ACTIAN™

Advanced Operations Guide

Zen v16

Activate Your Data™

Contents

Configuration Reference   33

High Availability Support   201

Zen and Hypervisor Products   217

Workgroup Engine in Depth   223

Monitoring 233

Testing Btrieve Operations 301

About This Documentation

This documentation covers procedures and technical information for the advanced user.

Some of the content presented here may not apply to you. For example, the topic on gateway configuration does not apply to Zen server engines. These differences are clearly marked.

Actian would appreciate your comments and suggestions. As a user of our documentation, you are in a unique position to provide ideas that can have a direct impact on future releases of this and other manuals. If you have comments or suggestions for the product documentation, post your request in the community forum on the Actian website.

Who Should Read This Documentation

This documentation is provided for advanced users. Advanced users are considered to have a strong understanding of the underlying operating systems on which you run your Zen-based applications. Advanced users should be comfortable configuring their operating system, and in many cases, must have administrative permissions to configure the database engine. Advanced users may include the following:

• Network administrators of networks where one or more Zen-based applications are installed

• Value-added resellers of Zen-based applications

• Developers of Zen-based applications

Zen Databases

This section is divided into the following topics:

- Named Database
- Metadata
- Identifiers and Object Names
- The Default Database and the Current Database
- File Structure
- Access Methods
- Client-Server Communications
- Database Code Page
- ODBC DSN Creation Options
- Using the idshosts File

Named Database

A named database (also called a DBname) is a database with a logical name that allows users to identify it without knowing its location. Zen requires that all databases be named. When you name a database, you associate that name with a particular dictionary directory path and one or more data file paths.

A named database is connected to through various access methods. For ODBC access, for example, you must set up a data source name (DSN) to refer to the named database. Multiple DSNs may point to the same named database. See ODBC Database Access in *ODBC Guide*. For other access methods, application developers can connect to a named database using the API for that access method. Refer to the developer reference guides in the Zen documentation.

Note: To work with named databases, you must log into the computer where the database engine is located, using an operating system user name that has administrator-level privileges or is a member of the Zen_Admin security group.

The easiest way to create a named database is by using Zen Control Center. See To create a new database in *Zen User's Guide*. Application developers can also create a named database through different access methods APIs. For example, see CREATE DATABASE for SQL, PvCreateDatabase() for DTI, and Data Access Application Blocks for ADO.NET.

Metadata

The Relational Engine supports two versions of metadata, referred to as version 1 (V1) and version 2 (V2). V2 metadata allows for identifier names up to 128 bytes long for many identifiers, permissions on views and stored procedures, and data dictionary files (DDFs) specific for V2 metadata.

See SQL Grammar Support in *ODBC Guide*.

Identifiers and Object Names

An *identifier* is the name of a database or of a column, table, procedure, or other named object within the database. Identifiers are designated as either regular or delimited.

Regular Identifiers

A *regular identifier* is an identifier that is not surrounded by double quotes. Regular identifier must begin with a letter, either upper or lower case. The remainder of the identifier can consist of any combination of upper or lower case letters, digits, and valid characters.

You cannot use a reserved word as a regular identifier.

Regular identifiers are case-insensitive.

Delimited Identifiers

A *delimited identifier* is an identifier surrounded by double quotes. Delimited identifier can consist of any string of valid characters enclosed in double quotes.

While it is not recommended, reserved words can be used as delimited identifiers. For example, INSERT is not permitted as a regular identifier, but "INSERT" is permitted as a delimited identifier. If an identifier is also a keyword, it must be delimited by double quotation marks. For example, SELECT "*password*" FROM my_pword_tbl. *Password* is a keyword in the SET PASSWORD statement, so it must be delimited.

Identifier Restrictions

In addition to the general restrictions listed above, the following table lists restrictions specific to each type of identifier.

