ圧縮は、以下の条件の場合に最も効果的です。 • データが ZIP タイプの圧縮アルゴリズムを使用して高度に圧縮可能である。ページ圧縮によってファイル サイズが 1/4 程度に著しく縮小できる場合、ファイルのパフォーマンスはかなり向上します。 • ページは、追加、更新、削除されることに比べ、より頻繁に読み取られる。 圧縮しても意味がない場合、データベース エンジンはデータ ページを圧縮しないで物理ストレージに書き込みます。
X	X	レコード圧縮およびページ圧縮の使用は、レコードに大きな空白部分が含まれる場合と、ページが追加、更新、削除されることに比べ頻繁に読み取られる場合に最も効果的です。

Maintenance ツールのインターフェイス

オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から、あるいは Zen Control Center の［**ツール**］メニューから Maintenance ユーティリティにアクセスします。Maintenance ユーティリティのメイン ウィンドウは次のようになっています。

図 18 Maintenance ツールのメイン ウィンドウ



メニュー オプション

対話型 Maintenance ツールには、以下のメニューが用意されています。

オプション	**ファイル情報エディター**の表示、オーナー ネームの設定およびクリア、ファイル情報レポートの生成、ツールの終了を行います。
インデックス	インデックスの作成と削除を行います。
データ	ASCII ファイルからのデータのロード、ASCII ファイルへのデータの保存、データ ファイル間のレコードのコピー、および最後のバックアップからシステム エラーが発生するまでの間に行ったデータ ファイルへの変更を回復するためのロール フォワードを行います。
ヘルプ	Maintenance ツールのヘルプを起動します。

ヘルプの起動

Maintenance ツールのヘルプを使用するには、ヘルプの必要なダイアログ ボックスで［**ヘルプ**］をクリックするか、以下のような［**ヘルプ**］メニューのコマンドを選択します。

ヘルプの起動	Maintenance ツールのヘルプ トピックが表示されます。
バージョン情報	著作権情報および製品のバージョン番号が表示されます。

ファイル情報エディター

このセクションでは、構築するファイルおよびキー仕様に基づいて新しいファイルを作成する［ファイル情報エディター］に関する一般的な説明を行います。このエディターでは、既存ファイルの情報をロードできるため、既存データ ファイルのファイル仕様やキー仕様の確認に便利です。また、既存ファイルのファイル仕様やキー仕様に基づいた新規ファイルの作成も可能です（コマンド ライン Maintenance ツールである butil の clone コマンドと同様）。

注意 同じディレクトリに、ファイル名が同一で拡張子のみが異なるようなファイルを置かないでください。たとえば、同じディレクトリ内のデータ ファイルの 1 つに Invoice.btr、もう 1 つに Invoice.mkd という名前を付けてはいけません。このような制限が設けられているのは、データベース エンジンがさまざまな機能でファイル名のみを使用し、ファイルの拡張子を無視するためです。ファイルの識別にはファイル名のみが使用されるため、ファイルの拡張子だけが異なるファイルは、データベース エンジンでは同一のものであると認識されます。

ファイル情報エディターを開くには、［**オプション**］>［**情報エディターの表示**］をクリックします。

図 19 ファイル情報エディター

［ファイル情報エディター］ダイアログの項目

エディターの上部には、以下のボタンが表示されます。

情報のロード	既存ファイルの情報をロードします。ファイル情報をロードしても、ファイルが編集されることはありません。既存ファイルに関する情報のコピーのみがロードされます。
ファイルの作成	ダイアログ ボックスの現在の情報に基づいて新規ファイルを作成します。
デフォルトに設定	設定をデフォルト値に戻します。
ディスクリプション コメント	ディスクリプション ファイルを作成する場合は、ファイルに関するメモを入力することができます。
ヘルプ	［**ファイル情報エディター**］ダイアログ ボックスのヘルプを表示します。

データ ファイル情報

［**ファイル情報エディター**］の上部の［データ ファイル情報］領域には、以下のコントロールが含まれます。

オーナー ネーム　ファイルのオーナー ネームが存在する場合は、アスタリスク文字列として表示されます。

バージョン　ファイルの全属性を読み取ることができるデータベース エンジンの最も古いバージョンを表します。たとえば、バージョン 9.5 の データベース エンジンでファイルを作成したが、バージョン 9.5 特有の属性を使用しなかった場合、Maintenance ツールでは、バージョン 9.0 と表示されます。ファイル形式のバージョンについては「ファイル バージョンに関する注意」を参照してください。

総レコード数　ファイルのレコード総数を表します。

ファイル仕様

［ファイル情報エディター］の中央部に［**ファイル仕様**］領域があります。表 66 はこのボックス内のコントロールの説明です。

表 66　［ファイル仕様］のコントロール

コントロール	説明	範囲	デフォルト
レコード長	ファイルの固定長レコードの論理データ レコード長（バイト単位）を設定します。 レコード長およびオーバーヘッドの説明については、『*Zen Programmer's Guide*』の「レコード長」セクションを参照してください。	最低 4 バイト。最大値はファイル バージョンによって異なります。指定されたレコード長が、ページ サイズからオーバヘッドを差し引いた数値を超える場合、データベース エンジンは自動的にそのファイル形式で次に使用可能なページ サイズになるよう試行します。レコード長が、最大ページ サイズからオーバーヘッドを差し引いた数値を超える場合、エンジンはレコードの圧縮をオンにします。	100
ページ サイズ	ファイルの物理ページ サイズ（バイト単位）を設定します。	512 － 4096：9.0 より前のファイル バージョンの場合（512 バイトの倍数で最大 4096 バイト） 512、1024、1536、2048、2560、3072、3584、4096 または 8192：ファイル バージョンが 9.0 の場合 1024、2048、4096、8192 または 16384：ファイル バージョンが 9.5 の場合 4096、8192 または 16384：ファイル バージョンが 13.0 の場合	4096
キー数	（キー セグメントと対照して）エディターで現在定義されている識別キーの数を表します。キー リストのキーの数が反映されます。	0 – 119	0
セグメント数	エディターで現在定義されているキー セグメントの数を表します。セグメント リストのセグメントの数が反映されます。	0 － 119：9.5 より前のファイル バージョンの場合。 0 － 420：ファイル バージョン 9.5 の場合。 0 － 378：ファイル バージョン 13.0 の場合。	0

表 66 ［ファイル仕様］のコントロール

コントロール	説明	範囲	デフォルト
リンク可能キー	将来のリンク重複キーに予約する 8 バイト プレースホルダーの数を設定します。既存データ ファイルの情報をロードしている場合、そのファイルで現在使用可能なプレースホルダーの数がこの数値に反映されます。元から予約してあるプレースホルダーの数は、ファイルに保存されません。	0 – 119	3
キーオンリー	ファイルがキー オンリーかどうかを表します。レコード タ圧縮または可変長レコードを有効にした場合、およびファイルに複数のキーを定義した場合は、この設定は使用できません。	オンまたはオフ	オフ
インデックス バランス	ファイルにおけるキー ページ管理のインデックス バランスの使用を設定します。	オンまたはオフ	オフ
プリアロケーション	ファイルにおけるプリアロケート ページの使用を設定します。	オンまたはオフ	オフ
ページ数	ファイル作成時にプリアロケートするページの数を設定します。［プリアロケーション］がオンの場合にのみ設定できます。既存データ ファイルの情報をロードしている場合、そのファイルの未使用の残りプリアロケート ページ数がこの数値に反映されます。元からプリアロケートしてあるページの数は、ファイルに保存されません。	1 – 65535	0
レコード圧縮	ファイルにおけるレコード圧縮の使用を設定します。キー オンリー ファイル、またはブランク トランケーションを使用するファイルには対応していません。「レコードおよびページ圧縮」も参照してください。	オンまたはオフ	オフ
ページ圧縮	ファイルにおけるページ圧縮の使用を設定します。「レコードおよびページ圧縮」も参照してください。	オンまたはオフ	オフ
可変長レコード	ファイルに、可変長レコードを含むことができるかどうかを設定します。	オンまたはオフ	オフ
ブランク トランケーション	ディスク領域を確保するために、ファイルが可変長レコードに対してブランク トランケーションを使用するかどうかを設定します。この設定は、可変長レコード有効時のみ可能です。	オンまたはオフ	オフ
VAT	ファイルが、長いレコード内のデータへ高速にアクセスするため、可変長部割り当てテーブルを使用するかどうかを設定します。この設定は、可変長レコード有効時のみ可能です。	オンまたはオフ	オフ
% 空きスペース	新規可変ページ作成前に、ファイルの可変ページが確保する未使用スペースの量を設定します。この設定は、［レコード圧縮］または［可変長レコード］がオンの場合にのみ設定できます。	5、10、20、30	5

キー

ダイアログ ボックスの左下部には［**キー**］グループ ボックスがあります。表 67 はこの領域のコントロールの説明です。これらのコントロールは、現在のキー セグメントだけでなく、［**キー**］リストで選択されているキーに固有です。これらの設定を変更した場合、指定したキーの**すべての**セグメントに対して変更が適用されます。

表 67 ［キー］のコントロール

コントロール	説明	デフォルト
重複可能	キーが、重複する値（リンク重複）を持つことができるかどうかを設定します。 「重複キーの操作方法」を参照してください。	オン
変更可能	キーの値を、作成後に変更できるかどうかを設定します。キー値の変更を許可してもパフォーマンスには影響しません。キー ページは、実際のキー値が変更された場合にのみ更新され、特定のレコードのキー以外のフィールドが変更されても更新されません。	オン
繰り返し重複	重複キー値を格納するのにデータベース エンジンがくり返し重複の手法を使用するよう指定します。 「重複キーの操作方法」を参照してください。	オフ
散布キー（ヌル キー）	散布キーには、ファイル内にあるレコード数よりも少ない数のキー値が含まれます。インデックスから除外するキー値を指定するには、この次にある 2 つのコントロールを参照してください。この設定は、ヌル可能なセグメントが含まれるキーの場合のみ可能です。	オフ
全セグメント（ヌル）	レコードのすべてのキー セグメントにヌル値が含まれる場合に、そのレコードをインデックスに含まれないよう設定します。この設定は、散布キー（ヌル キー）有効時のみ可能です。キー フラグ 0x0008 に相当します。セグメントがヌルとして評価されるかどうかは、そのフィールドのヌル インジケーター セグメントだけを基に判断されます。そのフィールドの内容は評価されません。	オフ
一部セグメント（マニュアル）	1 つまたは複数のキー セグメントにヌル値が含まれる場合に、そのレコードをインデックスに含めないよう指定します。この設定は、散布キー（ヌル キー）有効時のみ可能です。キー フラグ 0x0200 に相当します。セグメントがヌルとして評価されるかどうかは、そのフィールドのヌル インジケーター セグメントだけを基に判断されます。そのフィールドの内容は評価されません。	オフ
ACS 情報	キーのオルタネート コレーティング シーケンス（ACS）を設定します。この設定は、キーのセグメントに対する［**ACS を使用する**］チェック ボックスがオンの場合にのみ使用できます。	オフ
ユニーク値	ファイルにある重複しないキー値の数を表します。これは、既存データ ファイルの情報をロードしている場合のみ表示されます。	適用外

［キー］リストおよび［セグメント］リスト

ダイアログ ボックス中央下部の［**キー**］リストには、ファイルに定義されているキー番号が表示されます。バージョン 6.*x* 以降のファイルでは、このキー番号が連続した数値である必要はありません。Maintenance ツールでは、選択されたキーの仕様が、ダイアログ ボックス左下部の［**キー**］ボックスに表示されます。

また、ダイアログ ボックス中央下部の［**セグメント**］リストには、［**キー**］リストで選択されたキーに定義されている、キー セグメント番号が表示されます。Maintenance ツールでは、選択されたセグメントの仕様が、ダイアログ ボックス右下部の［**セグメント**］ボックスに表示されます。

また、［**キー**］と［**セグメント**］リストには、以下のボタンが表示されます。

挿入	新しいキーまたはセグメントを定義します。
削除	選択されたキーまたはセグメントの仕様を削除します。
圧縮	キーの番号を再割り当てします。このボタンは、キー仕様を削除することによって生じたキー番号の不連続をなくすことができます。

これらのボタンは、作成するファイルのキー仕様を設定するもので、既存ファイルのキーに対して使用することはできません。既存ファイルのインデックスを作成または削除する場合は、「インデックスの作業」を参照してください。

キー セグメント

ダイアログ ボックスの右下部には［**キー セグメント**］グループ ボックスがあります。表 68 はこの領域のコントロールの説明です。これらのコントロールは、［**セグメント**］リストで選択されているセグメントに固有のものです。

表 68 ［キー セグメント］グループ ボックスのコントロール

コントロール	説明	デフォルト
データ型	キー セグメントのデータ型を指定します。 NULL データ型は、インデックスが 1 バイトのヌル インジケーター セグメントであることを示します。このセグメントはマルチ セグメント キー内にあり、NULL 型でないほかのキー セグメントの前にある必要があります。このキー タイプに使用される Btrieve API の番号は 255 です。	String
位置	レコード内でのこのキー セグメントの相対的な開始位置を、数値で設定します。この数値は、レコード長を超えて設定することはできません。	1
長さ	キー セグメントの長さ（バイト単位）を設定します。この数値は、セグメントのデータ型に指定された制限を超えることはできません。キー位置とキー長の合計も、レコード長を超えないように注意してください。	10
ヌル値（16 進）	キー セグメントのヌル文字値（16 進数）を設定します。この設定は、キーに対する［**ヌル キー**］チェック ボックスがオンの場合にのみ使用できます。	バイナリ ゼロ
大小文字無視	セグメントで大文字小文字が区別されるかどうかを設定します。この設定は、STRING、LSTRING、ZSTRING のデータ型、または ACS を使用しないキーに対してのみ有効です。	オン
降順ソート	キー セグメント値を降順（大きいものから小さいものへ）でソートするかどうかを指定します。	オフ
ACS を使用する	セグメントで、キーに定義されたオルタネート コレーティング シーケンスを使用するかどうかを設定します。この設定は、大文字小文字が区別される `STRING`、`LSTRING`、`ZSTRING` のデータ型に対してのみ使用できます。	オフ
ヌル値の離散順序	ヌル値の離散順序は、ヌル インジケーター セグメント（NIS）で、MicroKernel エンジンが NIS を Boolean として扱って非ゼロ値をすべてヌルと見なすのか、1 バイトの Integer として扱ってゼロが非ヌルでその他すべての値を異なるタイプのヌルと見なすのかを決定するのに使用されます。この場合、これらは離散値としてソートされます。Btrieve API は NO_CASE フラグ 0x0400 がこれまで Integer 値に使用されていなかったので、これを使用し、離散順序ソートが実行されるようにします。	オフ

重複キーの操作方法

複数のレコードが、インデックス キーに同一の重複した値を持つことができます。重複キー値を持つレコードを記憶しておく 2 つの方法は、リンク重複および繰り返し重複と呼びます。

リンク重複

リンク重複の手法では連鎖技術を使用し、グループ内の各レコードをポインターを使用して隣のレコードと連結します。同一インデックス ページ上の各エントリは 1 組のレコード ポインターを持ち、キー値が重複しているレコードの連鎖の最初と最後のリンクを示します。このため、各キー ページのエントリは繰り返し重複インデックスより 4 バイト長くなります。さらに、データ ページの各レコードでは、リンク重複インデックスごとに 8 バイトのオーバーヘッドが必要です。この 8 バイトは、連鎖中の次のレコードと前のレコードを指す 2 つのレコード ポインターです。

最初のレコード ポインターは、最初に、または最も古い時期に格納されたレコードのアドレスを保持します。2 番目のポインターは最近または最後に格納されたレコードのアドレスを保持します。最初のレコードが書き込まれた後でほかのレコードが追加される前には、キー ページ エントリ上の両方のポインターに最初のレコードのアドレスが保持されています。その後にレコードが追加されると、2 番目のポインターが追加された各レコードを指すように変更されます。このことにより、最後のレコードのレコード ポインターが、レコードの追加時にデータ ページ内に構築される前のレコードの連鎖のリンクとして使用されること、および前のレコードの検索に使用されることが可能になります。

繰り返し重複

繰り返し重複の手法では、各重複キー値はインデックス ページとデータ ページ上のレコード内の両方に格納されます。各キー値には、レコード ポインターが 2 つではなく 1 つだけあります。この手法ではデータ レコードの連鎖が不要で、レコードごとの 8 バイトのオーバーヘッドを節約します。キー値が重複レコードごとに繰り返されるため、インデックスのサイズが大きくなるという影響があります。

手法の比較

リンク重複および繰り返し重複の手法は、以下の基準によって比較することができます。

- 順序付け
- ストレージ
- パフォーマンス
- 並行性

順序付け

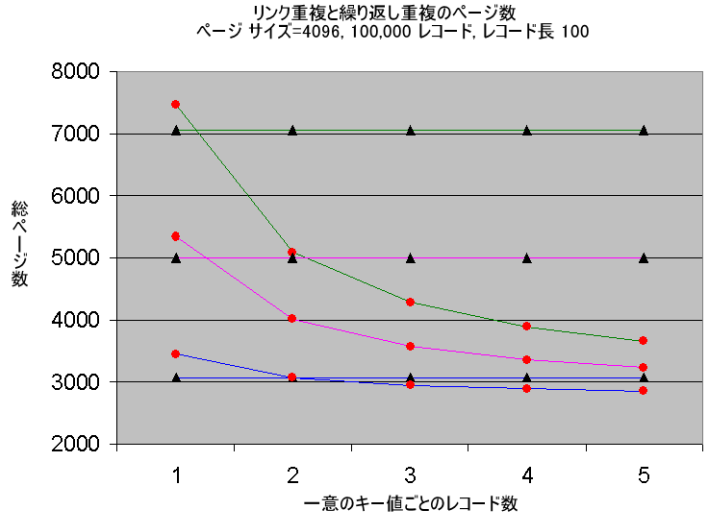
リンク重複インデックスは、データが追加された順序で重複を取得します。繰り返し重複インデックスは、データがファイル内に存在する順序で重複を取得します。ファイル内での位置は制御できないため、順序はランダムであると考える必要があります。

ストレージ

リンク重複インデックスでは、各重複キー値の最初の出現で 12 バイト余分に必要になります。これには、各レコードについての 8 バイトとキー ページ エントリとしての 4 バイトが含まれます。ただし、重複レコードはキー ページでこれ以外の余計な空間を必要とせず、レコードごとに 8 バイトが追加されるだけです。したがって、キー値ごとの重複の数が増加してキー値のサイズが増加しても、リンク重複インデックスはキー ページが使用するストレージ空間を著しく節約することができます。ただし、重複キーを持つレコードがほとんどファイルに格納されていないか、キー長が非常に短い、またはその両方に該当する場合は、使用するストレージ空間は増加します。

次の図で、リンク重複インデックスの使用によって節約されるストレージ空間の量を例示します。リンク重複インデックスは、キー値ごとの重複レコード数が少ない場合にはより多くの空間を必要とするので注意してください。しかし、キー値ごとの重複レコード数が増えるにつれ、リンク重複インデックスが必要とするページ数は少なくなり、ストレージ空間を著しく節約します。

図 20 重複キー手法によるページ数の比較



凡例： ● = リンク重複 ▲ = 繰り返し重複

先頭の 2 行はキー長 100 を表します
中央の 2 行はキー長 50 を表します
下部の 2 行はキー長 4 を表します

パフォーマンス

インデックス検索に関わるページが少ない場合、ディスクから読み取るページ数が少なくなるため、結果として速いパフォーマンスが得られます。リンク重複手法は一般的に少しの物理ストレージ空間しか使用しないため、速いパフォーマンスが得られます。繰り返し重複手法は、重複のあるキー数が少ない場合にのみパフォーマンスでの利点があります。

並行性

データベース エンジンは、複数の並行トランザクションが同一ファイルに対し同時にアクティブである場合に、ページ レベルの並行性を提供します。これは、キー ページへのほとんどの変更と、データ ページへのすべての変更に適用されます。並行性とは、同一ページが別個のトランザクションからの複数の保留中の変更を同時に持つことができることを意味します。また、これらのトランザクションはどのような順序でもコミットすることができます。繰り返し重複インデックスはこの並行性を最もよく利用しています。

リンク重複インデックスでは、繰り返し重複には存在しない並行性の制限が追加されます。新しい重複データが作成された場合、そのレコードはリストの最後で別のレコードにリンクされます。このレコードのリンクにより、1 つではなく 2 つのレコードがロックされることになります。すべての重複データはリンク レコードの連鎖の最後に追加されるため、重複は一度に 1 つしか追加できません。

このようなレコード ロックの衝突により、通常、ほかのクライアントは最初のトランザクションがコミットされるまで待機状態になります。並行動作が行われる環境では、すべての新規レコードが同じ重複値を使用する場合、並行性は事実上一度に 1 トランザクションというレベルまで低減します。さらに、トランザクションが大きいかまたは長く継続する場合、この連続状態がパフォーマンスに非常に大きく影響します。

並行環境で更新されるデータベースに繰り返し重複インデックスを使用した場合、パフォーマンスは一般的に向上します。したがって、リンク重複手法を使用せざるを得ない理由がある場合を除き、並行環境で更新されるデータベースには繰り返し重複インデックスを使用してください。

情報エディターでの作業

ファイル情報エディターを使用して次の作業を行うことができます。

- 「既存ファイルからの情報のロード」
- 「新規ファイルの作成」
- 「Btrieve データ ファイルのコンパクト化」
- 「キーのオルタネート コレーティング シーケンスの指定」

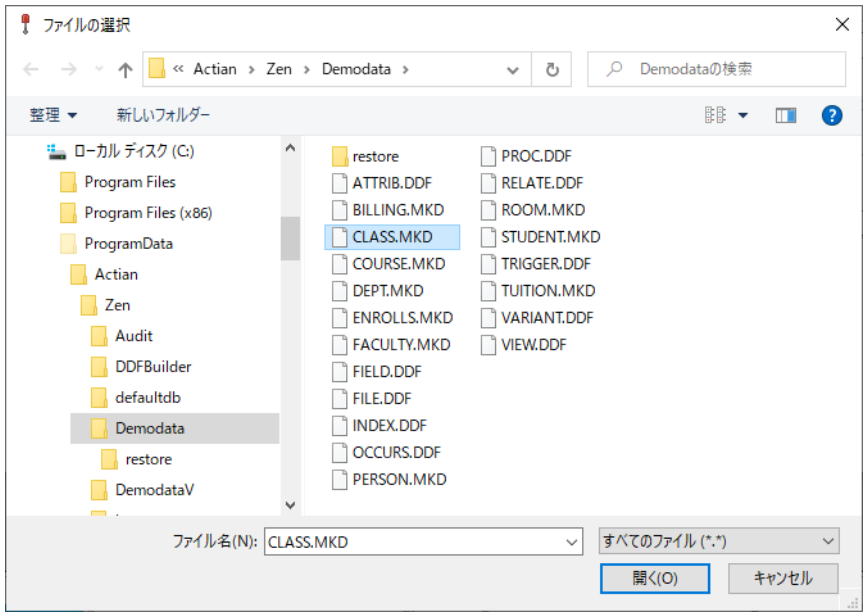
既存ファイルからの情報のロード

既存ファイルから情報をロードした場合、既存ファイルを編集するわけではありません。既存ファイルに関する情報のコピーのみがロードされます。一般的に、ファイル情報エディターでほかの作業をする前にファイルをロードしますが、これは必須ではありません。

▶▶ 既存データ ファイルからファイル情報エディターに情報をロードするには

1 ［ファイル情報エディター］ダイアログ ボックスの上部の［**情報のロード**］をクリックします。［**ファイルの選択**］ダイアログ ボックスが表示されます。

図 21 ［ファイルの選択］ダイアログ ボックス



2 情報をロードするファイルの名前とパスを指定します。デフォルトでは、データ ファイルの拡張子は .mkd になっていますが、それ以外の拡張子、または拡張子なしも可能です。

Maintenance ツールでは、指定されたファイルを、まずデータ ファイルとして開こうとします。ファイルにオーナー ネームが必要な場合は、オーナー ネームを求めるメッセージが表示されます。指定されたファイルがデータ ファイルでない場合、このツールではそのファイルを、ディスクリプション ファイルとして開こうとします。

新規ファイルの作成

ファイル情報エディターで、現在の情報または入力した新しい情報に基づいた新規ファイルを作成することができます。

▶▶ ファイル情報エディターの現在の情報に基づいて新規ファイルを作成するには

1 ［**ファイル情報エディター**］ダイアログ ボックスの上部の［**ファイルの作成**］をクリックします。［**ファイルの作成**］ダイアログ ボックスが表示されます。

図 22 ［ファイルの作成］ダイアログ ボックス

2 ［**ファイルの作成**］ダイアログ ボックスで設定を行います。各コントロールについては、表 69 を参照してください。

表 69 ［ファイルの作成］ダイアログのコントロール

コントロール	説明	デフォルト
ファイル名	ファイルの名前とパスを指定します。デフォルトでは、データ ファイルの拡張子は .mkd になっています。	適用外
ファイルの種類	作成するファイルの種類を指定します。ディスクリプション ファイルを作成する場合は、［**インデックス オンリー**］オプションを使用して、butil ツールで既存データ ファイルへのインデックス追加に使用できるディスクリプション ファイルを作成できます（詳細については、「インデックスの作成」を参照してください）。	MicroKernel 互換
システム データ	ファイルにシステム データを含めるかどうかを設定します。［**エンジン設定を使用**］を選択した場合は、［システム データ］設定オプションの設定が適用されます。［**システム データなし**］を選択した場合、エンジンの設定にかかわらず、システム データは作成されません。［**システム データを強要**］または［**システム データ v2 を強要**］を選択した場合は、エンジンの設定にかかわらず、システム データが作成されます。 システム データ v2 には 13.0 ファイル バージョンが必要です。	エンジン設定を使用

ディスクリプション ファイルへのコメントの追加

コメントは、ディスクリプション ファイルを作成したときに、その先頭に出力されます。たとえば、コメント「これは私のファイルです」はディスクリプション ファイルの先頭に /* これは私のファイルです */ と表示されます。ディスクリプション ファイル作成後にコメントを追加する場合は、コメントを追加してファイルを作成し直す必要があります。

▶▶ ディスクリプション ファイルにコメントを追加するには

1 ［**ディスクリプション コメント**］をクリックします。［**ディスクリプション ファイル コメント**］ダイアログ ボックスが表示されます。

2 全角で 2,560 文字（半角で 5120 文字）以内のコメントを入力します。

3 コメント入力後、［OK］をクリックします。

Btrieve データ ファイルのコンパクト化

未使用スペースを除去することにより Btrieve データ ファイルのコンパクト化を行うことができ、通常ファイルのサイズを小さくすることができます。この処理は、コマンド ライン Maintenance ツール（「Btrieve データ ファイルをコンパクト化するには」を参照）を使用して実行することもできます。

▶▶ Btrieve ファイルをコンパクト化するには

1 ファイル情報エディターの［**情報のロード**］をクリックし、コンパクト化するファイルを選択します。

2 ［**ファイルの作成**］をクリックし、［**ファイルの作成**］ダイアログ ボックスに、複製する新しいファイルの名前を入力して［OK］をクリックします。

3 メイン ウィンドウの［**データ**］メニューから［**保存**］をクリックします。［**データの保存**］ダイアログ ボックスの［**保存元 MicroKernel ファイル**］ボックスに元のファイルの名前を入力し、［**保存先シーケンシャル ファイル**］ボックスに出力ファイルの名前（たとえば＜**元のファイル名** >.out）を入力します。

4 ［**実行**］をクリックします。［**データの保存**］ダイアログ ボックスに、保存の結果が表示されます。［**閉じる**］をクリックします。

5 ［**データ**］メニューの［**ロード**］をクリックします。［**データのロード**］ダイアログ　ボックスの［**ロード元シーケンシャル ファイル**］ボックスに、保存したシーケンシャル データ ファイルの名前を入力します。ステップ 2 で作成した複製ファイルの名前を［**ロード先 MicroKernel ファイル**］ボックスに入力します。

6 ［**実行**］をクリックします。［**データのロード**］ダイアログ　ボックスに、ロードの結果が表示されます。［**閉じる**］をクリックします。

元のファイルと複製されたファイルのサイズを比較すると、コンパクト化の結果を確認できます。

キーのオルタネート コレーティング シーケンスの指定

オルタネート コレーティング シーケンス（ACS）を使用して、文字型のキー（STRING、LSTRING および ZSTRING）を標準 ASCII 照合順序とは異なる順序でソートすることができます。1 つまたは複数の照合順序を使用して、次のようにキーをソートすることができます。

- 独自のユーザー定義ソート順序による方法。この方法は、英数字（A ～ Z、a ～ z、0 ～ 9）を非英数字（# など）と混用するソート順序を必要とするような場合に使用します。
- インターナショナル ソート規則（ISR）による方法。これはマルチバイト照合要素、区分発音符、文字の拡張および短縮など、言語固有の照合順序に対応しています。

ファイルには、キーごとに異なる ACS を持つことができますが、1 つのキーには 1 つの ACS のみです。したがって、キーがセグメント化されている場合、各セグメントはそのキーに指定された ACS を使用するか、または ACS をまったく使用しないかのいずれかになります。あるファイルに、一部のセグメントにだけ ACS が指定されているキーがある場合、Btrieve は指定されたセグメントのみ、ACS を使用してソートします。

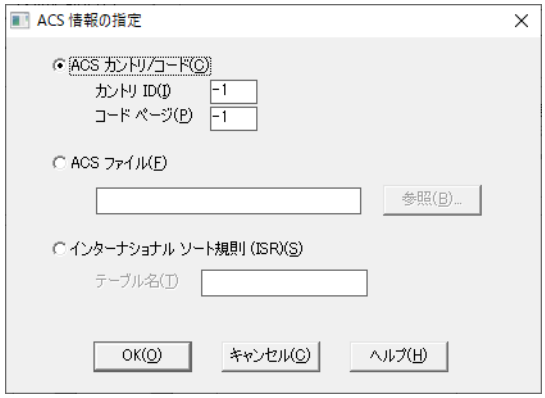
ISR テーブルは ISO の標準ロケール テーブルに基づいており、Zen によって提供されます。ISR テーブルは Zen のデータベース エンジンと一緒にインストールされた collate.cfg ファイルに格納されています。複数のデータ ファイルが 1 つの ISR を共有できます。

▶▶ キーのオルタネート コレーティング シーケンスを指定するには

1 ［**ACS 情報**］をクリックします。

Maintenance ツールに［**ACS 情報の指定**］ダイアログ ボックスが表示されます。

図 23 ［ACS 情報の指定］ダイアログ ボックス



2 ACS ファイル名またはインターナショナル ソート規則（ISR）のいずれかを、以下のように指定できます。

表 70 ACS 情報のコントロール

コントロール	説明	デフォルト
ACS カントリ / コード	使用されなくなりました。	適用外
ACS ファイル	オルタネート コレーティング シーケンス ファイルのフル パス名を指定します。	適用外
インターナショナル ソート規則	国際データをソートするために使用する、ISR テーブルを選択することができます。Zen には、『Zen Programmer's Guide』に一覧表示された ISR テーブルがあらかじめ用意されています。	

3 データ ファイルに ACS ファイル名を指定した場合は、ACS ファイルの内容がデータ ファイルにコピーされます（データ ファイルには、ACS ファイルの名前は含まれません）。ACS は、8 バイトの名前（UPPER など）で識別され、Maintenance ツールでは、データ ファイルの ACS 情報に、元の ACS のファイル名ではなく、この 8 バイトの名前が表示されます。

4 ディスクリプション ファイルに ACS ファイルを指定した場合は、ACS ファイルの実際のパスとファイル名がディスクリプション ファイルにコピーされます。その結果、ディスクリプション ファイルの ACS 情報を表示する場合、Maintenance ツールは指定した ACS ファイルを検索しようとします。

ISO で定義されている言語固有の照合順序を使用して文字列の値をソートする ACS を指定する場合は、ISR テーブル名を指定します。［**テーブル名**］フィールドへの入力は、半角で 16 文字までです。ISR の詳細については、開発者リファレンスの『*Zen Programmer's Guide*』を参照してください。

オーナー ネームの管理

MicroKernel は、個々のデータ ファイルにそれぞれオーナー ネームを設定することにより、データ ファイルへのアクセスを個別に制限することができます。オーナー ネームを提供するユーザーのみが、ファイルに対する読み取り / 書き込みが可能となります。このトピックでは、Maintenance ツールを使用したオーナー ネームの管理について説明します。詳細については、『*Advanced Operations Guide*』の「オーナー ネーム」トピックを参照してください。

オーナー ネームによる制限があるテーブルへリレーショナル アクセスしようとしたときに、そのオーナー ネームの文字列値を提供しないと ODBC エラーになります。たとえば、これは ZenCC でテーブルを削除しようとする場合などです。GRANT または SET OWNER ステートメントを使用すれば、セッション内で 1 つまたは複数のテーブルにオーナーネームを指定することができます。特定のユーザーまたはグループにアクセス権を付与して、ODBC 経由でテーブルをリレーショナル操作するには、GRANT ステートメントを使用します。Master ユーザーは GRANT ステートメントに正しいオーナー ネームを使用する必要があります。詳細については、『*SQL Engine Reference*』の 「GRANT」を参照してください。

SET OWNER ステートメントを使用して、現在のデータベース セッション中にファイル アクセスを可能にするオーナー ネームを 1 つまたは複数提供します。詳細については、『*SQL Engine Reference*』の 「SET OWNER」を参照してください。

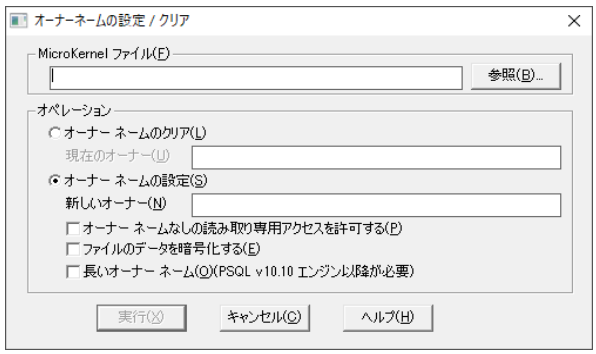
オーナー ネームを設定またはクリアする

オーナー ネームを設定すると、データ ファイルへのアクセスが制限されます。オーナー ネームをクリアすると、この制限が解除されます。

▶▶ オーナー ネームを設定またはクリアするには

1 Maintenance ツール ウィンドウで、[**オプション**] > [オーナー ネームの設定 / クリア] を選択して、[オーナー ネームの設定 / クリア] ダイアログを開きます。

図 24 [**オーナー ネームの設定 / クリア**] ダイアログ



2 [**MicroKernel ファイル**] ボックスで、オーナー ネームを設定またはクリアするファイルを入力します。オーナー ネームをクリアする場合は、[**オーナー ネームのクリア**] をクリックし、[**現在のオーナー**] フィールドにファイルのオーナー ネームを入力してそのクリアを許可します。

3 オーナー ネームを設定する場合は、[**オーナー ネームの設定**] をクリックし、[**新しいオーナー**] フィールドに新しいオーナー ネームを入力して、その後、希望するオプションを選択します。

- [**オーナー ネームなしの読み取り専用アクセスを許可する**] をオンにすると、すべてのユーザーが、データ ファイルへの読み取り専用アクセスを許可されます。
- [**ファイルのデータを暗号化する**] をオンにすると、デバッガーやファイル ダンプ ツールを使用したデータへの不正アクセスを防ぐことができます。暗号化と復号により処理時間が増えるため、使用する環境でデータのセキュリティが優先される場合にのみ、このチェック ボックスをオンにしてください。

- ［**長いオーナー ネーム**］を選択して、8 バイト長の短いオーナー ネームよりも長いオーナー ネームを作成します。長いオーナー ネームの長さは、ファイル形式によって異なります。詳細については、『*Advanced Operations Guide*』の「オーナー ネーム」トピックを参照してください。

4 ［**実行**］をクリックしてオプションを適用します。

メモ オーナー ネームを指定するには GRANT ステートメントを使用することもできます。『*SQL Engine Reference*』の「オーナー ネームによるアクセス権の付与」を参照してください。

情報レポート

ファイル情報レポートの作成は、データベース エンジンのトランザクション一貫性保持機能でファイルのログを作成できるかどうかを確認する際に便利です。レポートは、ファイルにシステム データがあるかどうか、キーがユニークであるかどうかを表示します（ユニーク キーには、重複が許可されていることを示す D フラグが付きません）。情報レポートはファイルのメタデータを提供します。この情報は、問題の解決に使用したり、よく似たファイルを作成するのに役立ちます。

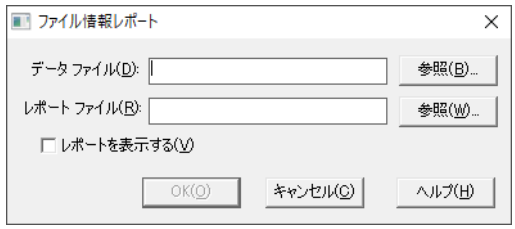
情報レポートの作業

情報レポートを作成する手順を以下に示します。

▶▶ 既存データ ファイルのファイル情報レポートを作成するには

1 メイン ウィンドウのメニューで、［**オプション**］>［情報レポートの作成］をクリックします。Maintenance ツールに［**ファイル情報レポート**］ダイアログ ボックスが表示されます。

図 25 ［ファイル情報レポート］ダイアログ ボックス



2 使用するデータ ファイルとレポートのファイル名を指定します。作成後にレポートを表示する場合は、［**レポートを表示する**］チェック ボックスをオンにします。

レポートを表示する設定にした場合、次のような［ファイルの表示］ウィンドウが表示されます。

図 26 ファイル情報レポートの例

```
File Statistics for person.mkd

File Version = 9.50
Page Size = 4096
Page Preallocation = No
Key Only = No
Extended = No
Total Number of Records = 1500
Record Length = 333
Record Compression = No
Page Compression = No
Variable Records = Yes
```

ファイル情報レポートの情報ヘッダーは、「ファイル情報エディター」に記述されているファイル情報エディターのコントロールに対応しています。

ファイル情報レポート下部の凡例には、レポートのキー / セグメントの部分で使用されている記号の説明が表示されます。この情報には、キーやキー セグメントの数、ファイル中のキーの位置、キー長などが含まれます。

凡例：

```
< = Descending Order
D = Duplicates Allowed
I = Case Insensitive
M = Modifiable
R = Repeat Duplicate
A = Any Segment (Manual)
L = All Segments (Null)
* = The values in this column are hexadecimal.
?? = Unknown
-- = Not Specified
```

インデックス

インデックスは、特定キーのすべてのキー値をソートするためのものです。Btrieve アクセスではオーバーラップするインデックス（列の一部を含むインデックス）が許可されます。ODBC を介したリレーショナル アクセスでは、オーバーラップするインデックスは許可されません。[エディターの起動] ボタンをクリックして表示できるファイル情報エディターを使用して、オーバーラップするインデックスを作成することができます。

インデックスの作業

インデックスに関連する以下の作業を行います。

- 「インデックスの作成」
- 「インデックスの削除」

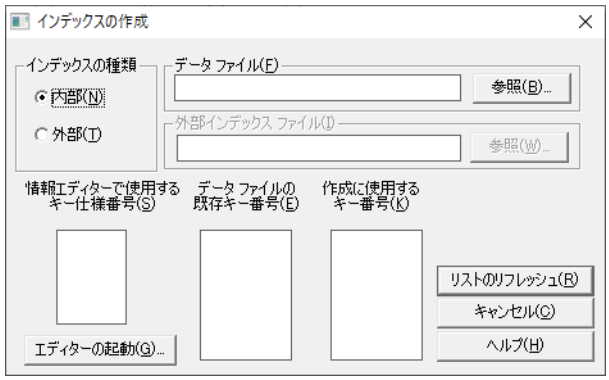
インデックスの作成

ファイルにキーが定義されていない場合、ファイルのインデックスを作成することはできません。ファイル情報エディター（「ファイル情報エディター」を参照）を使用してキーを作成することができます。

▶▶ インデックスを作成するには

1 メイン メニューで、[**インデックス**] > [**作成**] をクリックすると、[インデックスの作成] ダイアログ ボックスが開きます。

図 27 [インデックスの作成] ダイアログ ボックス



2 [**インデックスの作成**] ダイアログ ボックスで以下のオプション設定を行います。

インデックスの種類	内部インデックスと外部インデックスのどちらを作成するかを指定します。内部インデックスは、データ ファイルの一部として動的に管理されます。外部インデックスは、必要に応じて作成する別のファイルとなります。 外部インデックス ファイルは、指定するキーでソートされたレコードを含む標準データ ファイルです。各レコードは、以下の項目から成ります。 • 元のデータ ファイルにおけるレコードの物理位置を識別する 4 バイト アドレス • キー値
データ ファイル	インデックスを作成するデータ ファイルの名前を指定します。
外部インデックス ファイル	外部インデックスに作成するファイルの名前を指定します。内部インデックスには使用できません。

情報エディターで使用するキー仕様番号	**ファイル情報エディター**で定義されたキー番号が表示されます。
データ ファイルの既存キー番号	［**リストのリフレッシュ**］をクリックすると、ファイルに定義されているキー番号が表示されます。ファイルにシステム定義のログ キーが含まれている場合は、このリストに SYSKEY が表示されます。
作成に使用するキー番号	［**リストのリフレッシュ**］をクリックすると、ファイルに定義されていない、使用可能なキー番号が表示されます。インデックスの作成時に使用するキー番号を選択します。 ファイルにシステム定義のログ キー（システム データ）が含まれ、そのキーが削除されている場合は、リストに SYSKEY が表示されます。それを選択することにより、システム定義のログ キーを再びファイルへ追加することができます。

3 ［**エディターの起動**］をクリックすると、［**ファイル情報エディター**］ダイアログ ボックスにキーの詳細情報が表示されます。［**リストのリフレッシュ**］をクリックして、データ ファイルからキー情報を読み取ると、［**データ ファイルの既存キー番号**］リストと［**作成に使用するキー番号**］リストをリフレッシュできます。インデックスを作成する前に［**リストのリフレッシュ**］をクリックしてください。

4 ［**インデックスの作成**］ダイアログ ボックスの設定完了後、［**実行**］をクリックしてインデックスを作成します。インデックス作成の所要時間は、ファイルに含まれるデータの量によって異なります。

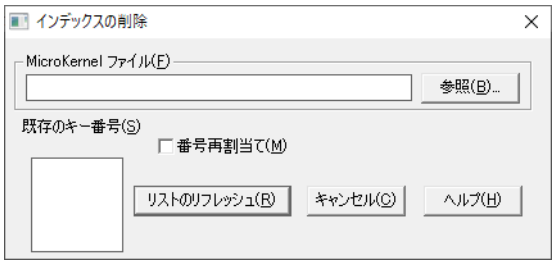
インデックスの削除

インデックスを削除する前に、アプリケーション プログラムが行うアクセスについてよく理解しておいてください。必要なインデックスがないと、GET NEXT などの機能が動作しません。その結果、アプリケーション プログラムは正常に機能しなくなります。

▶▶ インデックスを削除するには

1 メイン メニューで、［**インデックス**］>［削除］をクリックします。［**インデックスの削除**］ダイアログ ボックスが表示されます。

図 28 ［インデックスの削除］ダイアログ ボックス



2 ［**インデックスの削除**］ダイアログ ボックスで以下のオプション設定を行います。

MicroKernel ファイル	インデックスを削除するデータ ファイルの名前を指定します。
既存のキー番号	［**リストのリフレッシュ**］をクリックすると、ファイルに定義されているキー番号が表示されます。削除するインデックスのキー番号を選択します。ファイルにシステム定義のログ キーが含まれている場合は、リストに SYSKEY が表示され、それを選択することにより、システム定義のログ キーをファイルから削除することができます。
番号再割当て	キーの番号を再割り当てします。このチェック ボックスをオンにすると、インデックス削除によって生じたキー番号の欠番がなくなります。

3 ［**リストのリフレッシュ**］をクリックし、指定したファイルからキー情報を取得します。

データ

［データ］メニューのコマンドを使用して、データ ファイルのレコードをインポート、エクスポート、およびコピーすることができます。また、ロール フォワード機能を利用して、システム エラー発生後にデータを修復することも可能です。

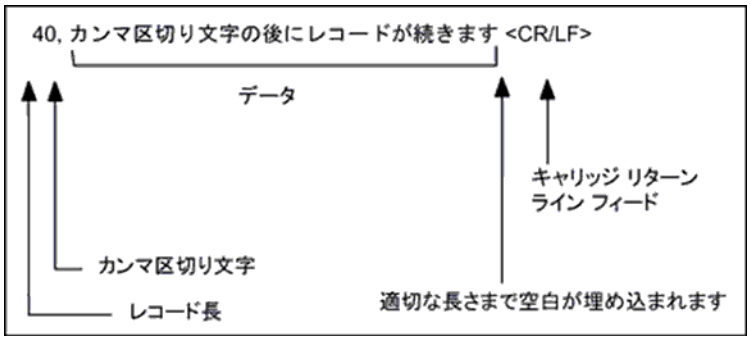
ASCII ファイル形式のインポートとエクスポート

データを保存すると、ASCII ファイルのレコードは、以下の形式になります。以下の仕様に従うことにより、ASCII テキスト エディターを使用してロード可能なファイルを作成できます。多くのテキスト エディターでは、バイナリ データを編集することができないので注意してください。

- 最初のフィールドは、レコードの長さを指定する左揃えの整数（ASCII 形式）です（この値の計算時、行末のキャリッジ リターン / ライン フィードは無視します）。このフィールドの値は、データ ファイルで指定されているレコード長と同じになります。
  - 固定長レコードを含むファイルでは、データ ファイルのレコード長と同じ長さを指定します。
  - 可変長レコードを含むファイルでは、データ ファイルの固定長レコード以上の長さを指定します。
- 長さフィールドの次に、区切り文字（カンマまたはスペース）が入ります。
- 区切り文字の次に、レコード　データが続きます。データの長さは、長さフィールドで指定したバイト数とまったく同じにします。テキスト エディターを使用してインポート ASCII ファイルを作成する場合は、各レコードが適切な長さになるように、必要に応じてスペースで埋めてください。
- 各行末には、キャリッジ リターン / ライン フィード（16 進数の 0D0A）を使用します。Maintenance ツールでは、データ ファイルにキャリッジ リターン / ライン フィードが挿入されません。
- ファイルの最終行には、ファイル終了文字（CTRL+Z または 16 進数の 1A）を使用します。多くのテキスト エディターでは、ファイルの最後にこの文字が自動的に挿入されます。

図 29 は、入力 ASCII ファイルのレコード形式です。この例では、データ ファイルに 40 バイトのレコード長が定義されています。

図 29　入力シーケンシャル ファイルのレコード形式



データに関する作業

以下のデータに関する作業は Maintenance ツールを使用して行うことができます。

- 「ASCII データをインポートするには」
- 「ASCII レコードをエクスポートするには」
- 「MicroKernel データ ファイル間でレコードをコピーするには」
- 最後のバックアップからシステム エラーが発生する間に行った、データ ファイルへの変更の回復（ロール フォワード）の方法は、『*Advanced Operations Guide*』の「ログ、バックアップおよび復元」の章を参照してください。

ASCII ファイルからのレコードのインポート

Maintenance ツールでは、ASCII ファイルから標準データ ファイルにレコードをインポートすることができます。この処理では、データ変換は行われません。テキスト エディター、または Maintenance ツールを使用してインポート ファイルを作成することが可能です（「ASCII ファイルへのレコードのエスクポート」を参照）。

▶▶ ASCII データをインポートするには

1 メイン メニューで［**データ**］>［ロード］をクリックします。［**データのロード**］ダイアログ ボックスが表示されます。

図 30 ［データのロード］ダイアログ ボックス

指定する ASCII ファイルは、「ASCII ファイル形式のインポートとエクスポート」で説明されている仕様に従っている必要があります。指定する標準データ ファイルのレコード長は、ASCII ファイル内のレコードと適合している必要があります。

2 ［**実行**］をクリックしてレコードをインポートします。

データのインポート中は、インポートされたレコードの数とパーセンテージ、およびステータス メッセージが表示されます。このとき、Maintenance ツールでは別の作業（新しい［**データのロード**］ダイアログ ボックスを開くなど）を続けることもできます。

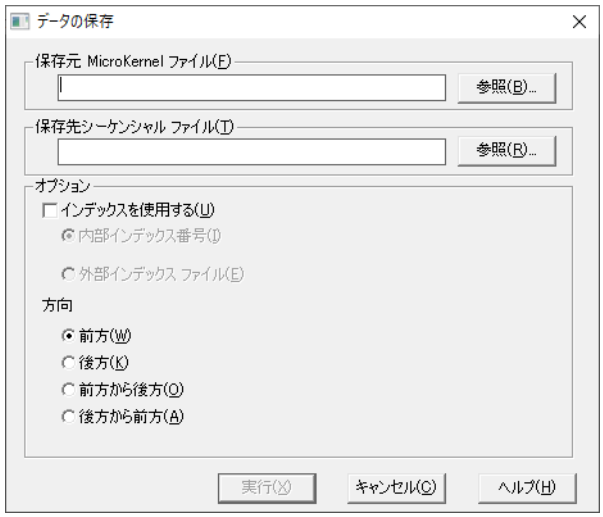
ASCII ファイルへのレコードのエスクポート

Maintenance ツールでは、データ ファイルから ASCII ファイルにレコードをエクスポートすることができます。

▶▶ ASCII レコードをエクスポートするには

1 メイン メニューで[**データ**]>[保存]をクリックします。[**データの保存**]ダイアログ ボックスが表示されます。

図 31 ［データの保存］ダイアログ ボックス



2 ［**データの保存**］ダイアログ ボックスで、以下のオプションを設定します。

保存元 MicroKernel ファイル	保存する既存 MicroKernel 互換ファイルの名前を指定します。
保存先シーケンシャル ファイル	作成するシーケンシャル ファイルの名前を指定します。
インデックスを使用する	エクスポートするレコードをソートする際に、指定したインデックスを使用します。デフォルトでは、Maintenance ツールでインデックスは使用されず、レコードはデータ ファイル内の物理位置に従ってエクスポートされます。
	内部インデックス番号： 指定したキー番号を使用します。［**保存元 MicroKernel ファイル**］ボックスのファイルを変更した場合は、［**インデックス リストをリフレッシュ**］をクリックして使用可能なインデックスを更新します。
	外部インデックス ファイル： 指定した外部インデックスを使用します（外部インデックスの作成は、「インデックスの作成」を参照）。
方向	前方：これがデフォルトの設定で、データ回復がファイルの先頭から末尾に向かって行われます。 後方：データ回復がファイルの末尾から行われます。 前方から後方：エラーが発生するまで、ファイルを前方から読み取ります。次に、エラーが発生したレコード、または別のエラーが発生したレコードまで、ファイルを末尾から読み取ります。 後方から前方：エラーが発生するまで、ファイルを後方から読み取ります。次に、エラーが発生したレコード、または別のエラーが発生したレコードまで、ファイルを先頭から読み取ります。

3 ［**実行**］をクリックしてデータをエクスポートします。Maintenance ツールによって、「ASCII ファイル形式のインポートとエクスポート」の形式に従って、指定された ASCII ファイルが作成されます。その後、この ASCII ファイルを編集し、［**ロード**］コマンドを使用して、編集したテキストを別の標準データ ファイルにインポートできます。

データ ファイル間のレコードのコピー

Maintenance ツールでは、MicroKernel データ ファイルから別の MicroKernel データ ファイルにデータをコピーすることができます。指定するデータ ファイルの両方のレコード長は、同じでなければなりません。

▶▶ MicroKernel データ ファイル間でレコードをコピーするには

1 メイン メニューで［**データ**］>［コピー］をクリックします。［**データのコピー**］ダイアログ ボックスが表示されます。

図 32 ［データのコピー］ダイアログ ボックス

2 コピーするファイルの名前を［**コピー元 MicroKernel ファイル**］ボックスに入力し、ファイルのコピー先のパスを［**コピー先 MicroKernel ファイル**］ボックスに入力します。

指定するデータ ファイルの両方のレコード長は、同じでなければなりません。

データ ファイルへの変更の回復（ロール フォワード）

『*Advanced Operations Guide*』の「ログ、バックアップおよび復元」を参照してください。

Btrieve の Maintenance コマンド ライン ツール（butil）

butil はコマンド ライン形式のインターフェイスを使用したい場合、または Continuous オペレーションを開始または停止したい場合に使用します。この実行可能プログラムは、butil.exe（Windows の場合）および butil（Unix ベースのシステムの場合）です。コマンド プロンプトで butil コマンドを実行する、または、実行可能なバッチ スクリプトとして実行することができます。butil コマンドを実行する前に、ここで説明する当コマンドの概念や構文を理解してください。

Btrieve の Maintenance コマンド ライン ツールでは、以下のファイル処理およびデータ処理を行います。

- 「データのインポートとエクスポート」
- 「データ ファイルの作成と変更」
- 「ファイルのページ キャッシュの管理」
- 「データ ファイル情報の表示」
- 「MicroKernel エンジンのバージョンの表示」
- 「MicroKernel エンジンとリクエスターのアンロード（DOS のみ）」
- 「Continuous オペレーションの実行」
- 「アーカイブ ロギングの実行」

リターン コード

butil は実行を完了すると、終了コードまたは DOS エラー レベルのリターン コードをオペレーティング システムに返します。リターン コードの種類は次の表に示すとおりです。

コード	説明
SUCCESS_E = 0	要求された処理は正常に終了しました。
PARTIAL_E = 1	要求された処理は完了しましたがエラーが発生しました。
INCOMPLETE_E = 2	要求された処理は正常に終了しませんでした。
USAGE_E = 3	入力の構文エラー。使用方法を画面に表示し、終了します。

コマンド

次の表は、butil で使用できるコマンドの一覧です。表内のリンクから、詳細な情報へ移動できます。

コマンド	説明
「Cache」	ファイルのページをキャッシュへ事前ロードします。ただし、ファイルが完全にキャッシュされているか、またはキャッシュがいっぱいの場合にはロードされません（コマンド プロンプトに戻ります）。Purge コマンドと対になるコマンドです。
「Clone」	既存ファイルの仕様を使って、新しい空のデータ ファイルを作成します。
「Close」	［ファイルを閉じるまでの待ち時間］設定オプションによって開いたままになっているファイルに対して、この設定を上書きする要求を送信します。
「Clrowner」	データ ファイルのオーナー ネームをクリアします。
「Copy」	データ ファイルの内容を、別のデータ ファイルにコピーします。
「Create」	データ ファイルを作成します。
「Drop」	インデックスを削除します。

コマンド	説明
Endbu	バックアップを対象として定義されたファイルの Continuous オペレーションを停止します。
「Index」	外部インデックス ファイルを作成します。
「Load」	データ ファイルにシーケンシャル ファイルの内容を読み込みます。
「Purge」	ファイルの不要なキャッシュ ページをすべて消去します。ファイルがオープン ハンドルを持つ場合は消去されません（ただちにコマンド プロンプトに戻ります）。cache コマンドと対になるコマンドです。
「Recover」	データ ファイルから連続的にデータを読み取り、シーケンシャル ファイルに結果を書き込みます。（ただし DOS バージョンでは、rollfwd は使用できません）。破損したファイルがある場合は、このコマンドを使用します。
Rollfwd	最後のバックアップからシステム エラーが発生する間に行った、データ ファイルへの変更を回復します。「アーカイブ ロギングの実行」を参照してください。
「Save」	キー パスのデータを読み取り、シーケンシャル ファイルに結果を書き込みます。
「Setowner」	データ ファイルにオーナー ネームを割り当てます。
「Sindex」	インデックスを作成します。
Startbu	バックアップを対象として定義されたファイルの Continuous オペレーションを開始します。『*Advanced Operations Guide*』の「ログ、バックアップおよび復元」を参照してください。
「Stat」	ファイル属性の情報およびデータ ファイルの現在のサイズを表示します。
「Stop」（DOS のみ）	MicroKernel エンジンとリクエスターをアンロードします。
「Ver」	サーバーにロードされているデータベース エンジンおよびリクエスターのバージョンを表示します。

コマンド構文の表示

各コマンドの使用方法を表示するには、ファイル サーバー上のプロンプトで butil コマンドを入力します。

コマンド形式

butil の形式は以下のとおりです。

`butil [-command [parameter ...]] | @commandFile`

−command	copy などの Maintenance ツールのコマンドが入ります。コマンドの前にはハイフン（-）を付け、ハイフンの前にはスペースを入力します。
parameter	コマンドに必要な情報です。詳細は各コマンドに応じて説明します。
@commandFile	コマンド ファイルのフル パス名

コマンド ファイル

コマンド ファイルは、以下の処理に使用します。

- コマンド ラインに収まらない、長いコマンドの実行
- 頻繁に使用するコマンドの実行（コマンド ファイルに一度入力し、その後はそのコマンド ファイルを実行）
- 以下のコマンド形式を使用した、コマンドの実行とその出力のファイルへの書き込み
  `butil @commandFile [commandOutputFile]`

コマンドを実行すると、結果の出力ファイルには、実行したコマンドとその結果が表示されます。すべてのメッセージは、サーバーのコンソール画面にも表示されます。

- 複数コマンドの連続実行

コマンド ファイルには、コマンド ラインで必要な情報と同じものが含まれます。

コマンド ファイルの規則

Maintenance ツールのコマンド ファイルを作成する際、以下の規則に従ってください。

- 1 つのパラメーターを 2 行に分割しない。
- 各コマンドは、<end> または [end] で終わる必要があります。複数のコマンドを実行する場合は、各コマンドが <end> で終わっている必要があります。<end> または [end] には、小文字を使用します。

コマンド ファイルの例

以下は、コマンド ファイルの例 copycrs.cmd です。このファイルでは、butil -clone コマンドを呼び出し、course.mkd ファイルを複製することによって newcrs.mkd ファイルを作成します。次に -create コマンドを呼び出し、newfiles.des ディスクリプション ファイルのディスクリプションを使用して newfile.dta ファイルを作成します。

```
-clone newcrs.mkd course.mkd <end>
-create newfile.dta newfiles.des <end>
```

以下のコマンドでは、copypats.cmd ファイルを使用して copypats.out ファイルに書き込みを行います。

```
butil @copypats.cmd copypats.out
```

ディスクリプション ファイル

ディスクリプション ファイルは、データ ファイルとインデックスの作成に使用するファイルのディスクリプションおよびキー仕様を含む、ASCII ファイルです。作成したファイルの情報を保存するための媒体としてディスクリプション ファイルを使用することもできます。ディスクリプション ファイル形式の詳細については、『*Advanced Operations Guide*』の「ディスクリプション ファイル」を参照してください。

拡張ファイルのサポート

データベース エンジンの データ ファイルは、オペレーティング システムのファイル サイズ上限を超えた容量を持つことが可能です。データを、MicroKernel の拡張ファイルからシーケンシャル ファイルにエクスポートした際、実際の形式の違いから、そのシーケンシャル ファイルの容量がデータベース エンジンのファイル サイズ上限を超える場合があります。

大きなサイズのファイルがエクスポートされる際、対話型 Maintenance ツールでは、シーケンシャル ファイルがオペレーティング システムのファイル サイズの上限（2 GB）を超えていることを検出すると、エクステンション ファイルの作成が開始されます。この処理は自動的に行われます。エクステンション ファイルおよび元のシーケンシャル ファイルは、同じボリューム内に存在する必要があります。エクステンション ファイルの名前には、ベース ファイルと似た名前を付ける方式が使用されます。エクステンション ファイルを示すのに、ネイティブな MicroKernel エンジン エクステンション ファイルがキャレット（"^"）を使用するのに対し、シーケンシャル ファイルのエクステンション ファイルはチルダ（"~"）を使用するので、同じベース ファイル名を持つ既存の MicroKernel エンジン拡張ファイルを上書きするのを防ぎます。最初のエクスポート エクステンション ファイルには、ベース ファイル名に ".~01" という拡張子が付きます。次のエクステンション ファイルには、ベース ファイル名に ".~02" というように拡張子が付けられます。これらの拡張子は 16 進形式で追加されます。

名前付け規則は 255 までのエクステンション ファイルをサポートしますが、現在のエクステンション ファイルの最大数は 64 です。したがって、最大ファイル サイズは 128 GB です。

シーケンシャル ファイルに対する、サイズの大きなファイルの 「Save」または 「Recover」については、それぞれのコマンドを参照してください。また、シーケンシャル ファイルからデータをインポートする場合は、そのファイルが拡張されているかどうかがチェックされ、エクステンション ファイルからデータがロードされます。

オーナー ネーム

MicroKernel を使用すると、ファイルのオーナー ネームを指定することにより、個々のファイルへのアクセスを制限することができます。オーナー ネーム文字列は暗号化にも使用されます。

エラー メッセージのリダイレクト

エラー メッセージをリダイレクトする場合は、フル パス名(ドライブ文字または UNC パスを含む)を入力します。

⏩ エラー メッセージをファイルへリダイレクトするには

- 以下のコマンド形式を使用します。

  `butil -`*command commandParameters* `>` *filePath*

ASCII ファイル形式

「対話型 Maintenance ツール」セクションの、「ASCII ファイル形式のインポートとエクスポート」を参照してください。

異なるプラットフォームでのファイル名指定の規則

Windows、Linux、または macOS で butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。

データのインポートとエクスポート

このトピックでは、butil コマンドの 「Copy」、「Load」、「Recover」、および 「Save」を使用したデータのインポートおよびエクスポートの詳細について説明します。

表 71　データのインポートとエクスポートを行うコマンド

コマンド	説明
「Copy」	データ ファイルの内容を、別のデータ ファイルにコピーします。
「Load」	データ ファイルにシーケンシャル ファイルの内容を読み込みます。
「Recover」	データ ファイルからシーケンシャルにデータを読み取り、シーケンシャル ファイルに結果を書き込みます。
「Save」	キー パスのデータを読み取り、シーケンシャル ファイルに結果を書き込みます。

メモ　SQL ステートメントによって取得したデータをエクスポートする方法については、『*Zen User's Guide*』のデータ エクスポート ユーティリティ 「deu」を参照してください。

Copy

copy コマンドは、MicroKernel ファイルの内容を、別のファイルにコピーします。copy は、入力データ ファイルの各レコードを取得して、出力データ ファイルに挿入します。両ファイルのレコードのサイズは同じである必要があります。レコードのコピー後、copy によって新しいデータ ファイルに挿入されたレコードの総数が表示されます。

メモ　Copy は、「Recover」コマンドと 「Load」コマンドに相当する処理を一度に実行します。

copy コマンドを使用することにより、古いファイルからのデータと新しいキー属性を持つデータ ファイルを作成できます。

▶▶ MicroKernel データ ファイルをコピーするには

1　「Create」コマンドを使用し、任意のキー属性（キー位置、キー長、重複キー値）を持つ空のデータ ファイルを作成します。

　または

　「Clone」コマンドを使用して、既存ファイルの属性を使った空のデータ ファイルを作成します。

2　copy コマンドを使用し、既存データ ファイルの内容を、新しく作成したデータ ファイルにコピーします。

形式

```
butil -copy sourceFile outputFile [/O<owner1 | /PROMPT>
    [/O<owner2 | /PROMPT>]] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	レコードをコピーする元となるデータ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
outputFile	レコードを挿入する先のデータ ファイルのフル パス名。出力データ ファイルにはデータが入っていても空でもかまいません。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
/O*owner1*	ソース データ ファイルのオーナー ネーム（必要な場合）。出力データ ファイルのみでオーナー ネームが必要な場合は、/O に続けて *owner1* にスペースを指定します（使用例を参照してください）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/O*owner2*	出力データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

例

以下のコマンドは、course.mkd のレコードを newcrs.mkd にコピーします。course.mkd 入力ファイルに、オーナーネームは必要ありませんが、newcrs.mkd 出力ファイルでは、オーナー ネーム Pam が使用されています。

```
butil -copy course.mkd newcrs.mkd /O /OPam
```

この例から最初の /O を省略した場合、オーナー ネーム Pam は、出力データ ファイルではなく、入力データ ファイルのものとして認識されます。

ファイルをコピーするときに、元のファイルとコピーされるファイルの両方のオーナーネームが必要な場合、-copy オプションでは次の例に示すように、両方のオーナーネームを指定します。

```
butil -copy originalFile copiedFile /Od3ltagamm@ /OV3rs10nXIII
```

最初のオーナーネーム d3ltagamm@ は originalFile を開くために必要です。2 番目のオーナーネーム V3rs10nXIII は copiedFile の作成に使用します。

オーナーネームが対話形式で指定される場合、このコマンドは次に示す例のようになります。

```
butil -copy originalFile copiedFile /PROMPT /PROMPT
```

これを実行すると、最初にユーザーは originalFile にアクセスするためのオーナーネームの入力を求められます。そのファイルが開いたら、次に copiedFile に割り当てるオーナーネームの入力が求められます。

Load

load コマンドは、入力 ASCII ファイルのレコードをファイルに挿入します。入力 ASCII ファイルは、1 つのファイルまたは拡張ファイル（ベース ファイルと複数のエクステンション ファイル）のどちらでもかまいません。load コマンドでは、入力 ASCII ファイルのデータは変換されません。データ ファイルへのレコード移動後、ロードされたレコードの総数が表示されます。

メモ load コマンドは、アクセラレイティド モードで出力ファイルを開きます。ロード オペレーション中は、ファイルのログは作成されません。アーカイブ ログを使用している場合は、load コマンド使用後にデータ ファイルを再度バックアップします。

拡張ファイル：次のエクステンション ファイルが検出された場合は、ロードが継続されます。save コマンドおよび recover コマンドで作成してあるエクステンション ファイルは、削除しないでください。ファイルに 3 つのエクステンション ファイルが存在し、ユーザーが 2 つ目のエクステンション ファイルを削除した場合は、1 つ目のエクステンション ファイル処理後に、レコードの**ロード**が中止されます。

save または recover コマンドにより作成されたエクステンション ファイルは 3 つなのに、4 つ目のエクステンション ファイルが以前の save または recover コマンドで作成されて存在している場合、4 つ目のエクステンション ファイルからもレコードが読み取られ、**データベース エンジン** ファイルに挿入されます。4 つ目のエクステンション ファイルが存在する場合は、load 処理の開始前に削除してください。

load コマンド実行前には、入力 ASCII ファイルおよびデータ ファイルを作成する必要があります。入力 ASCII ファイルは、標準のテキスト エディターやアプリケーションを使用して作成できますが、そのファイル形式（「ASCII ファイル形式のインポートとエクスポート」を参照）に従う必要があります。データ ファイルは、「Create」コマンドまたは 「Clone」コマンドを使用して作成できます。

形式

```
butil -load unformattedFile outputFile [/O<owner> | /PROMPT] [/UIDuname /PWDpword [/DBdbname]]
```

unformattedFile	データ ファイルにロードするレコードを含む ASCII ファイルのフル パス名。Windows プラットフォームでは、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
outputFile	レコードを挿入する先のデータ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
/O*owner*	データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

例

以下の例では、シーケンシャル レコードを course.txt から course.mkd に読み込みます。course.mkd のオーナー ネームは Sandy です。

```
butil -load course.txt course.mkd /OSandy
```

Recover

recover コマンドは、MicroKernel ファイルからデータを取り出し、「Load」コマンドで使用する入力 ASCII ファイルと同じ形式の ASCII ファイルに書き込みます。これは、破損した MicroKernel ファイルから、一部または全部のデータを取り出すときに便利です。recover コマンドは、すべてではないにしても、ファイルのレコードの

多数を取り出すことができます。この後は、load コマンドを使用し、新しい MicroKernel ファイルに回復したレコードを挿入します。

メモ Maintenance ツールでは、レコードのデータは変換されないため、テキスト エディターを使用してバイナリ データを含む出力ファイルを編集する場合、テキスト エディターによってはバイナリ データが変更されてしまい、予期しない問題が発生する場合があります。

形式

```
butil -recover sourceFile unformattedFile [/O<owner> | /PROMPT] [/Q] [/J] [/I]
    [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	回復するデータを含むデータ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
unformattedFile	回復したレコードを保存する ASCII ファイルのフル パス名。
/O*owner*	データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/Q	既存のシーケンシャル ファイルに上書きするかどうかを指定します。デフォルトでは、既存のファイルに上書きされます。このオプション使用時に同名のファイルが存在する場合は、エラー メッセージが表示されます。 また、回復の対象となるデータベース エンジン ファイルが拡張されているかどうかがチェックされます。ファイルが拡張されている場合は、存在する可能性のあるエクステンション ファイルと同名のファイルが存在するかどうかもチェックされます。このいずれかのファイルが存在する場合は、エラー メッセージが表示されます。
/J	ファイルを後方から読み取ります。このオプションを使用した場合は、step last と step previous オペレーションを使用してデータベース エンジン ファイルのデータ回復が行われます。デフォルトでは、step first と step next オペレーションを使用してファイルの先頭からデータが読み取られます。
/I	ファイルを前方から読み取ります。デフォルトでは、前方から読み取られますが、このオプションを使用して前方および後方を指定することができます。/I および /J の両方を指定した場合は、エラーが発生するまで前方からファイルが読み取られます。次に、発生したエラーまで、または別のエラーが発生するまで、ファイルを最後から読み取ります。 /J を最初に指定した場合は、まず後方から、次に前方から読み取ります。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

ソース ファイルの各レコードでは、recover コマンドで可変ページ エラー（ステータス コード 54）が発生した場合、現在のレコードから取得できるすべてのデータをシーケンシャル ファイルに書き込み、回復処理を継続します。

ツールでは、以下のメッセージが表示されます。

- 最後に作成されたエクステンション ファイルの名前に関する情報が表示されます。
- 次のエクステンション ファイルの存在をチェックし、存在する場合、削除が指示されます。

- 拡張シーケンシャル ファイルを移動した場合、ベース ファイルとその全エクステンション ファイルの移動が指示されます。

例

以下のステートメントでは、course.mkd からレコードを取り出し、course.txt に書き込みます。

```
butil -recover course.mkd course.txt
```

Save

save コマンドは、指定したインデックス パスを使用して MicroKernel ファイルからデータを読み込み、load コマンドで使用する形式に対応した ASCII ファイルに書き込みます。その後、その ASCII ファイルを編集し、load コマンドを使用して編集後のデータを、別のデータ ファイルに保存することができます。ASCII ファイル形式の詳細については、「ASCII ファイル形式のインポートとエクスポート」を参照してください。

save は、入力データ ファイルの各レコードに対して、出力 ASCII ファイルにレコードを 1 つ作成します。完了後、save によって保存されたレコードの総数が表示されます。

メモ Maintenance ツールでは、レコードのデータは変換されないため、テキスト エディターを使用してバイナリ データを含む出力ファイルを編集する場合、テキスト エディターによってはバイナリ データが変更されてしまい、予期しない問題が発生する場合があります。

形式

```
butil -save sourceFile unformattedFile [Y indexFile | N <keyNumber | -1>] [/O<owner1 | /PROMPT>
    [/O<owner2 | /PROMPT>]] [/Q] [/J] [/I] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	保存するレコードを含むデータ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
unformattedFile	レコードを保存する ASCII ファイルのフル パス名。
indexFile	デフォルトの設定である最小キー番号を使用してレコードを保存しない場合、レコードを保存する外部インデックス ファイルのフル パス名。
keyNumber	デフォルトの設定である最小キー番号を使用してレコードを保存しない場合、レコードを保存するキー番号（0 以外）。
-1	Btrieve Step オペレーションを使用して、物理的順序に従ってレコードを保存する場合に指定します。
/O*owner1*	ソース ファイルのオーナー ネーム（必要な場合）。インデックス ファイルのみでオーナー ネームが必要な場合は、/O に続けて *owner1* にスペースを指定します。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/O*owner2*	インデックス ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/Q	既存のシーケンシャル ファイルに上書きするかどうかを指定します。デフォルトでは、既存のファイルに上書きされます。このオプション使用時に同名のファイルが存在する場合は、エラー メッセージが表示されます。 また、回復の対象となるデータベース エンジン ファイルが拡張されているかどうかがチェックされます。ファイルが拡張されている場合は、存在する可能性のあるエクステンション ファイルと同名のファイルが存在するかどうかもチェックされます。このいずれかのファイルが存在する場合は、エラー メッセージが表示されます。

/J	ファイルを後方から読み取ります。このオプションを使用した場合は、get last と get previous オペレーションを使用してデータベース エンジン ファイルのデータ回復が行われます。デフォルトでは、get first と get next オペレーションを使用してファイルの保存が行われます。
/I	ファイルを前方から読み取ります。デフォルトでは、前方から読み取られますが、このオプションを使用して前方および後方を指定することができます。/I および /J の両方を指定した場合は、エラーが発生するまで前方からファイルが読み取られます。次に、発生したエラーまで、または別のエラーが発生するまで、ファイルを最後から読み取ります。 /J を最初に指定した場合は、まず後方から、次に前方から読み取ります。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

ツールでは、以下のメッセージが表示されます。

- 最後に作成されたエクステンション ファイルの名前に関する情報が表示されます。
- 次のエクステンション ファイルの存在をチェックし、存在する場合、削除が指示されます。
- 拡張シーケンシャル ファイルを移動した場合、ベース ファイルとその全エクステンション ファイルの移動が指示されます。

例

以下の 2 つの例は、save コマンドを使用した、データ ファイルからのレコード取得方法です。

この例では、newcrs.idx 外部インデックス ファイルを使用して course.mkd からレコードを取得し、それらを course.txt という名前のシーケンシャル テキスト ファイルに保存します。

```
butil save course.mkd course.txt newcrs.idx
```

この例では、キー番号 3 を使用して course.mkd からレコードを取得し、それらを course.txt という名前のシーケンシャル テキスト ファイルに保存します。

```
butil -save course.mkd course.txt n 3
```

データ ファイルの作成と変更

このトピックでは、次の表に示す各種 butil コマンドを使用したデータ ファイルの作成、変更、および管理に関する詳細について説明します。また、Btrieve データ ファイルの未使用スペースの削除に関して「Btrieve データ ファイルのコンパクト化」で説明します。

表 72　データ ファイルの作成と変更を行うコマンド

コマンド	説明
「Clone」	既存ファイルの仕様を使って、新しい空のデータ ファイルを作成します。
「Close」	［ファイルを閉じるまでの待ち時間］設定より優先されます。
「Clrowner」	データ ファイルのオーナー ネームをクリアします。
「Create」	データ ファイルを作成します。
「Drop」	インデックスを削除します。
「Index」	外部インデックス ファイルを作成します。
「Setowner」	データ ファイルにオーナー ネームを割り当てます。
「Sindex」	インデックスを作成します。

注意　同じディレクトリに、ファイル名が同一で拡張子のみが異なるようなファイルを置かないでください。たとえば、同じディレクトリ内のデータ ファイルの 1 つに Invoice.btr、もう 1 つに Invoice.mkd という名前を付けてはいけません。このような制限が設けられているのは、データベース エンジンがさまざまな機能でファイル名のみを使用し、ファイルの拡張子を無視するためです。ファイルの識別にはファイル名のみが使用されるため、ファイルの拡張子だけが異なるファイルは、データベース エンジンでは同一のものであると認識されます。

Clone

clone コマンドは、既存ファイルと同じファイル仕様（オーナー ネームを含まず、補足インデックスを含む）を持つ、空の新規ファイルを作成します。新しいデータ ファイルには、既存ファイルのすべての定義済みキー属性（キー位置、キー長、重複キー値など）が含まれます。

clone コマンドでは、ファイル情報（「システム データ」を参照）に影響する MicroKernel 設定オプションのすべてが無視されます（ファイル バージョンを**除く**）。clone コマンドは、［**作成ファイルのバージョン**］オプションで指定したデータベース エンジンのファイル バージョンを使用して新規ファイルを作成します。

形式

```
butil -clone outputFile sourceFile [/O<owner> | /PROMPT] [/pagecompresson | /
   pagecompressoff] [/recordcompresson | /recordcompressoff] [/UIDuname /PWDpword [/
   DBdbname]] [/S]
```

outputFile　空の新規データ ファイルに使用するフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。

sourceFile　複製する既存データ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。

/O*owner*	ソース データ ファイルにオーナー ネームが割り当てられている場合は、そのオーナー ネーム。ソース ファイルのオーナー ネームは出力ファイルには複製されないので注意してください。その出力ファイルにオーナー ネームが必要な場合は、個別に追加しなければなりません。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/pagecompresson	データベース エンジンのファイル互換性プロパティにある［作成ファイルのバージョン］が 9.5 または 13.0 の場合は、*outputFile* のページ圧縮をオンにします。
/pagecompressoff	*outputFile* のページ圧縮をオフにします。このパラメーターは、*sourceFile* にページ圧縮が使用されていない場合は無効です。
/recordcompresson	*outputFile* のページ圧縮をオンにします。
/recordcompressoff	*outputFile* のレコード圧縮をオフにします。このパラメーターは、*sourceFile* にレコード圧縮が指定されていない場合は無効です。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

備考

Btrieve 6.0 以降では、ページ サイズが 1024 バイトのデータ ファイルで最大 23 のキー セグメントを使用することができるため、clone コマンドでは、既存データ ファイルに 24 のキー セグメントが含まれ、ページ サイズが 1024 バイトの場合、新規データ ファイルのページ サイズは 2048 バイトに設定されます。これは、既存データ ファイルがバージョン 6.0 より前の形式で、データベース エンジン ロードの際に、［**作成ファイルのバージョン**］オプションが 5.*x* または 6.*x* に設定されていない場合に発生します。

バージョン 7.*x* 以前のファイルを複製する場合は、作成する新規ファイルのバージョン設定を確認してください。たとえば、6.15 ファイルを 9.5 形式で複製する場合は、データベース エンジンのファイル互換性プロパティで［作成ファイルのバージョン］オプションを 9.5 に設定します。

メモ ソース ファイルが 8.x、9.5、または 13.0 形式でシステム データを含まない場合は、データベース エンジンの設定にかかわらず、出力ファイルにもシステム データは含まれません。既存ファイルへのシステム データの追加については、『*Getting Started with Zen*』を参照してください。

ステータス コード 30（指定されたファイルは MicroKernel ファイルではありません）が発生し、ソース ファイルのヘッダー ページが破損している可能性がある場合、ディスクリプション ファイルで 「Create」コマンドを使用して新規の MicroKernel ファイルを作成します。

例

以下のコマンドは、course.mkd ファイルを複製して newcrs.mkd ファイルを作成します。

```
butil -clone newcrs.mkd course.mkd
```

Close

close コマンドは、ファイルの最後のクライアント接続が閉じた後もファイルを開いたままにしておくための［ファイルを閉じるまでの待ち時間］設定を上書きする要求をデータベース エンジンに送信します。その後、これらの開いているファイルは直ちに閉じられます。このコマンドは、すべてのファイルに対して、または単一の

ファイル、ファイルのリストに対して実行することができます。ファイル パラメーターを指定しないで close を実行した場合は、閉じるときに遅延待ちするすべての開いているファイルに適用されます。サーバー パラメーターを指定しないで実行した場合は、localhost と見なされます。

形式

```
butil -close [sourceFile | @listFile] [/Sserver]
```

sourceFile	閉じる単一ファイルのフル パス名。Windows プラットフォームで butil を使用する際、ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
@listFile	閉じるファイルのフル パス名を含む、テキスト ファイルのフル パス名。これらのパスは新しい行で区切り、各パスを個別の行に置きます。リスト ファイルでエラーが発生した場合、butil はリスト内のファイルの処理を停止します。
/S*server*	データベース エンジンをホストするリモート サーバーの名前、または IP アドレス。このパラメーターが指定されていない場合は、localhost と見なされます。

このコマンドを使用すれば、ファイルのコピーなど特定の操作を実行する前に、ファイルが確実に閉じられているようにすることができます。詳細については、パフォーマンス チューニング プロパティの［ファイルを閉じるまでの待ち時間］を参照してください。

Clrowner

clrowner コマンドは、MicroKernel ファイルのオーナー ネームをクリアします。

形式

```
butil -clrowner sourceFile </O<owner | /PROMPT> [/UIDname /PWDword [/DBname]]
```

sourceFile	データ ファイルの完全なフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
/O*owner*	クリアするオーナー ネーム。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/UID*name*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名。
/PWD*word*	/UID*name* で識別されるユーザーのパスワード。/UID*name* が指定された場合、/PWD*word* は必ず指定する必要があります。
/DB*name*	セキュリティが設定されたデータベース名。省略した場合はデフォルトのデータベースと解釈されます。

例

以下のコマンドは、tuition.mkd のオーナー ネームをクリアします。オーナー ネームは Sandy です。

```
butil -clrowner tuition.mkd /OSandy
```

Create

create コマンドは、ディスクリプション ファイルで指定されている属性を使用して、空の MicroKernel ファイルを作成します。create コマンドを実行する前に、新しいキー属性を指定するディスクリプション ファイルを作成する必要があります。詳細については、『*Advanced Operations Guide*』の「ディスクリプション ファイル」を参照してください。

形式

```
butil -create outputFile descriptionFile [< Y | N >] [/UIDuname /PWDpword [/DBdbname]]
```

outputFile	作成するデータベース エンジン ファイルのフル パス名。ファイル名が既存の MicroKernel ファイルと同名の場合、既存ファイルの代わりに空の新規ファイルが作成されます。既存ファイルに保存されているデータは消去され、回復することができません。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
descriptionFile	新規 MicroKernel ファイルの仕様を含むディスクリプション ファイルのフル パス名。
Y | N	既存のファイルに上書きするかどうかを指定します。このオプションに N を指定し、同名の MicroKernel ファイルが存在する場合は、エラー メッセージが表示されます。デフォルト設定は、Y です。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

例

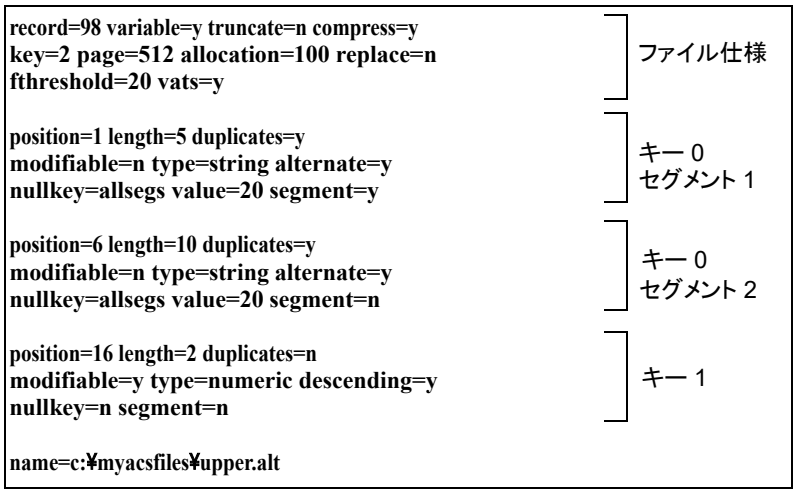
以下のコマンドは、create.des ディスクリプション ファイルで定義された内容を使用し、ファイル course.mkd を作成します。

```
butil -create course.mkd create.des
```

create コマンドで使用するディスクリプション ファイルの例

図 33 の例では、MicroKernel 形式のファイルが作成されます。ファイルのページ サイズは 512 バイト、キーは 2 つに設定されています。ファイルの各レコードの固定長部は 98 バイト に設定されています。このファイルでは、可変長レコードがブランク トランケーション、レコード圧縮、可変長部割り当てテーブル（VAT）を使用しないように指定されています。空きスペース スレッショルドは、20 パーセントに設定されています。プリアロケーションは、100 ページに設定されています。ファイル作成時に、100 ページつまり 51,200 バイトがプリアロケートされます。

図 33 create コマンドで使用するディスクリプション ファイルの例



キー 0 は、重複可能で変更不可能な文字列セグメントを 2 つ含むセグメント キーで、この 2 つのセグメントには、16 進数で 20（スペース）のヌル値が指定されています。キー 0 では、照合順序 upper.alt が使用されます。

キー 1 は、重複不可能で変更可能な数値型のセグメント化されていないキーです。これは、降順でソートされます。

Drop

drop コマンドは、ファイルからインデックスを削除し、それ以降のキー番号から 1 を引くことによって、残りのインデックスのキー番号を調整します。キーの番号を調整しない場合は、削除に指定したキー番号に 128 を加算します。この調整は、6.0 以降のファイルでのみ使用可能です。

形式

```
butil -drop sourceFile <keyNumber | SYSKEY | SYSKEY2> [/Oowner | /PROMPT] [/UIDname /PWDword
    [/DBname]]
```

sourceFile
インデックスを削除するファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。

keyNumber
削除するキー番号。元のキー番号を維持する場合は、指定したキー番号に 128 というバイアス値を足します。

SYSKEY
システム定義のログ キー（システム データ）を削除します。システム定義のログ キーを削除した場合でも、レコードから値は消去されず、新しく挿入されたレコードには、重複しないシステム定義のログ キー値が割り当てられます。

しかし、ユーザー定義の重複しないキーが存在しない限り、システム定義のログ キーが削除されたファイルのログは実行されません。この理由から、システム定義のログ キーが破損している可能性があり、キーを追加し直す場合にのみ、このオプションを使用してください。

「Sindex」コマンドを使用すると、削除したシステム定義のログ キーを再利用できます。

SYSKEY2
システム データ v2 用の 2 番目のシステム キー（124）を削除します。このキーを削除してもレコードからシステム データは削除されず、新しいレコードおよび更新されたレコードの値は引き続き保持されます。

ただし、インデックスがない場合、最近変更されたレコードを効率的に見つけることはできません。この理由から、キーが破損している可能性があり、キーを追加し直す場合にのみ、このオプションを使用してください。Sindex コマンドを使用すると、削除したシステム キーを再利用できます。

/O*owner*
ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。

/UID*name*
セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名。

/PWD*word*	/UID*name* で識別されるユーザーのパスワード。/UID*name* が使用された場合、/PWD*word* は必ず指定する必要があります。
/DB*name*	セキュリティが設定されたデータベース名。省略した場合はデフォルトのデータベースと解釈されます。

例

以下の例では、course.mkd に 3 つのキーが存在します。ファイルの元のキーは、それぞれの番号が 0、1、2 となっています。

最初の例では、butil -drop コマンドで、course.mkd からキー番号 1 を削除し、残りのキー番号をそれぞれ 0 と 1 に調整します。

```
butil -drop course.mkd 1
```

以下の例では、butil –drop コマンドでキー番号 1 を削除し、キー番号の調整は行いません。キー番号は、それぞれ 0 と 2 のままになります。

```
butil -drop course.mkd 129
```

Index

index コマンドは、既存ファイルでキーとして指定されていないフィールドに基づき、既存 MicroKernel ファイルの外部インデックス ファイルを構築します。index コマンドを実行する前に、新しいキー属性を指定するディスクリプション ファイルを作成します。ディスクリプション ファイルの詳細については、『*Advanced Operations Guide*』の「ディスクリプション ファイル」を参照してください。

新規ファイルのレコードは、以下から構成されます。

- 既存データ ファイルに含まれる各レコードの 4 バイトのアドレス
- ソートの基準として使用する新しいキー値

メモ ディスクリプション ファイルで指定したキー長が 10 バイトの場合、外部インデックス ファイルのレコード長は 14 バイト（10 + 4 バイト アドレス）になります。

形式

```
butil -index sourceFile indexFile descriptionFile [/O<owner> | /PROMPT] [/UIDname /PWDword [/
    DBname]]
```

sourceFile	外部インデックスを構築する既存ファイルの完全なフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
indexFile	外部インデックスを保存するインデックス ファイルのフル パス名。
descriptionFile	新しいキー定義を含む、作成したディスクリプション ファイルのフル パス。ディスクリプション ファイルには、新しいキーの各セグメントに対する定義が含まれている必要があります。
/O*owner*	データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/UID*name*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。

/PWD*word*	/UID*name* で識別されるユーザーのパスワード。/UID*name* が使用された場合、/PWD*word* は必ず指定する必要があります。
/DB*name*	セキュリティが設定されたデータベース名。省略した場合はデフォルトのデータベースと解釈されます。

備考

index コマンドは、外部インデックス ファイルを作成し、インデックスが設定されたレコードの数を表示します。外部インデックス ファイルを使用して、データ ファイルのレコードを読み込む場合は、「**Save**」コマンドを使用します。

index コマンドで使用するディスクリプション ファイルの例

以下のディスクリプション ファイルでは、1 つのセグメントを含む新しいキーを定義します。キーは、レコードの 30 バイト目から始まり、10 バイトの長さを持ちます。また、重複と変更が可能な STRING 型で、オルタネート コレーティング シーケンスを使用しません。

図 34 INDEX コマンドで使用するディスクリプション ファイルの例

```
position=30 length=10 duplicates=y modifiable=y
type=string alternate=n segment=n
```

例

以下のコマンドは、データ ファイル course.mkd を使用し、外部インデックス ファイル newcrs.idx を作成します。course.mkd ファイルには、オーナー ネームは必要ありません。新しいキーの定義を含むディスクリプション ファイルは、newcrs.des という名前です。

```
butil -index course.mkd newcrs.idx newcrs.des
```

Setowner

setowner コマンドは、データ ファイルのオーナー ネームを設定します。

形式

```
butil -setowner sourceFile /O<owner | /PROMPT> level [/L] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	データ ファイルのフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
/O*owner*	設定するオーナー ネーム。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
level	データ ファイルのアクセス制限の種類。以下は、このパラメーターの説明です。 *level* は /O パラメーターの直後に置く必要があります。 0：すべてのアクセス モードでオーナー ネームが必要（データ暗号化なし） 1：リード オンリー アクセスにはオーナー ネームは必要なし（データ暗号化なし） 2：すべてのアクセス モードでオーナー ネームが必要（データ暗号化あり） 3：リード オンリー アクセスにはオーナー ネームは必要なし（データ暗号化あり）

/L	長いオーナー ネームを指定します。 オーナー ネームでは大文字と小文字が区別されます。また、短いものと長いものがあります。短いオーナー ネームは半角 8 文字までの範囲で指定できます。長いオーナー ネームの長さは、ファイル形式によって異なります。長いオーナー ネームに関する制限事項については、『*Btrieve API Guide*』の 「Set Owner（29）」の「手順」トピックを参照してください。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

例

以下の例では、course.mkd データ ファイルの短いネームのオーナーを作成します。オーナー ネーム Sandy に制限レベル 1 を設定します。

```
butil -setowner course.mkd /OSandy 1
```

以下の例では、billing.mkd データ ファイル用の長いオーナー ネームを作成し、そのオーナー ネームとファイルを暗号化し、すべてのアクセス モードを制限します。

```
butil -setowner billing.mkd /Ohr#Admin$945k7YY%svr 2 /L
```

Sindex

sindex コマンドは、既存 MicroKernel ファイルに追加インデックスを作成します。デフォルトでは、新規インデックスのキー番号は、データ ファイルでそれまで最大だったキー番号より 1 つ大きな番号になりますが、キー番号を指定することも可能です。ただし、drop コマンドでインデックスを削除し、残りキー番号の調整を行わなかった場合は未使用のキー番号が存在するため、例外が発生します。この場合は、新しいインデックスに最初の未使用番号が割り当てられます。

キー番号オプションを使用することにより、新規インデックスのキー番号を指定できます。指定するキー番号は、ファイルで使用されていない有効なキー番号でなければなりません。無効なキー番号を指定した場合は、ステータス コード 6 が返されます。

このコマンドで SYSKEY または SYSKEY2 オプションを使用しない場合は、sindex コマンドを使用する前に、インデックスのキー仕様を定義するディスクリプション ファイルを作成する必要があります。詳細については、『*Advanced Operations Guide*』の「ディスクリプション ファイル」を参照してください。

形式

```
butil -sindex sourceFile <descriptionFile | SYSKEY | SYSKEY2> [keyNumber] [/O<owner> | /PROMPT]
    [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	外部インデックスを構築する既存ファイルの完全なフル パス名。Windows プラットフォームで butil を使用する際、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
descriptionFile	新しいキー定義を含む、作成したディスクリプション ファイルのフル パス。ディスクリプション ファイルには、新しいキーの各セグメントに対する定義が含まれている必要があります。
SYSKEY	システム キー（125）が削除されたファイルに、システム キーを戻します。
SYSKEY2	システム データ v2 用のシステム キー（124）を戻します。

/O*owner*	データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

例

以下の例では、course.mkd にインデックスを追加します。ディスクリプション ファイルの名前は、newidx.des です。

```
butil -sindex course.mkd newidx.des
```

以下の例では、システム定義キーが削除された course.mkd にシステム定義キーを追加します。

```
butil -sindex course.mkd syskey
```

Btrieve データ ファイルのコンパクト化

butil では、複数のコマンド（「Clone」、「Recover」、「Load」）を使用してデータ ファイルの未使用スペースを削除し、ファイル サイズを縮小することができます。

▶▶ Btrieve データ ファイルをコンパクト化するには

1. データ ファイルの名前を変更し、「Clone」オプションを使用して元のファイルと同名の空のデータ ファイルを作成します。
2. 「Recover」を使用し、複製したファイルのデータを、シーケンシャル テキスト ファイルに保存します。
3. 「Load」を使用し、回復されたデータを複製ファイルにロードします。

   空のレコード以外のデータを含むすべてのレコードは、新しく作成したデータ ファイルにロードされます。この処理は、対話型 Maintenance ツールを使用して実行することもできます。

ファイルのページ キャッシュの管理

パフォーマンスを向上させるために、butil で cache コマンドと purge コマンドを使用してファイルのページ キャッシュを管理することができます。

注記

butil -cache および butil -purge コマンドがクライアント キャッシュ エンジンやレポート エンジンから実行される場合、そのコマンド アクションが適用されるのはローカル キャッシュにあるファイルのみです。

ファイルがクライアント キャッシュに残るようにするには、以下のいずれかの状況になっていなければなりません。

- ファイルが閉じている間にそのファイルに書き込みを行うマシンがほかにない
- 最低 1 つのアプリケーションがクライアント マシンでファイルを開いておく必要がある

クライアント キャッシュ エンジンがアプリケーションとしてインストールされている場合は、シャットダウンしてクライアントがすべてのファイルを閉じた直後にキャッシュを空にします。これは、サービスとしてインストールされているクライアント キャッシュ エンジンやレポート エンジンの場合には問題になりません。

Cache

cache コマンドは、ファイルのページをキャッシュへ事前ロードします。ただし、ファイルが完全にキャッシュされているか、またはキャッシュがいっぱいの場合にはロードされません（コマンド プロンプに戻ります）。

形式

```
butil -cache <sourceFile | @listFile>
```

sourceFile	キャッシュに事前ロードするデータ ファイルのフル パス名。Windows プラットフォームでは、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
@listFile	キャッシュに事前ロードするファイルのフル パス名を含む、テキスト ファイルのフル パス名。これらのパスは新しい行で区切り、各パスを個別の行に置きます。リスト ファイルでエラーが発生した場合、butil はリスト内のファイルの処理を停止します。

Purge

purge コマンドは、ファイルの不要なキャッシュ ページをすべて消去します。ファイルがオープン ハンドルを持つ場合は消去されません（ただちにコマンド プロンプトに戻ります）。

形式

```
butil -purge <sourceFile | @listFile>
```

sourceFile	キャッシュから消去するデータ ファイルのフル パス名。Windows プラットフォームでは、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
@listFile	キャッシュから消去するファイルのフル パス名を含む、テキスト ファイルのフル パス名。これらのパスは新しい行で区切り、各パスを個別の行に置きます。リスト ファイルでエラーが発生した場合、butil はリスト内のファイルの処理を停止します。

データ ファイル情報の表示

このトピックでは、stat を使用してデータ ファイルの特性や統計情報に関するレポートを生成する方法を説明します。

Stat

stat コマンドは、定義されているデータ ファイルの特性、およびファイルの内容に関する情報を含むレポートを作成します。stat コマンドは、Create Index オペレーションと Create オペレーションによってファイルに定義されたキーの区別を付けません。

形式

```
butil -stat <sourceFile> [/O<owner> | /PROMPT] [/UIDuname /PWDpword [/DBdbname] /JSON]]
```

sourceFile	情報を表示するデータ ファイルのフル パス名。Windows プラットフォームでは、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
/O*owner*	データ ファイルのオーナー ネーム（必要な場合）。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。
/PROMPT	ユーザーがオーナー ネームを入力するための対話型プロンプトが提示されることを示します。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。
/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。
/JSON	JSON 形式で出力を返します。

ファイル統計情報とエラー メッセージの形式

butil -stat ステートメントは、ファイル統計情報とエラー メッセージの出力として 2 つの形式を提供します。デフォルトはテキスト出力ですが、/json オプションを使用すれば同じ情報が JSON 形式で返されます。

次の例は、これらのオプションを示しています。

- 「テキスト形式のファイル統計情報」
- 「JSON 形式のファイル統計情報」
- 「テキスト形式のエラー メッセージ」
- 「JSON 形式のエラー メッセージ」

テキスト形式のファイル統計情報

次の例は、patients.dta ファイルの情報をテキストとして表示します。

```
butil -stat D:¥ClinicDB¥patients.dta
```

以下の情報が返ります。

```
************************************************
Btrieve Maintenance Utility 15.00.034.000
Copyright (C) Actian Corporation 2020
All Rights Reserved.
```

```
File Statistics for D:¥ClinicDB¥patients.dta

File Version = 8.00
Owner Name Protection = Not Present
Encryption Level = None
Page Size = 2048
Page Preallocation = No
Key Only = No
Extended = No

Total Number of Records = 16
Record Length = 104
Record Compression = No
Variable Records = No

Available Linked Duplicate Keys = 0
Balanced Key = No
Log Key = 1
 System Data = No
Total Number of Keys = 3
Total Number of Segments = 4

Key  Segment Position Length  Type  Flags Null Values*  Unique  ACS Values
0    1       21       20      String MD   --            16       0
0    2        7       12      String MD   --            16       0
1    1        1        6      String M    --            16       0
2    1       83       10      String MD   --            7        0

   Alternate Collating Sequence (ACS) List:
 0 UPPER

Legend:
 < = Descending Order
 D = Duplicates Allowed
 I = Case Insensitive
 M = Modifiable
 R = Repeat Duplicate
 A = Any Segment (Manual)
 L = All Segments (Null)
 * = The values in this column are hexadecimal.
?? = Unknown
-- = Not Specified
```

この例では、patients.dta のファイル バージョンが 8.0 であることを表しており、これがファイル形式を読むことができる Btrieve の一番古いバージョンであることを示します。ファイルには、2048 バイトのページ サイズが指定されており、プリアロケート ページはありません。これは、キー オンリー ファイルでも拡張ファイルでもありません。

このファイルには 16 件のレコードが挿入されています。また、104 バイトのレコード長が定義されており、レコード圧縮は使用せず、可変長レコードも許可されていません。

このファイルには、利用可能なリンク重複キーはありません。インデックス バランスも使用しません。キー 1 を使用したログが実行され、ファイルには、システム定義のデータも存在しません。また、4 つのキー セグメントから構成される 3 つのキーが存在します。

メモ 「Sindex」で作成したインデックスには、「予約重複ポインター」要素を指定しない限り、デフォルトで R が割り当てられます。

統計情報レポートには、特定のキーに関する情報も表示されます。たとえば、キー 0 は 2 つのセグメントから構成され、重複可能、変更可能であることがレポートに表示されます。このほか、以下も表示されます。

- 最初のセグメントはポジション 21 から始まり、長さが 20 バイト、重複可能、変更可能、そして string 型として保存されます。ハイフンは、ヌル値が定義されていないことを表します。Unique Values の列は、16 個の重複しない値がセグメントに挿入されていることを表しています。このセグメントでは、オルタネート コレーティング シーケンス ファイル upper.alt が使用されます。
- 2 つ目のセグメントはポジション 7 から始まり、長さが 12 バイト、重複可能、変更可能、そして string 型として保存されます。16 個の重複しない値が、このセグメントに挿入されています。このセグメントでは、オルタネート コレーティング シーケンス ファイル upper.alt が使用されます。

キー 1 は、このファイルのログに使用されるキーです。これは 1 つのセグメントで構成されます。ポジション 1 から始まり、長さが 6 バイト、重複不可、変更可能、そして string 型として保存されます。16 個の重複しないキー値が、このキーに挿入されています。このキーでは、オルタネート コレーティング シーケンス ファイル upper.alt が使用されます。

キー 2 は、1 つのセグメントで構成されます。ポジション 83 から始まり、長さが 10 バイト、重複可能、変更可能、そして string 型として保存されます。7 個の重複しないキー値が、このキーに挿入されています。このキーでは、オルタネート コレーティング シーケンス ファイル upper.alt が使用されます。

JSON 形式のファイル統計情報

次の例は、patients.dta ファイルの情報を JSON として表示します。

```
butil -stat D:¥ClinicDB¥patients.dta /json
```

「テキスト形式のファイル統計情報」に示されている、デフォルトの -stat オプションと同じ内容が返ります。

```
{
   "description":"Btrieve Maintenance Utility",
   "title":"Btrieve JSON output",
   "version":"15.00.034.000",
   "copyright":"Copyright (C) Actian Corporation 2020.All Rights Reserved.",,
   "file_statistics_for":"D:¥¥ClinicDB¥¥patients.dta",
   "file_version":"8.00",
   "Owner_Name_Protection":"Not Present",
   "Encryption_Level":"None",
   "Page_Size":2048,
   "Page_Preallocation":"No",
   "Key_Only":"No",
   "Extended":"No",
   "Total_Number_of_Records":16,
   "Record_Compression":104,
   "Page_Compression":"No",
   "Variable_Records":"No",
   "Available_Linked_Duplicate_Keys":0,
   "Balanced_Key": "No",
   "Log_Key": "1",
   "System_Data": "No",
   "Total_Number_of_Keys": 3,
   "Total_Number_of_Segments": 4,
   "Keys" :
   [
      {"key":"0", "segments" :
    [
      {"segment":1, "position":21, "length":20, "type":"String ", "flag":"   MD",
    "null_values":"--", "unique_values":16, "acs":"0"},
```

```
     {"segment":2, "position":7, "length":12, "type":"String ", "flag":"   MD", "null_values":"-
    -", "unique_values":16, "acs":"0"}
       ]
       },
       {"key":"1", "segments" :
         [
       {"segment":1, "position":1, "length":6, "type":"String ", "flag":"   M ", "null_values":"-
    -", "unique_values":16, "acs":"0"}
         ]
       },
       {"key":"2", "segments" :
         [
         {"segment":1, "position":83, "length":10, "type":"String ", "flag":"    MD",
    "null_values":"--", "unique_values":7, "acs":"0"}
         ]
       }
   ],
   "Alternate_Collating_Sequence(ACS)_List":
   [
      {"id": 0, "acs_name": "UPPER    " }
   ],
   "Legend" :
   [
      "< = Descending Order",
      "D = Duplicates Allowed",
      "I = Case Insensitive",
      "M = Modifiable",
      "R = Repeat Duplicate",
      "A = Any Segment(Manual)",
      "L = All Segments(Null)",
      "* = The values in this column are hexadecimal.",
      "? ? = Unknown",
      "-- = Not Specified"
   ],
   "execution_status":"コマンドが完了しました。"
}
```


テキスト形式のエラー メッセージ

次の例は、エラー メッセージをテキストとして表示します。

```
butil -stat D:¥ClinicDB¥wrongdir¥patients.dta
```

以下のエラーが返ります。

```
Btrieve Maintenance Utility 15.00.034.000
Copyright (C) Actian Corporation 2020
All Rights Reserved.
BUTIL-14：エラーが発生したファイルは D:¥ClinicDB¥wrongdir¥patients.dta です。
BUTIL-100：MicroKernel エラー = 35。アプリケーションでディレクトリ エラーが発生しました。
BUTIL-9：修復不能なエラーのため、コマンドを完了できませんでした。
```

JSON 形式のエラー メッセージ

次の例は、エラー メッセージを JSON として表示します。

```
butil -stat D:¥ClinicDB¥wrongdir¥patients.dta /json
```

「テキスト形式のエラー メッセージ」に示されている、デフォルトの -stat オプションと同じ内容が返ります。


```
{
"description":"Btrieve Maintenance Utility",
    "title":"Btrieve JSON output",
    "version":"15.00.034.000",
    "copyright":"Copyright (C) Actian Corporation 2020.All Rights Reserved.",
    "error": [
```

```
        "BUTIL-14: エラーが発生したファイルは D:¥ClinicDB¥wrongdir¥patients.dta です。",
        "BUTIL-100: MicroKernel エラー = 35。アプリケーションでディレクトリ エラーが発生しました。"
       ],
    "execution_status":"BUTIL-9: 修復不能なエラーのため、コマンドを完了できませんでした。"
}
```

システム データ v2

ファイルにシステム データ v2 が含まれている場合は、次の 2 つの追加エントリが出力に含まれます。

```
System Data v2 = Yes
SYSKEY v2 Status = Present
```

ファイル バージョンに関する注意

ファイル形式のバージョンを求められたら、データベース エンジンは指定したファイルを読むことのできる最も古いエンジンのバージョンを指定します。たとえば、5.*x* 形式で作成されたファイルがある場合でも、4.*x* または 5.*x* の機能を使用していないのであれば、レポートにはバージョン 3.*x* ファイルとして出力されます。6.*x* 形式で始めるには、ファイル自体にバージョン スタンプを含める必要があります。6.*x* より前のバージョンの場合、ファイル形式のバージョンを調べる唯一の方法は、ファイルで使用されている機能を調べることでした。これは、ファイル バージョンごとに利用できる機能に違いがあり、新しいファイル バージョンにはそれ以前のバージョンでは利用できない機能があるからです。

バージョン 5.*x* およびそれ以前のファイルについては、使用されている機能と、それによって特定できるファイル形式のバージョンを次の表に示します。

出力されるファイル バージョン	以下の機能の 1 つ以上が使用されている場合
5.*x*	• 圧縮レコード • キー オンリー ファイル
4.*x*	• 拡張キー タイプ • 可変長レコード • Create Index オペレーションで追加されたキー
3.*x*	上記のいずれも使用していない

MicroKernel エンジンのバージョンの表示

このセクションでは、ver コマンドを使用した、MicroKernel エンジンのバージョンの表示に関して説明します。

Ver

ver コマンドは、MicroKernel エンジンおよびリクエスター（アクセス モジュール）の両方のバージョン番号を表示します。

形式

```
butil -ver
```

備考

ver コマンドを実行した場合、以下のようなメッセージが表示されます。

```
Btrieve リクエスターのバージョンは 14.00 です。
Btrieve のバージョンは 14.00 で、xxx 版です。
```

MicroKernel エンジンとリクエスターのアンロード（DOS のみ）

Stop

stop コマンドは、MicroKernel エンジンおよびリクエスターをアンロードします。

形式

```
butil -stop
```

Continuous オペレーションの実行

Continuous オペレーションに関連するコマンド startbu および endbu については、『*Advanced Operations Guide*』の「ログ、バックアップおよび復元」で説明しています。

アーカイブ ロギングの実行

Maintenance ツール（GUI またはコマンド ラインの butil）は、アーカイブ ログ ファイルをデータ ファイルにロール フォワードする手段を提供します。『*Advanced Operations Guide*』の「ログ、バックアップおよび復元」も参照してください。

butil **rollfwd** コマンドは、最後のバックアップからシステム エラーが発生する間に行った、データ ファイルへの変更を回復します。システム エラーが発生した場合は、データ ファイルのバックアップ コピーを復元した後、butil rollfwd コマンドを使用し、ログに保存されているすべての変更を復元したデータ ファイルに適用します。バックアップからデータ ファイルを復元しない限り、このコマンドを使用しないでください。

メモ rollfwd コマンドを利用するには、MicroKernel の［選択ファイルのアーカイブ ロギング］オプションを有効にし、システム エラー発生*前*にファイルのバックアップを行う必要があります。

また、rollfwd コマンドを使用し、ログ オペレーションの出力ファイルを作成することも可能です。rollfwd コマンドでは、ロール フォワード前、またはロール フォワードと同時に出力ファイルを作成できます。

単一のファイルをロール フォワードすることもできますし、ボリューム上のすべてのデータ ファイル、ドライブ上のすべてのデータ ファイル、もしくは、ファイル、ボリュームおよび / またはドライブのリストをロール フォワードすることもできます。

GUI の使用

1 オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から、あるいは Zen Control Center の［**ツール**］メニューから Maintenance ユーティリティにアクセスします。

2 ［Maintenance］ウィンドウで、［**データ**］>［**ロール フォワード**］を選択します。［**ロール フォワード**］ダイアログ ボックスが表示されます。

図 35 ［ロール フォワード］ダイアログ

3 オペレーション タイプとして、" 単一ファイル "、" ファイルのリスト "、" ボリューム名 "、" ドライブ レター " のいずれかを選択します。" ボリューム名 " または " ドライブ レター " を選択する場合は、名前の最後に円記号（**¥**）またはスラッシュ（/）を挿入する必要があります（例、**¥¥server¥vol1¥** または **D:¥** など）。

4 ロール フォワード タスクの実行に必要なすべての Btrieve オペレーションのリストである、ダンプ ファイルと呼ばれるログ ファイルを作成することができます。

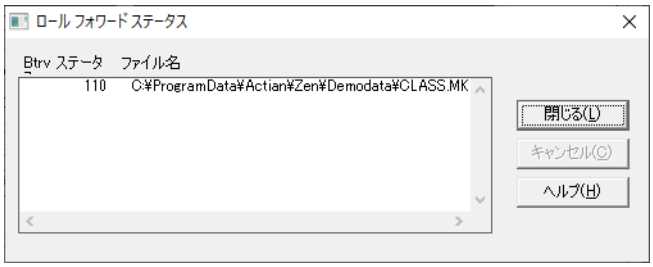
デフォルトでは、このファイルは作成されません。ファイルを作成する場合は、[**ダンプ ファイルを作成する**] チェック ボックスをオンにします。以下のオプションも設定可能です。

表 73　ロール フォワード GUI オプション

ダンプ ファイルの作成のみ	ダンプ ファイルのみが作成され、ロール フォワードは実行されません。
ダンプ ファイル名	これには、円記号 (¥) またはスラッシュで始まるダンプ ファイル名が含まれ、ドライブ文字、サーバー名またはボリューム名は含まれません。
データ バッファー長	各 Btrieve オペレーションで、ダンプ ファイルに書き込まれるデータ バッファーのバイト数を表します。
キー バッファー長	各 Btrieve オペレーションで、ダンプ ファイルに書き込まれるキー バッファーのバイト数を表します。
16 進で数値を表示	このオプションを選択した場合は、ダンプ ファイル出力の数が 16 進形式で表示されます。このオプションを選択しない場合、数は十進形式で表示されます。
詳細	ユーザー名、ネットワーク アドレス、タイムスタンプなど、ダンプ ファイルの補足情報が含まれます。

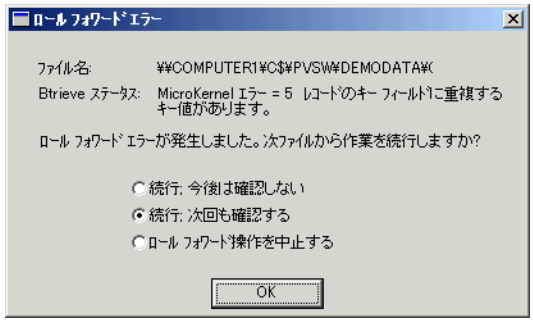
5　[**実行**] をクリックし、ダンプ ファイルの作成やロール フォワードを実行します。データが有効な場合は、[**ロール フォーワード ステータス**] ダイアログ ボックスが表示されます。

図 36　[ロール フォワード ステータス] ダイアログ ボックス



処理されたファイルは、リスト ボックスに追加され、ファイル名、およびロール フォワード オペレーションから返される Zen ステータス コードが表示されます。

処理中にエラーが発生した場合は、**エラー時のロール フォワード続行**を指定できるダイアログ ボックスが表示されます。このダイアログ ボックスでは、このダイアログを再表示することなく処理を継続するか、処理を継続して必要な場合にこのダイアログを再表示するか、またはファイル処理の中断を選択できます。



コマンド ラインの使用

このセクションでは、ロール フォワードをコマンド ラインで使用する構文を説明します。

```
butil -rollfwd <sourceFile | drive | @listFile>
    [</L[dumpFile] | /W[dumpFile]> [/T<dataLength>]
    [/E<keyLength>] [/H] [/V] [/O<ownerList | owner> | /PROMPT]]
    [/A] [/UID<name> </PWD<word>> [/DB<name>]]
```

sourceFile	変更をロール フォワードするデータ ファイルのフル パス名。Windows プラットフォームでは、データ ファイルが現在のディレクトリに存在する場合は、パスを指定する必要はありません。
drive	変更をロール フォワードするドライブ文字。ボリューム名の最後には、円記号（¥）またはスラッシュ（/）を使用します（例：F:¥、F:/）。
@listFile	変更をロール フォワードするファイルのパス、ボリューム、ドライブを含む、テキスト ファイルのフル パス名。これらのパスは、キャリッジ リターン / ライン フィードで区切ります。エラーが発生した場合は、現在のファイルのロール フォワードが中断されますが、その時点までの変更はロール バックされません。/A オプションを指定した場合は、次のファイルからロール フォワードが継続されます。
/L*dumpFile*	ロール フォワードを実行せず、出力ファイルを作成します。
/W*dumpFile*	ロール フォワードを実行し、出力ファイルを作成します。
dumpFile	ログ オペレーションが書き込まれる出力ファイルの名前。デフォルトは ¥blog¥broll.lst で、物理ドライブのルートに関連付けられます。ファイル名は、スラッシュ（/）または円記号（¥）で始まり、ドライブ文字やボリューム名は使用しません。ファイルは、blog.cfg と同じボリュームに配置されます。
/T*dataLength*	出力ファイルに書き込む、オペレーションのデータ バッファーの長さを指定します。このオプションを指定しない場合は、出力ファイルにデータ バッファーの内容が含まれません。
/E*keyLength*	出力ファイルに書き込む、オペレーションのキー バッファの長さを指定します。このオプションを指定しない場合は、出力ファイルにキー バッファーの内容が含まれません。
/H	出力ファイルの数値表示を 16 進数形式にします。このオプションを指定しない場合、出力ファイルの数値は ASCII 形式で表示されます。このオプションは、**エントリ数**、**オペレーション コード**、**キー番号**、**データ長**の各フィールドの形式に影響します。
/V	出力ファイルに、補足情報（ユーザー名、ネットワーク アドレス、タイム スタンプなど）が追加されます。
/O	データ ファイルのオーナー ネームを指定します（必要な場合）。ログに記録したオペレーションの出力ファイルをリクエストし、データ ファイルのバックアップ コピーにリード オンリー アクセス用のオーナー ネームが存在する場合に、オーナー ネームが必要になります。/PROMPT オプションを使用すると、実行時にオーナー ネームの対話型プロンプトが生成されます。 2 つ以上のファイルにオーナー ネームがある場合は、それぞれのオーナー ネームはカンマで区切る必要があります。
/A	複数ファイルのロール バック中にエラーが発生した場合、次のファイルからロール フォワードを継続します。 このオプションを指定しない場合は、エラー発生時にロール フォワードが中断されます。その時点までの変更はロール バックされません。 **メモ**：/A オプションを使用した場合は、出力をファイルにリダイレクトできます。「エラー メッセージのリダイレクト」および「コマンド ファイル」の説明を参照してください。
/UID<*name*> /UID*uname*	セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。

/PWD<*word*> /PWD*pword*	*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。
/DB<*name*> /DB*dbname*	セキュリティが設定されたデータベース名を指定します。省略した場合はデフォルトのデータベースと解釈されます。

メモ 実行する Btrieve オペレーションに対して、キー バッファーまたはデータ バッファーが入力パラメーターでない場合、ダンプ ファイルには何も書き込まれません。

例

例 A 次の例は、class.mkd ファイルに、デフォルトのアーカイブ ログおよびログ ロケーションから変更を回復します。

```
butil -rollfwd file_path¥Zen¥Demodata¥class.mkd
```

ファイルのデフォルトの保存場所については、『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

例 B この例では変更を回復し、それらを以下のオプションを使用して d:¥ ボリュームのすべてのファイルに出力します。

- デフォルトのダンプ ファイル使用
- 書き込むデータ バッファーの長さを 32 バイトに設定
- 書き込むキー バッファーの長さを 4 バイトに設定
- 16 進モードで書き込み

```
butil -rollfwd d:¥/W /H /T32 /E4
```

例 C 以下の例では、ロール フォワードは実行されず、以下のダンプ オプションに従って、files.txt にリストされているファイルへの変更が出力されます。

- d:¥temp¥files.lst をダンプ ファイルとして使用
- 詳細モードを使用
- データ ファイルにはオーナー ネーム own123 および own321 が含まれる
- データ バッファーやキー バッファーは出力しない

```
butil -rollfwd d:¥temp¥files.txt /L¥temp¥files.lst /V /Oown123,own321
```

データ ファイルの変換

16

データ ファイル互換性を維持する

Zen には、Btrieve ファイルを、Zen エンジンの最新バージョンで利用できるようにする変換するツールがあります。以下のトピックでは、ツールの概念と変換を行うための各種ツールについて説明します。

- 「Rebuild ツールの概念」
- 「Rebuild ツールの GUI のリファレンス」
- 「Rebuild ツールの使用」

Rebuild ツールの概念

Rebuild ツールを使用すると、MicroKernel データ ファイルと辞書ファイルに対して以下の操作を行うことができます。

- 旧ファイル形式を新しい Zen 形式に変換する
- 新しいファイル形式を 6.x 形式より古くない形式に変換する
- 同じファイル形式（6.x、7.x、8.x、9.5、または 13.0）を使用してファイルをリビルドする
- ファイルにインデックスを追加する
- ファイルのページ サイズを変更する
- ファイルをリビルドして、システム データとシステム キーを保持、追加、または削除する
- Rebuild で使用するログ ファイルの場所と名前を指定する

データベースで辞書ファイル（DDF）を使用している場合は、データ ファイルだけでなくこれらもリビルドする必要があります。

データ ファイルのリビルドの概念についてより理解を深めるため、以下のセクションをお読みください。

- 「サポートされるプラットフォーム」
- 「ファイル形式」
- 「コマンド ライン パラメーター」
- 「テンポラリ ファイル」
- 「Rebuild 処理の最適化」
- 「ログ ファイル」

Rebuild ツールの使用についての情報は、以下のトピックを参照してください。

- 「Rebuild ツールの GUI のリファレンス」
- 「Rebuild ツールの使用」
- 「CLI バージョン Rebuild 操作」

サポートされるプラットフォーム

Rebuild には 2 つの様式があります。1 つは Windows 用の 32 ビット GUI バージョンで、もう 1 つは Linux、macOS、Raspbian および Windows 用のコマンド ライン バージョンです。「Rebuild ツールの GUI のリファレンス」および「CLI バージョン Rebuild 操作」を参照してください。

Linux、macOS および Raspbian CLI Rebuild

Rebuild は、Linux、macOS および Raspbian では rbldcli というプログラムとして実行されます。デフォルトでは、このプログラムは /usr/local/actianzen/bin にあります。

Windows CLI Rebuild

Rebuild は Windows では rbldcli.exe というプログラムとして実行されます。これらのファイルはデフォルトで、Program Files ディレクトリにインストールされます。

ファイル形式

現在のデータベース エンジンはいくつかの古いデータ形式や辞書ファイル形式と互換性を保っていますが、最新の機能を利用するためにファイルを最新の形式に変換したい場合があります。次の表に、旧形式から新しい形式に変換する一般的な理由を挙げます。

表 74　Rebuild ツールによる変換

変換前のファイル形式	変換後のファイル形式	変換の理由
9.5	13.0	ファイルのサイズ（テラバイト単位）。システム データ v2 を追加できる。
9.0	9.5	120 個以上のセグメント キーと最大 256 GB のファイル サイズ。
8.*x*	9.x	最大 128 GB のファイル サイズのサポートの追加。
8.*x*	8.*x*	ファイルから削除レコードのスペースを除去する、ページ サイズを変更する、またはシステム データを追加する。
v8.*x* より前	8.*x*	挿入（Insert）、更新（Update）、削除（Delete）パフォーマンスの向上（Turbo Write Accelerator によっても実現）。
7.*x*	7.*x*	変換前のファイルには、システム キーがない。
v7.*x* より前	7.*x*	バージョン 7.*x* の機能の利用および全般的なパフォーマンスの向上。
v6.0 より前	6.*x*	バージョン 6.*x* の機能の利用および全般的なパフォーマンスの向上。バージョン 6..*x* のエンジンを実行している場合にのみ、このオプションを使用します。

コマンド ラインの Rebuild を使用してできたファイル形式は、**-f**　パラメーターによって異なります。**-f** パラメーターを指定しないと、Rebuild は MicroKernel エンジンの［作成ファイルのバージョン］設定の値を使用します。たとえば、［作成ファイルのバージョン］の値が 9.5 の場合、バージョン 9.0 のファイルに対して Rebuild ツールを実行すると、このファイルは バージョン 9.5 の形式に変換されます。

Rebuild を実行する前に、変換するデータ ファイルをすべてバックアップしておくことをお勧めします。これは、元のファイルと同じ場所にリビルドする場合には特に重要です（この場合リビルドされたファイルは元のファイルを置き換えます）。バックアップ コピーを取っておけば、必要な場合に元のファイルを復元することができます。バックアップを確実に実行するためには、以下のいずれかの操作を行います。

- バックアップを実行する前にすべてのデータ ファイルを閉じます。
- Continuous オペレーションを使用します（バックアップ中のみ）。

メモ　Continuous オペレーション モードになっているファイルで Rebuild を実行することはできません。

テンポラリ ファイル

Windows では、Rebuild は TMP システム環境変数で指定されたディレクトリにテンポラリ ファイルを作成します。デフォルトで、Linux、macOS および Raspbian では Rebuild は出力ディレクトリ（**-b** オプションが使用されていない場合はソース ディレクトリ）にテンポラリ ファイルを作成します。このため、テンポラリ ファイル ディレクトリには元のファイルと新しいファイルの両方を格納するために十分なディスク容量が必要です（Rebuild の実行中）。Rebuild GUI バージョンの［出力ディレクトリ］オプションを使用するか、CLI バージョンの **-b** オプションを使用して、これらのファイルを別のディレクトリに保存するように指定できます。

通常、変換が終了するとテンポラリ ファイルは自動的に削除されます。ただし、停電などの重大な障害が発生した場合は、テンポラリ ファイルが削除されないことがあります。このような場合は、次の表に示すタイプのテンポラリ ファイルを手動で削除する必要があります。

プラットフォーム	テンポラリ ファイル名
Linux、macOS、Raspbian	_rbldxxxxxx、ここで、xxxxxx はランダムな 6 個の文字です。 **注意**：Rebuild 実行モジュールである rbldcli を削除しないようにしてください。
Windows	_rbldx、x は 数字です。

Rebuild 処理の最適化

Rebuild はデータベース エンジンに Btrieve 呼び出しを行います。そのため、データベース エンジンの構成設定と処理メモリの容量がリビルド処理のパフォーマンスに影響を与えます。これは特に、サイズの大きなデータ ファイルをリビルドする際の所要時間に関しては顕著です。

一般的に、インデックスを構築するのはデータ ページを構築するよりはるかに時間がかかります。インデックスを多数持つデータ ファイルがある場合、同じファイルでインデックスが少ないものに比べ、ファイルを構築するのにより多くの時間を要します。

リビルドの処理時間には以下の項目が影響します。

- 「CPU 速度およびディスク速度」
- 「メモリ量」
- 「ソート バッファー サイズ」
- 「MicroKernel の最大メモリ使用量」
- 「キャッシュ割当サイズ」
- 「インデックス ページ サイズ」
- 「インデックス数」

CPU 速度およびディスク速度

CPU（中央処理装置）の速度および物理ストレージ ディスクへのアクセス速度が、リビルドの処理時間に影響します。一般的に、これらの両方の速度が速いほどリビルド処理も速く行われます。ディスク速度は、メモリに全体が入りきらない大きなファイルのリビルドではより重要になります。

ヒント　ギガバイト範囲のサイズが大きいファイルでは変換に数時間かかる場合があります。複数のデータベース エンジンが使用できる場合、リビルド処理をいくつかのマシンの CPU 間で分担したいと考えるでしょう。たとえば、すべてのファイルの中からいくつかずつを、データベース エンジンがインストールされている各マシンにコピーし、リビルド処理が終わったらファイルをコピーして元の場所に戻します。

メモリ量

Rebuild では、デフォルトの方法ともう 1 つの方法の 2 つの方法を用いてファイルをリビルドすることができます。「-m<0 | 2>」パラメーターを参照してください。選択する方法は、使用できるメモリの量によって決まります。デフォルトの方法（-m2）では、利用可能なメモリがあれば、Rebuild は以下の手順を行います。

1　元のファイルに定義されているのと同じレコード構造とインデックスを持つ空のデータ　ファイルを新規作成します。

2　新しいファイルからすべてのインデックスを削除します。

3　新しいファイルにインデックスなしですべてのデータをコピーします。

4 以下の処理でインデックスを追加します。

a. 元のファイルの特定のキーについて、Extended Step オペレーションを使用してメモリ バッファーにできるだけ多くのキー値を読み込みます。

b. メモリ バッファーの値をソートし、ソートされた値をテンポラリ ファイルに書き出します。

c. 手順 a および b を繰り返し、各レコードのキー値を処理します。

これで、テンポラリ ファイルにはいくつかのキー値が設定されました。それぞれのキー値は個別にソートされています。

5 データのセットをインデックス ページにマージし、各ページの容量いっぱいまで書き込みます。各インデックス ページはデータ ファイルの最後に追加され、ファイル長は拡大されます。

6 残りの各キーについて、手順 4 および 5 を繰り返します。

この処理中にテンポラリ ファイルのオープンや書き込みに失敗などのエラーが発生した場合、Rebuild はもう 1 つの方法を用いてファイルのビルドを最初からやり直します。

デフォルトの方法には存在するメモリが足りない場合やデフォルトの方法で処理中にエラーが発生した場合、Rebuild はもう 1 つの方法（-m0）を使用します。

1 元のファイルに定義されているのと同じレコード構造とインデックスを持つ空のデータ　ファイルを新規作成します。

2 新しいファイルからすべてのインデックスを削除します。

3 新しいファイルにインデックスなしですべてのデータをコピーします。

4 以下の順序でインデックスを追加します。

a. 元のファイルの特定のキーについて、Step Next オペレーションを使用して、1 度に 1 レコードを読みます。

b. レコードからキー値を抽出し、インデックスの適切な場所に挿入します。この処理では、キー　ページがいっぱいになると必然的にページの分割が行われます。

c. 手順 a および b を繰り返し、各レコードのキー値を処理します。

5 残りの各キーについて、手順 4 を繰り返します。

もう 1 つの方法は、デフォルトの方法に比べて一般的にはるかに時間がかかります。多数のインデックスを持つサイズの大きなファイルがある場合、2 つの方法の差は何時間から何日にまで及ぶことがあります。Rebuild が必ずデフォルトの方法を使用するようにする唯一の方法は、十分な**使用可能**メモリを確保することです。設定プロパティのいくつかは、利用可能なメモリ量に影響を与えます。

必要なメモリ量を見積もる式

以下の式は、速い方法でファイル インデックスをリビルドするのに必要な連続した空きメモリの最適かつ最小の量を見積もります。最適なメモリ量は、RAM 上のすべてのマージ ブロックを格納するのに十分な量です。最小のメモリ量は、RAM 上の 1 つのマージ ブロックを格納できる量です。

キー長 = ファイル上で最大のキーのすべてのセグメントの合計サイズ
キー オーバーヘッド = 8（キー タイプがリンク重複でない場合）、12（キー タイプがリンク重複の場合）。
レコード数 = ファイル内のレコード数

最適なメモリ バイト = (((キー長 + キー オーバーヘッド) * レコード数) + 65536) / 0.6

最小メモリ バイト = 最適メモリ バイト / 30

たとえば、ファイルに 8,000,000 レコードが含まれていて、最も長いキーが 20 バイト（リンク重複キーではない）である場合、理想的なメモリの量は 373.5 MB、つまり (((20 + 8) * 8,000,000) + 65536) / 0.6 = 373,442,560 バイトとなります。

最適な連続空きメモリの量は 373.5 MB です。少なくともこれだけの空きメモリがあれば、Rebuild 処理はすべて RAM 上で行われます。60 % という割り当て制限のため、メモリの最適な量というのは、実際はリビルド処理開

始時に必要とされる空き領域の量で、リビルド処理が実際に使用する量ではありません。この最適量に 0.6 をかけると、Rebuild が実際に使用する最大量が決定されます。

メモリの最低量は最適量の 1/30、12,448,086 バイト、または 12.45 MB です。

除数に 30 を使用するのはデータベース エンジンが一度に追跡するマージ ブロックは 30 以下であるためですが、常に 1 つのマージ ブロックがメモリ上にある必要があります。除数に 0.6 を使用するのは、エンジンがリビルド処理に 60% より多くの物理メモリを割り当てることはないためです。

使用可能メモリが最小の量より少ない場合、Rebuild はデータ ファイルのリビルドにもう 1 つの方法を使用します。

最後に、割り当てられたメモリブロックは 2 つの条件を満たす必要があります。必要なブロックと割り当てられたブロック サイズは次のようになります。

必要なブロック数は 30 以下です。

必要なブロック数 = Round Up（最適なメモリ バイト / 割り当てられたブロック）

割り当てられたブロック サイズは以下の値以上である必要があります。

((2 * 最大キー数 + 1) * (キー長 + キー オーバーヘッド)) * 必要なブロック数

ページ サイズ 512 バイト、12.45 MB のブロック割り当てに成功したと仮定すると、必要なブロック数は次のようになります。

必要なブロック数 = 373,500,000 / 12,450,000 = 30

最初の条件に合っています。

割り当てられたブロック サイズの値は次のようになります。

```
最大キー数 = (512-12) / 28 = 18
(((2 * 18) + 1) * (20 + 8)) * 9 = 9324
```

割り当てられたブロック（12,500,000 バイト）は 9324 バイトより大きいですか ? そうであれば 2 番目の条件に合っています。インデックス キーはテンポラリ ファイルに 12.45 MB ずつの塊として書き込まれ、メモリに格納され、次にインデックスに書き込まれます。

ソート バッファー サイズ

この設定は、ランタイムでインデックスを作成中に MicroKernel がソートのために動的に割り当てる、または割り当てを解除するメモリの最大容量を指定します。

設定がゼロ（デフォルト）の場合、Rebuild は最適なメモリ バイト値を計算し、その値に基づいてメモリを割り当てます。メモリ割り当てに成功したら、割り当てられたブロックのサイズは少なくとも最小メモリ バイトで定義された値になります。「必要なメモリ量を見積もる式」を参照してください。

設定がゼロ以外の値で、値が計算された最小メモリ バイトより小さい場合、Rebuild はその値を使用してメモリを割り当てます。

最後に、Rebuild は割り当てるべきメモリ量と実際に利用可能な量の 60% とを比較します。そして、その 2 つのうち小さい方を割り当てようとします。メモリ割り当てに失敗した場合、Rebuild は最後に割り当てようとしたメモリ量の 80% の割り当てを試みることを続けます。メモリ割り当てに完全に失敗した場合（これは最小メモリ バイトよりメモリ量が少ないことを意味します）、Rebuild はファイルのリビルドにもう 1 つの方法を使用します。

MicroKernel の最大メモリ使用量

この設定は、総物理メモリに対して MicroKernel が消費できるメモリの割合を指定します。MicroKernel による L1、L2、およびその他すべてのメモリの使用量が含まれます。リレーショナル エンジンによる使用量は含まれません。

リビルドするサイズの大きなファイルがある場合、［MicroKernel の最大メモリ使用量］のパーセンテージを一時的にデフォルトより小さい値に設定します。リビルド処理が終わったら、再度適切なパーセンテージに設定し直します。

キャッシュ割当サイズ

この設定は、MicroKernel によって割り当てられるレベル 1 キャッシュのサイズを指定します。MicroKernel では、すべてのデータ ファイルへのアクセスにこのキャッシュが使用されます。

この設定は、データベース エンジンがデータ ファイルにアクセスするのにどれだけのメモリが使用できるかを決定します。インデックスを構築する際に使用するものではありません。

キャッシュ割り当てサイズを大きな値に増やしても、インデックスの構築は速くなりません。実際、重要なメモリを占めることにより Rebuild にとっては利用不可能となり、処理を遅くする可能性があります。大きなファイルをリビルドする場合には、キャッシュ値を低い値に抑えます。たとえば、現在の値の 20% で、ただし 5 MB を下回らないようにします。こうすると、できるだけ多くのメモリをインデックスのリビルドに残すことができます。

インデックス ページ サイズ

ファイルのページ サイズも、インデックス構築の速度に影響します。Rebuild がもう 1 つの方法を使用した場合、小さいキー ページを使用するとインデックスの構築に必要な時間は劇的に増加します。Rebuild がデフォルトの方法を使用した場合には、キー ページ サイズはインデックスの構築にはあまり影響を与えません。

Rebuild は、ページ サイズをアプリケーションのパフォーマンスまたはディスク ストレージに対して最適化します。

データ アクセスの速度の観点から、パフォーマンス向けにページ サイズを最適化するために、Rebuild は 4096 バイトのデフォルトページ サイズを使用します。これにより、物理ストレージのページサイズが大きくなり、リビルドの時間は遅くなります。

ディスク ストレージに対するページ サイズの最適化に関する解説は、『*Zen Programmer's Guide*』の「ページ サイズの選択」を参照してください。

アプリケーションで、8,000,000 レコード、20 バイトのキーおよび 512 バイトのページ サイズを使用すると仮定します。MicroKernel は各インデックス ページに 8 個から 18 個の間のキー値を書き込みます。こうすると、各ページで必要となる物理ストレージ量が減ります。ただし、8,000,000 レコードにインデックスを付けると、およそ 7 階層の深さの B ツリーが作成され、ほとんどのキー ページが 7 番目のレベルに属します。パフォーマンスは低下します。

ページ サイズ 4096 バイトを使用すると、データベース エンジンは各インデックス ページに 72 から 145 個のキー値を書き込みます。B ツリーはおよそ 4 レベルの深さで済み、Rebuild が新しいキー値を挿入するときに調べるページ数が少なくなります。パフォーマンスは向上しますが、物理ストレージをより必要とします。

インデックス数

インデックス数もインデックス構築の速度に影響します。一般的に、インデックス数が多いほどビルド処理には時間がかかります。インデックスの構築に必要な時間は、B ツリーの深さの増加により幾何級数的に増加します。

ログ ファイル

リビルド処理からの情報はテキスト ログ ファイルに追加されます。ログ ファイルはデフォルトで現在の作業ディレクトリに保存されます。

CLI Rebuild を Windows、Linux、macOS、および Raspbian で実行した場合、デフォルトのファイル名は rbldcli.log です。デフォルトを使用せず、ログ ファイルの場所と名前を指定することもできます。「-lfile」パラメーターを参照してください。

ログはテキスト エディターで読み込むこともできます。記録される情報は、以下のとおりです。

- リビルド処理の開始時刻
- コマンド ラインで指定されたパラメーター
- ステータス コードおよびエラー説明（エラーが発生した場合）
- 処理中のファイル

- 処理に関する情報（ページ サイズの変更など）
- 処理済みレコード総数
- 再構築されたインデックス総数（-m2 処理方法が使用された場合）
- リビルド処理の終了時刻
- 処理のステータス（たとえば、ファイルのリビルドが成功したかどうかなど）

Rebuild ツールの GUI のリファレンス

このトピックでは、Rebuild ツールのグラフィカル ユーザー インターフェイスのオブジェクトについて説明します。

ファイル オプションの画面

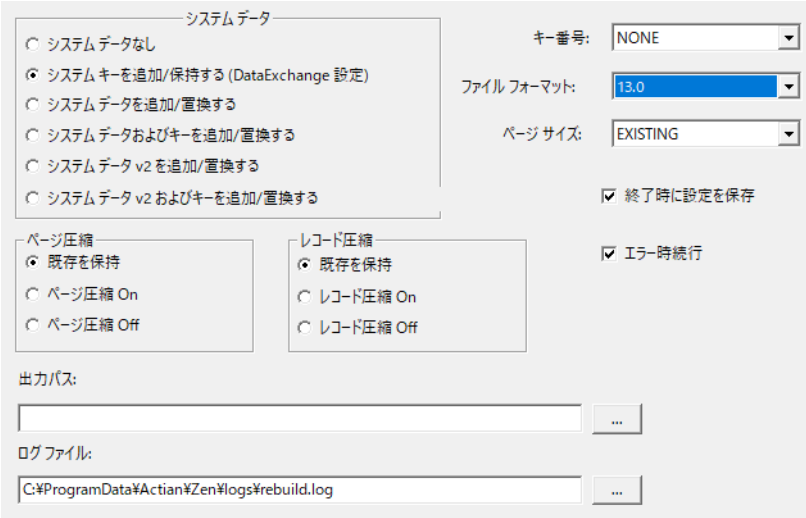
この画面を使用すると、リビルドのリストにファイルを追加することができます。

GUI のオブジェクト	説明	関連情報
選択ファイル	リビルドするためにリストされたデータ ファイルおよび辞書ファイル。[追加] ボタンを使用して選択したものです。	「ファイルをリビルドするには」
[追加] ボタン	リビルドするファイルのリストにデータ ファイルまたは辞書ファイルを追加します。	「ファイルをリビルドするには」
[削除] ボタン	選択したデータ ファイルまたは辞書ファイルをリストから削除します。	「ファイルをリビルドするには」
[クリア] ボタン	選択されたデータ ファイルおよび辞書ファイルのリスト全体をクリアします。	「ファイルをリビルドするには」

Rebuild オプションの画面

この画面を使用すると、ファイルのリビルドに関するオプションを選択することができます。画像上のそれぞれの領域をクリックするとその詳細が表示されます。

図 37 Rebuild ツールのファイル オプション



GUI のオブジェクト	説明	関連情報
システム データ	Rebuild でシステム データとキーを保持、追加、または削除するかどうかを指定します。 ユーザー定義のユニークなデータ キーが存在しない場合、システム データはトランザクション一貫性保守とトランザクション ログに使用されます。さらに、システム データ v2 はレコードの更新の追跡にも使用できます。	「ファイルをリビルドするには」
ページ圧縮	ファイルにページ圧縮を使用するかどうかを指定します。選択肢は、" 既存を保持 "、" 圧縮 On"、および " 圧縮 Off" です。" 既存を保持 " を選択すると、ファイルにページ圧縮が設定されている場合は、それを維持します。	ページ圧縮には 9.5 以上のファイル形式であることが必要です。 「レコードおよびページ圧縮」
レコード圧縮	ファイルにレコード圧縮を使用するかどうかを指定します。選択肢は、" 既存を保持 "、" 圧縮 On"、および " 圧縮 Off" です。" 既存を保持 " を選択すると、ファイルにレコード圧縮が設定されている場合は、それを維持します。	「レコードおよびページ圧縮」.
エラー時続行	リビルド処理中にエラーが発生した場合に、Rebuild の処理を続行するかどうかを決定します。このオプションを選択すると、エラーが発生した場合でも、ツールは次のファイルの処理を続行します。ツールにより、MicroKernel データ ファイル以外のファイルであることやその他のエラーが発生したことが通知されますが、データ ファイルのリビルドは続行されます。このオプションを選択しない場合、エラーが検出されたときにはリビルド処理が停止します。 このオプションは、リビルド ファイルにワイルドカード文字を指定した場合に効果的です。	「ファイルをリビルドするには」
終了時に設定を保存	このダイアログ ボックスの現在の設定を保存して、次回のリビルドのセッションでもこの設定を使用できるようにします。	「ファイルをリビルドするには」

GUI のオブジェクト	説明	関連情報
キー番号	ファイルのリビルド処理を行う際にツールが読み込むキーを指定します。このオプションで［NONE］を選択した場合、ファイルの複製、インデックスの削除、新しいファイルへのレコードのコピーが行われた後、インデックスがリビルドされます。この方法は、キー番号を指定するよりも短時間でリビルドでき、ファイルのサイズも小さくなるため、できる限りこの方法を使用するようにしてください。 この方法では、元のファイルとは異なる物理順序のレコードを含む新規ファイルが作成されることがあります。 キー番号を指定した場合には、インデックスの削除と置き換えは行われず、ファイルの複製とコピーが行われます。この方法は、［NONE］を指定するよりもリビルドに時間がかかりますが、インデックスをリビルドしたくない場合には、この方法を使用します。	• 「ファイルをリビルドするには」 • 『*Zen Programmer's Guide*』の「キー属性」
ファイル　フォーマット	以前の Rebuild ツールは、エンジンのファイル互換性プロパティにある［作成ファイルのバージョン］設定に基づくファイル バージョンを使用していました。 最新の Rebuild ツールを使用すると、その設定とは無関係にファイル バージョンを指定することができます。	「ファイルをリビルドするには」
ページ サイズ	新しいファイルのページ サイズをバイト単位で指定します。"EXISTING"、" 最適化（ディスク スペース）"、" 最適化（データ アクセス）" のいずれかを選択するか、サイズをバイト単位で指定します。"EXISTING" を選択した場合は現在のページ サイズが使用されます。何らかの理由で元のサイズが使用できない場合は、ツールにより、ほかのページ サイズに変更されます。 たとえば、ページ サイズが 1024 で、24 個のキーを持つ v5.*x* ファイルがあるとします。Btrieve v6.0 以降のバージョンでは、ページ サイズが 1024 の場合、23 個までのキーのみをサポートするため、ファイルには、ツールによって新しいページ サイズが自動的に割り当てられ、ログ ファイルに通知メッセージが記述されます。	• 「ファイルをリビルドするには」 • データ アクセスの最適化については、「Rebuild 処理の最適化」を参照してください。 • ディスク スペースの最適化については、『*Zen Programmer's Guide*』の「ページ サイズの選択」を参照してください。
出力パス	リビルドされたファイルの保存先となる別の場所を指定します（デフォルトの場所は現在のディレクトリです）。必ず既存のディレクトリを指定してください。 このオプションを使用すれば、サイズの大きいファイルを別のサーバーでリビルドすることができます。MicroKernel とその通信コンポーネントが、リビルドされたファイルを格納するサーバーでロードされている必要があります。パスには、ワイルドカード文字を**使用しないでください**。 出力ディレクトリの場所が元のファイルの場所と異なる場合、元のファイルはリビルド中に削除されません。出力ディレクトリが元のファイルと同じディレクトリである場合には、リビルドの完了時に元のファイルは削除されます。 DefaultDB w/ DB security: ［名前付きデータベースの管理］で指定したデータベースのファイルの場所以外ではリビルドできません。	「ファイルをリビルドするには」
ログ ファイル	リビルド ログ ファイルの場所を指定します（デフォルトの場所は現在の作業ディレクトリです）。パスには、ワイルドカード文字を**使用しないでください**。	• 「ファイルをリビルドするには」 • 「ログ ファイル」

Rebuild ツールの使用

以下のトピックでは、Rebuild ツールの GUI バージョンとコマンド ライン バージョンについて説明します。

- 「GUI バージョン Rebuild 操作」
- 「CLI バージョン Rebuild 操作」

GUI バージョン Rebuild 操作

- 「GUI Rebuild ウィザードを開始するには」
- 「ファイルをリビルドするには」

▶▶ GUI Rebuild ウィザードを開始するには

Zen Control Center のメニューから［**ツール**］>［Rebuild］を選択するか、オペレーティング システムの［**スタート**］メニューまたは**アプリ**画面から Rebuild にアクセスします。

▶▶ ファイルをリビルドするには

1 Rebuild の開始画面で［**次へ**］をクリックすると、**ファイルの選択**画面が表示されます。

2 ［**追加**］をクリックしてリビルドするデータ ファイルまたは辞書ファイルを選択します。複数のファイルを一度に選択することもできます。

図 38 ファイルの選択用ダイアログ ボックス

元のファイルと同じディレクトリでファイルをリビルドする場合、元のファイルはリビルド完了後に削除されます。新しいファイルを別のディレクトに置く場合は、元のファイルは削除されません。

3 目的のファイルを追加したら、［**次へ**］をクリックします。

4 リビルド オプションを指定します。「Rebuild オプションの画面」を参照してください。

5 ［**次へ**］をクリックしてリビルド処理を開始します。

Rebuild は処理に関する情報をレポートします。リビルド処理が完了すると、成功か失敗かが表示され［**ログ ファイルを表示する**］ボタンが有効になります。

6 結果を表示するには、［**ログ ファイルを表示する**］をクリックします。ログ ファイルの内容は、オペレーティング システムのデフォルトのテキスト エディターで表示されます。

Rebuild では、変換を試みたすべてのファイルについてログ ファイルに書き込みます。［**エラー時続行**］の設定を無効にした場合、ログ ファイルには、エラーが発生した時点までの情報が記録されます。リビルドが失敗した場合、ログ ファイルにはそのエラーの原因を説明するメッセージが記録されます。

7 ファイルのリビルドが完了してログ ファイルを見終わったら、［**完了**］をクリックします。

CLI バージョン Rebuild 操作

Rebuild コマンド ライン ツールの名前は、Windows の場合は rbldcli.exe、Linux、macOS および Raspbian の場合は rbldcli です。以下のトピックでは、コマンド ライン構文と典型的な Rebuild 操作について説明します。

- 「コマンド ライン パラメーター」
- 「Rebuild を Linux、macOS および Raspbian で実行するには」
- 「Rebuild を Windows で実行するには」
- 「リビルド中のファイルの進行状況を見るには」

コマンド ライン パラメーター

パラメーター（*parameter*）には、このツールで使用する設定を指定します。パラメーターはどのような順序でも使用できます。各パラメーターの前にはハイフン（-）を付けます。ハイフンの後、または、1 文字のパラメーターおよびパラメーター値の後にスペースを入れないでください。

メモ Linux、macOS および Raspbian では、パラメーターで大文字小文字が区別されます。

parameter は以下のように定義されています。

-c

エラーが発生しても、次のデータ ファイルまたは辞書ファイルのリビルドを続行するように、Rebuild に指示します。ツールにより、MicroKernel データ ファイル以外のファイルであることや MicroKernel ファイルでエラーが発生したことが通知されますが、データ ファイルのリビルドは続行されます。エラーはログ ファイルに書き出されます。「ログ ファイル」を参照してください。

ヒント：このパラメーターは、混合するファイルのセットに対してワイルドカード文字（*.*）を指定した場合に特に便利です。混合ファイルのセットとは、MicroKernel ファイルと非 MicroKernel ファイルの組み合わせです。Rebuild は非 MicroKernel ファイルを処理するたび（または MicroKernel ファイルのエラー時）にエラーを報告しますが、処理を続行します。

-d

-d を指定すると、バージョン 6.0 より前の補足インデックス（重複が可能）を 6.x、7.x または 8.x のリンク重複キーのあるインデックスに変換します。

このパラメータを指定しないと、Rebuild はインデックスを繰り返し重複キーとして保持します。

MicroKernel エンジンを介してのみデータ ファイルにアクセスし、かつファイルに比較的多数の重複キーがある場合には、-d オプションを使用して Get Next オペレーションや Get Previous オペレーションのパフォーマンスを向上させることができます。

-m<0 | 2>

"m" パラメータは処理方法を示します。Rebuild はこのパラメータが指定されているかどうかによって処理方法を選択します。このパラメーターを指定しない場合、Rebuild は以下のように処理します。

- 十分なメモリが使用可能である場合は、-m2 をデフォルトの処理方法とします。
- メモリが不十分である場合はもう 1 つの方法の -m0 を使用します。

選択した方法によって影響を受けるメモリの量については、「メモリ量」を参照してください。

0

インデックスの削除や置き換えを行うことなくデータ ファイルまたは辞書ファイルの複製を作成してコピーします。この方法は -m2 の方法に比べて時間がかかります。インデックスをリビルドしない場合には、この方法が使用できます。

-m0 を使用すると、各キー ページの使用率が 55% から 65% のファイルが作成されます。このファイルは書き込み用により最適化されていて、読み取りには最適化されていません。状況によりますが、リビルドにかける時間の余裕がある場合は、書き込み用に最適化されるリビルドを行うのが望ましいでしょう。

「Rebuild 処理の最適化」も参照してください。

2

データ ファイルまたは辞書ファイルの複製を作成し、インデックスを削除して新しいファイルにレコードをコピーした後、インデックスをリビルドします。この方法は、-m0 の方法よりも短時間でリビルドでき、ファイルのサイズも小さくなります。

-m2 の方法では、元のファイルとは異なる物理順序のレコードを含むファイルが作成されることがあります。

-m2 の方法を使用して作成されるファイルのキー ページは 100% 使用済みです。こうすると、ファイルを読み取り用に最適化することができます。

-p<D | P | *bytes*>

- ページ サイズをディスク ストレージまたは処理に最適化します。またはリビルドするファイルに特定のページサイズを指定します。
- このパラメーターを指定しないと、Rebuild は元のファイルのページ サイズを使用します。元のファイルのページ サイズでは現在のデータベース エンジンで動作しない場合、Rebuild はページ サイズを変更し、変更したことを示す情報メッセージを表示します。たとえば、5.x などの古いファイル形式ではページ サイズが 1024 でキーを 24 個持つファイルをサポートしていました。8.x のファイル形式では、ページ サイズ 1024 で 23 個のキーしかサポートしません。したがって、Rebuild は 8.x ファイルを作成する場合、異なるページ サイズを選択します。
- データベース エンジンでは、指定されたページ サイズを無視してそのサイズを自動的に更新することがあります。たとえば、バージョン 9.5 のファイル形式の場合、1536 や 3072 などの奇数ページ サイズはサポートされません。データベース エンジンでは効率を良くするために、ページ サイズを次の有効なページ サイズへ自動的に更新します。旧バージョンのファイル形式の場合、データベース エンジンは追加の条件に基づいてページ サイズを更新することができます。

「インデックス ページ サイズ」も参照してください。

D

ページ サイズをディスク ストレージに最適化します。

『*Zen Programmer's Guide*』の 「ページ サイズの選択」を参照してください。

P

処理に対して最適化します（つまり、そのデータにアクセスするアプリケーションのための最適化です）。-pP では、Rebuild はデフォルトのページ サイズ 4096 バイトを使用します。

「Rebuild 処理の最適化」を参照してください。

bytes

新しいファイルのページ サイズをバイト単位で指定します。9.0 より前のファイル バージョンの場合、有効な値は 512、1024、1536、2048、2560、3072、3584 および 4096 です。ファイル バージョン 9.0 の場合、上記と同じ値に 8192 が加えられます。ファイル バージョン 9.5 の場合、有効な値は 1024、2048、4096、8192 および 16384 です。ファイル バージョン 13.0 の場合、有効な値は 4096、8192 および 16384 です。

-b*directoryname*

リビルドしたファイルに別の場所を指定します。別のサーバー上の場所も指定できます。デフォルトの場所は、データ ファイルのあったディレクトリです。必ず存在する場所を指定してください。Rebuild はディレクトリを作成しません。また、そのディレクトリはデータベース エンジンが起動しているマシン上のディレクトリでなければなりません。

完全修飾したパスまたは相対パスのいずれも使用できます。*directoryname* にはワイルドカード文字を使用しないでください。

ローカル サーバーの場合は、MicroKernel データベース エンジンとメッセージ ルーターがロードされている必要があります。リモート サーバーの場合は、MicroKernel データベース エンジンと通信コンポーネントがロードされている必要があります。

このパラメーターを指定しないと、リビルドされたファイルによって元のデータ ファイルが**置き換えられます**。元のファイルのコピーは保持されません。

このパラメーターを指定すると、リビルドされたファイルは指定の場所に置かれ、元のファイルが**保持されます**。これに関する例外があります。指定した場所に同じ名前のデータ ファイルが既に存在する場合です。元のファイルと同じ名前のファイルが指定された場所に含まれていた場合、Rebuild ではエラーが起こります。たとえば、folder1 というディレクトリにある mydata.mkd をリビルドしようとするとします。リビルドされたファイルを foder2 というディレクトリに保存したいとします。気付かないうちに folder2 にも mydata.mkd が存在していた場合は、Rebuild でエラーが起こり、ログ ファイルを調べるよう通知されます。

メモ：指定した場所（このパラメーターを指定しなかった場合は元のファイルの場所）に対して、ファイルを作成するアクセス権があることを確認してください。

-k*number*

キー番号を指定します。Rebuild はこれを使って元のファイルを読み、リビルドしたファイルを並べ替えます。このパラメーターを指定しないと、Rebuild はもとのファイルを物理順に読み、リビルドしたファイルも物理順で作成します。

「Rebuild 処理の最適化」も参照してください。

-s[D | K | 2D | 2K]

元のファイルの既存のシステム データとキーを保持するファイルをリビルドするか、それらが存在しない場合は追加したファイルをリビルドします。このパラメーターを指定しないと、Rebuild はシステム データまたはキーをリビルドしたファイルに保持しません。

D　新しいシステム データでファイルをリビルドします。システム データにインデックスは作成されません。

K　新しいシステム データでファイルをリビルドします。システム データにはインデックスが作成されます。

2D　新しいシステム データ v2 でファイルをリビルドします。システム データにインデックスは作成されません。13.0 形式のファイルの場合のみ。

2K　新しいシステム データ v2 でファイルをリビルドします。システム データにはインデックスが作成されます。13.0 形式のファイルの場合のみ。

-l*file*　Rebuild のログ ファイルにファイル名を指定します。オプションでパスの場所を指定することができます。デフォルトのファイル名は、Windows、Linux、macOS、および Raspbian では rbldcli.log です。デフォルトの場所は現在の作業ディレクトリです。

以下の条件が適用されます。

- パスの場所は既に存在している必要があります。Rebuild はパスの場所を作成しません。
- ファイル名を指定しないでパスの場所を指定した場合、Rebuild はこのパラメーターを無視してデフォルトのファイル名と場所を使用します。
- パスの場所を指定しないでファイル名を指定した場合、Rebuild はデフォルトの場所を使用します。
- 指定した場所に対する読み取りおよび書き込みのアクセス権を持っている必要があります。ファイルのアクセス権のためにログ ファイルを作成できなかった場合、Rebuild はデフォルトの場所を使用します。

「ログ ファイル」も参照してください。

-pagecompresson　「file」のページ圧縮をオンにすると、以下の条件が設定されます。

- データベース エンジンのバージョンは 9.5 以上
- 作成ファイルのバージョン設定は 0950（9.5）以上

-pagecompressoff　「file」のページ圧縮をオフにします。このパラメーターは、「file」にページ圧縮が指定されていない場合は無効です。

-recordcompresson　「file」のページ圧縮をオンにします。

-recordcompressoff　「file」のレコード圧縮をオフにします。このパラメーターは、「file」にレコード圧縮が指定されていない場合は無効です。

-f<6 | 7 | 8 | 9 | 95 | 13>　リビルド後のデータ ファイルまたは辞書ファイルのファイル形式を指定します。ファイル形式は、バージョン 6.x、7.x、8.x および 9.x が使用できます。次の例ではファイルを 9.0 形式でリビルドします。

```
rbldcli -f9 file_path¥class.mkd
```

次の例ではファイルを 9.5 形式でリビルドします。

```
rbldcli -f95 file_path¥class.mkd
```

指定しない場合、Rebuild は MicroKernel の［作成ファイルのバージョン］設定オプションに指定されている値を使用します。

メモ 1：現在のデータベース エンジンでサポートされているバージョンより新しいファイル形式を指定した場合、Rebuild はそのエンジンでサポートされている最新のファイル形式を使用します。Rebuild はこれに関してはエラーもメッセージも報告しません。

メモ 2：Rebuild はインデックスのデータ型を変換しません。ファイルを古いデータベース エンジンで使用するために古いファイル形式にリビルドする場合、エンジンが使用されているデータ型をサポートしているかどうかを確認してください。必要に応じ、アプリケーションとそのデータベース エンジンによって、手作業でデータ型を調整する必要があります。

例 1：データ ファイルに WZSTRING データ型を使用するインデックス フィールドが含まれています。データ ファイルを 6.x 形式にリビルドする場合、WZSTRING データ型は変換されません。このデータ ファイルを Btrieve 6.15 エンジンで使用することはできません。このエンジンは WZSTRING データ型をサポートしません。

例 2：データ ファイルに NULL が含まれています。データ ファイルを 7.x ファイル形式にリビルドします。真の NULL は変換されません。このデータ ファイルを Zen 7 エンジンで使用することはできません。このエンジンは 真の NULL をサポートしません。

-uid*uname*　セキュリティが設定されているデータベースにアクセスする権限を与えられたユーザー名を指定します。

-pwd*pword*　*uname* で識別されるユーザーのパスワードを指定します。*uname* が指定された場合、*pword* は必ず指定する必要があります。

-db*dbname*　セキュリティが設定されたデータベース名を指定します。

file および *@command_file* は次のように定義されます。

file　変換するデータ ファイルおよび辞書ファイルを指定します。元のファイルが現在の作業ディレクトリにない場合は、完全修飾パスまたは相対パスのいずれかを使用して場所を含めてください。ファイル名にはアスタリスク（*）ワイルドカード文字を使用して、複数のファイルを指定することができます。

メモ：元のファイルにオーナー ネームが含まれている場合、Rebuild はリビルドしたファイルにオーナー ネームとレベルを適用します。

@command_file　Rebuild で実行されるコマンド ファイルを指定します。1 つのコマンド ファイルに複数の項目を指定できます。コマンド ファイルの各項目には、コマンド ライン パラメーター（ある場合）と変換するファイルのセットを指定し、その後に <end> または ［end］を続けます。

変換するファイルを指定する場合には、完全なディレクトリ名を使用します。ファイル名にはアスタリスク（*）ワイルドカード文字を使用できます。

次に、Rebuild コマンド ファイルの例を示します。

```
-c d:¥mydir¥*.*<end>
-c -p1024 e:¥dir¥*.*<end>
-m0 -k0 d:¥ssql¥*.*<end>
```

▶▶ Rebuild を Linux、macOS および Raspbian で実行するには

1　ログインするアカウントが Zen ユーティリティを実行する権限を持っていることを確認してください。

デフォルトでは、ユーティリティを実行するには zen-svc ユーザーとしてログインする必要があります。zen-svc ユーザーにはパスワードがありません。su コマンドを使用することによって root アカウントでのアクセスのみを行うことができます。zen-svc 以外のアカウントからこのユーティリティを使用するには、まず .bash_profile を変更する必要があります。『*Getting Started with Zen*』の「Linux、macOS、Raspbian での Zen のアカウント管理」を参照してください。

2 `/usr/local/actianzen/bin` ディレクトリに移動します。

3 プロンプトに、次のどちらかのコマンドを入力します。

```
rbldcli [-parameter ...] file
```

または

```
rbldcli @command_file
```

parameter、*file*、および *@command_file* は、「コマンド ライン パラメーター」に定義されています。

使い方の例

以下の例はエラー時も処理を続けます。ページ サイズは 4096 バイトに設定され、リビルドされたファイルはサーバー上の別のディレクトリに保存します。

```
rbldcli -c -p4096 -b/usr/local/actianzen/tmp /usr/local/actianzen/data/Demodata/
    *.mkd
```

▶▶ Rebuild を Windows で実行するには

1 Zen がインストールされているシステムで、コマンド プロンプトを開きます。

2 状況により、¥bin ディレクトリをプログラム ファイルをインストールしたディレクトリに変更します。その場所が Path システム変数に含まれている場合は、この操作は不要です。

3 プロンプトに、次のどちらかのコマンドを入力します。

```
rbldcli [-parameter ...] file
```

または

```
rbldcli @command_file
```

parameter、*file*、および *@command_file* は、「コマンド ライン パラメーター」に定義されています。

使い方の例

以下の例はエラー時も処理を続けます。ページ サイズは 4096 バイトに設定され、リビルドされたファイルはサーバー上の別のディレクトリに保存します。

```
rbldcli -c -p4096 -bc:¥dbtemp c:¥datafiles¥*.mkd
```

▶▶ リビルド中のファイルの進行状況を見るには

Rebuild は、ファイルごとに処理されたレコード数を画面上に表示します。1 度に 50 レコードずつ増加します。さらに、Rebuild はこれらの情報をテキスト ログ ファイルに書き込みます。「ログ ファイル」を参照してください。

ディスクリプション ファイル

17

ディスクリプション ファイルを使用して Btrieve ファイル情報を保存する

ディスクリプション ファイルは、Maintenance ツールがデータ ファイルやインデックスの作成に使用するファイル スペックとキー スペックの記述を含む ASCII テキスト ファイルです。作成したファイルの情報を保存するための媒体としてディスクリプション ファイルを使用することもできます。ディスクリプション ファイルは、リレーショナル エンジンで使用されるデータ辞書ファイル（DDF）とは異なります。

ディスクリプション ファイルには 1 つ以上の要素が含まれます。1 つの要素は、キーワード、等号（=）、それに値がスペースなしで続く形式で構成されます。ディスクリプション ファイルの各要素は、データ ファイルまたはキー スペックの特性に対応しています。

メモ ディスクリプション ファイルを使用する前に、Btrieve の基本について熟知しておく必要があります。これらのトピックスの詳細については、『*Zen Programmer's Guide*』を参照してください。

この付録では以下の項目について説明します。

ディスクリプション ファイルの規則

ディスクリプション ファイルを作成する際は、以下の規則に従います。

- 要素は、大文字または小文字のいずれかで入力する。
- 要素を区切る場合は、以下の例のように、区切り文字（スペース、タブ、またはキャリッジ リターン / ライン フィード）を使用する。

  ```
  record=4000
  key=24
  ```

- ディスクリプション ファイルの要素を適切な順序で指定する。表 75 に要素の適切な順序が示されています。
- 要素の依存関係は、すべて指定する。たとえば、ディスクリプション ファイルで `nullkey=allsegs` を指定した場合は、`value=` という要素の値も指定します。
- キーには、**キー数**要素で指定した数と同じ数のキーを定義する。たとえば、`key=12` を指定した場合は、ディスクリプション ファイルでも 12 個のキーを定義します。
- キーに複数のセグメントがある場合は、各キー セグメントに以下の要素を定義する。
  - キー ポジション
  - キー長
  - 重複キー値
  - 変更可能キー値
  - キー タイプ

  各セグメントの**降順ソート**要素は、省略可能です。
- ファイル中のキーで ACS を使用している場合は、ACS ファイル名または ISR テーブル名を指定する。この情報は、キーの最後の要素（現在のキーのみに適用）またはディスクリプション ファイルの最後の要素（データ ファイル全体に適用）として含めることができます。
  - 1 つのキーに対して指定できる ACS は 1 つのみで、ACS ファイル名または ISR テーブル名も指定します。同一ファイル内の異なるキーには、異なる形式の照合順序を使用できます。たとえば、キー 0 に ACS ファイル名、キー 1 に ISR テーブル名を使用できます。
  - 同一キーの異なるセグメントに、異なる照合順序を使用することはできません。
  - ディスクリプション ファイルの最後に ACS を指定した場合は、その ACS がデフォルトとして使用されます。たとえば、キーに `alternate=y` を指定し、そのキーに ACS ファイル名または ISR テーブル名を指定しない場合、データベース エンジンでは、ファイルの最後に指定されている ACS ファイル名または ISR テーブル名を使用します。
  - 新しいキーを作成する際に `alternate=y` を指定し、ACS ファイル名または ISR テーブル名を指定しない場合、データベース エンジンでキーは作成されません。
- **ディスクリプション ファイル**の要素がオプションの場合は、それを省略することができる。
- ディスクリプション ファイルには、書式設定文字列を使用しない。一部のワード プロセッサでは、テキスト ファイルに書式設定文字列が埋め込まれます。

ディスクリプション ファイルの例

このセクションにあるディスクリプション ファイルのサンプルは、データ ファイルの説明です。このデータ ファイルのページ サイズは 512 バイトで、キーは 2 つです。レコードの固定長部分は 98 バイトです。ファイルで可変長レコードは使用可能ですが、ブランク トランケーションは使用できません。

ファイルはレコード圧縮を使用します。また、可変長部割り当てテーブル（VAT）を許可し、空きスペース スレッショルドは 20 % に設定されています。MicroKernel エンジンは、ファイルを作成する際に 100 ページ、つまり 51,200 バイトをプリアロケートします。ファイルには、キー 0 およびキー 1 の 2 つがあり、キー 0 には 2 つのセグメントがあります。

図 39 では、同一の ACS ファイル名（upper.alt）が両方のキーで使用されています。図 40 では、キー 0（lower.alt）およびキー 1（upper.alt）で、異なる ACS ファイル名を使用しています。図 41 のファイルには、ログに使用されるシステム定義キーのみが含まれています。

図 39 キー セグメントに ACS のファイル名を使ったディスクリプション ファイルのサンプル

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

position=1 length=5 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y

position=6 length=10 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n

position=16 length=2 duplicates=n
modifiable=y type=numeric descending=y
nullkey=n segment=n

name=c:¥myacsfiles¥upper.alt
```

ファイル仕様

キー 0 セグメント 1

キー 0 セグメント 2

キー 1

図 40 キー セグメントに ACS のファイル名を使ったディスクリプション ファイルのサンプル

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

position=1 length=5 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y
name=sys:¥zen¥demodata¥lower.alt

position=6 length=10 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n
name=c:¥myacsfiles¥lower.alt

position=16 length=2 duplicates=n
modifiable=y type=numeric descending=y
nullkey=n segment=n
name=c:¥myacsfiles¥upper.alt
```

ファイル仕様

キー 0 セグメント 1

キー 0 セグメント 2

キー 1

図 41 システム定義キーをログに使用するディスクリプション ファイルのサンプル

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y sysdataonrecord=loggable
```

ディスクリプション ファイルの要素

ディスクリプション ファイルの要素は特定の順序で並べる必要があります。表 75 は、ディスクリプション ファイルの要素を適切な順序で示しています。この表では、各要素に必要な書式および指定可能な値の範囲を示します。

- アスタリスク（*）は、その要素が省略可能であることを意味します。
- シャープ記号（#）は、その要素が以前のバージョンの MicroKernel にのみ適用されることを意味します。
- パーセント記号（%）は、その要素が現バージョンの MicroKernel にのみ適用されることを意味します。

表 75　ディスクリプション ファイルの概要

要素	キーワードとフォーマット	範囲	解説
ファイル スペック情報			
コメント ブロック *	/*.....*/	5120 バイト	なし。
レコード長	record=*nnnn*	4 – ページ サイズの制限	なし。
可変長レコード	variable=<y|n>	適用外	キー オンリー ファイルには適用できません。
予約重複ポインター *	dupkey=<*nnn*>	0 – 119	リンク重複キーを追加するファイルにのみ適用できます。
ブランク トランケーション *	truncate=<y|n>	適用外	レコード圧縮が使用されるファイルには適用できません。
レコード圧縮 *	compress=<y | n>	適用外	キー オンリー ファイルには適用できません。「レコードおよびページ圧縮」も参照してください。
キー数	key=*nnn*	0 – 119	データ オンリー ファイルを作成する場合は、0 を指定します。 キー カウントがゼロの場合、[データのインクルード] および [システム データの使用] は no に設定することはできません。
ページ サイズ	page=*nnnn*	512 – 9.0 より前のファイル バージョンの場合は 4096 バイト（512 バイトの倍数で最大 4096 バイト） 512、1024、1536、2048、2560、3072、3584、4096 または 8192 バイト（ファイル バージョンが 9.0 の場合） 1024、2048、4096、8192 または 16384 バイト（ファイル バージョンが 9.5 の場合） 4096、8192 または 16384 バイト（ファイル バージョンが 13.0 の場合）	
ページ プリアロケーション *	allocation=*nnnnn*	1 – 65535	なし。

表 75　ディスクリプション ファイルの概要

要素	キーワードとフォーマット	範囲	解説
既存ファイルの置き換え *#	replace=<y|n>	適用外	なし。
データのインクルード *	data=<y|n>	適用外	キー オンリー ファイルを作成する場合は、*n* を指定します。重複を許可し、システム定義のキーを使用するキー オンリー ファイルを作成することはできません。
空きスペース スレッショルド *	fthreshold=<5|10|20|30>	適用外	可変長レコードのあるファイルのみに適用できます。デフォルト値は 5 です。
可変長部割り当てテーブル（VAT）	huge=<y|n> # vats=<y|n>	適用外	可変長レコードのあるファイルのみに適用できます。
インデックス バランス *	balance=<y|n>	適用外	なし。
キー番号の使用 *	usekeynum=<y|n>	適用外	キー番号の要素と併用されます。
[1]システム データの使用 *%	sysdataonrecord= <n|loggable>	適用外	要素が指定されない場合は、MicroKernel の設定が使用されます。キー オンリーのファイルを作成する場合は、MicroKernel の設定が使用され、この要素は無視されます。
[1] システム データ v2 の使用 *	sysdata2=<n|y>	適用外	システム データ v2 をファイルへ追加します。キーオンリー ファイルでは使用しないでください。13.0 形式のファイルが必要です。
ページ圧縮 *	pagecompress=<y|n>	適用外	「レコードおよびページ圧縮」も参照してください。
ファイル バージョン*	version=0x<HH>	適用外	要素が指定されていない場合、システムのデフォルトのファイル バージョンが使用されます。要素が指定されている場合は、以下の例に示すように 16 進数を使用します。 • 0x90：バージョン 9.0 の場合 • 0x95：バージョン 9.5 の場合 • 0xD0：バージョン 13.0 の場合
キー スペック情報			
キー番号 *	keynum=*nnn*	0 – 118	ファイル内で重複せず、昇順に指定され、ファイルのページ サイズに有効であることが必要です。ファイルを作成する場合のみに適用されます。
キー ポジション	position=*nnnn*	1 – レコード長	レコード長を超えてはいけません。

表 75　ディスクリプション ファイルの概要

要素	キーワードとフォーマット	範囲	解説
キー長	length=*nnn*	キー タイプの制限	キー タイプが指示する制限を超えてはいけません。バイナリ キーのキー長は、偶数であることが必要です。キー ポジションとキー長の合計が、ファイルのレコード長を超えてはいけません。
重複キー値	duplicates=<y|n>	適用外	重複を許可し、システム定義のキーを使用するキー オンリー ファイルを作成することはできません。
変更可能キー値	modifiable=<y|n>	適用外	なし。
キー タイプ	type= *validMKDEKeyType*	適用外	名前全体（例、「float」）または最初の三文字（例、「flo」）を入力します。
降順ソート *	descending=<y|n>	適用外	なし。
オルタネート コレーティング シーケンス（ACS）	alternate=<y|n>	適用外	大文字小文字を区別する STRING や LSTRING、WSTRING、WZSTRING、ZSTRING キーのみに適用できます。既存ファイルにインデックスを追加作成する際に、そのインデックスでデータ ファイルの最初にある ACS 以外の ACS を使用する場合は、`caseinsensitive=y` と共に使用します。
大文字小文字無視のキー *	caseinsensitive=<y|n>	適用外	ACS が使用されない STRING、LSTRING、または ZSTRING キーのみに適用できます。
繰り返し重複 *	repeatdup=<y|n>	適用外	キー オンリーのファイルを作成する場合は、繰り返し重複を使用します。この要素を使用する場合は、duplicates=y を使用します。
ヌル セグメント *	nullkey=<allsegs | n | anyseg |>	適用外	なし。
ヌル キーの値	value=*nn*	16 進の 1 バイト	ヌル セグメント要素に使用されます。
セグメント キー	segment=<y|n>	適用外	なし。
オルタネート コレーティング シーケンス（ACS）	name=*sequenceFile* または *isr=table name (%)*	有効なパス、あるいは使用中のオペレーティング システムで有効な値、または -1	オルタネート コレーティング シーケンス要素と共に使用されます。
[1] データベース エンジンがシステム データを追加する場合、結果レコードがファイルの既存ページ サイズを超えることがあります。その場合、データベース エンジンは、ページ サイズを自動的に次の適切な大きさに調整します。			