Identifier	Length Limit (bytes)		Invalid Characters[1]	Notes
	V1[2]	V2[3]		
Column	20	128	\ / : * ? " < > |	Must begin with a letter Cannot be null
Database	20	20	` ~ ! @ # $ % ^ & * ( ) _ - + = }] { [| \ : ; " < , ' > . ? /	Must begin with a letter

Identifier	Length Limit (bytes)		Invalid Characters[1]	Notes
	V1[2]	**V2**[3]		
Function (user-defined)	30	128	For regular identifiers: `` ` `` ~ ! @ # $ % ^ & * ( ) - + = }] { [| \ : ; " < , ' > . ? /	Valid characters are letters, digits, and the underscore ("_") Must begin with a letter
			For delimited identifiers: none	Name must be enclosed in double quotes
Group	30	128	\ / : * ? " < > | (and space character)	Cannot be MASTER
Index	20	128	\ / : * ? " < > | (and space character)	Cannot start with UK_ if you create the index with Zen Control Center (ZenCC) If you create an index outside of ZenCC that starts with UK_, you cannot edit the index with ZenCC
Key (foreign or primary)	20	128	\ / : * ? " < > | (and space character)	Must begin with a letter A foreign key and an index cannot be named the same within the same table
Password	8	128	; ? " '	Cannot start with a blank (space character) Cannot be null Any displayable character is permissible except for those listed in the Invalid Characters column
Procedure (stored)	30	128	For regular identifiers: `` ` `` ~ ! @ # $ % ^ & * ( ) - + = }] { [| \ : ; " < , ' > . ? /	Valid characters are letters, digits, and the underscore ("_") Must begin with a letter
			For delimited identifiers: none	Name must be enclosed in double quotes
Table	20	128	\ / : * ? " < > | (and space character) # ##[4]	Invalid characters apply to both regular and delimited identifiers

Identifier	Length Limit (bytes) V1[2]	V2[3]	Invalid Characters[1]	Notes
Trigger	30	128	For regular identifiers: ` ~ ! @ # $ % ^ & * ( ) - + = }] { [	\ : ; " < , ' > . ? /	Valid characters are letters, digits, and the underscore ("_") Must begin with a letter
			For delimited identifiers: none	Name must be enclosed in double quotes
User	30	128	\ / : * ? " < >	(and space character)	Cannot be MASTER or PUBLIC
View	20	128	For regular identifiers: ` ~ ! @ # $ % ^ & * ( ) - + = }] { [	\ : ; " < , ' > . ? /	Valid characters are letters, digits, and the underscore ("_"). Must begin with a letter
			For delimited identifiers: none	Name must be enclosed in double quotes.

[1]Unless otherwise noted, invalid characters apply both to regular and to delimited identifiers.

[2]Applies to version 1 (V1) metadata. See SQL Grammar Support in *ODBC Guide*.

[3]Applies to version 2 (V2) metadata. See SQL Grammar Support in *ODBC Guide*.

[4]The names of temporary tables begin with # or ##. Therefore, # and ## are invalid characters with which to begin the name of permanent tables. See CREATE (temporary) TABLE in *SQL Engine Reference*.

Unique Scope

Identifiers generally must be unique within a certain *scope*. That is, instances of the same type of object using the same name cannot be used within the same arena. The following table shows the arena, or the *scope*, within which a given object name must be unique.

A name for this type of object...	... must be unique within this scope: Database	Table	Stored Procedure	Other
Database				All databases hosted by a given database engine
Table	X			

A name for this type of object...	... must be unique within this scope:			
	Database	**Table**	**Stored Procedure**	**Other**
Trigger, stored procedure, user-defined functions	X			
User or group	X			
View	X			
Constraint	X			
Column		X		
Index		X		Cannot have the same name as a foreign key
Key (foreign)		X		Cannot have the same name as an index
Cursor			X	

The Default Database and the Current Database

To support existing applications that do not specify a database name when creating or opening Btrieve files, Zen maintains the concept of a default database for each transactional database engine. The *default database* is a predefined database named DefaultDB. To make use of the new security models without having to modify your application code, you can associate your Btrieve data directories with the default database, then set up users and privileges in the default database to control access to the data files in those directories.

The database engine also understands the concept of the *current database* for each client connection. If no database name is specified in a Btrieve Login (78), Create (14), or Open (0) operation, the transactional engine assumes the operation is associated with the current database. For each client, the current database is the database to which the most recent Login (78) operation occurred (explicit login). If the client computer has requested no explicit login operations, the current database is the database to which the most recent Create (14) or Open (0) operation occurred (implicit login). If no explicit or implicit logins have occurred, then the current database is the default database, described in the preceding paragraph. Note that the current database may change at any time when the given client performs an implicit or explicit login, or closes the last file handle, making DefaultDB the current database. The current database for each client is independent of other clients' activities.

The simplest way to configure the new security model for existing applications is to associate all Btrieve data directories with the default database, and set up rights for the group PUBLIC within this database. The group PUBLIC is automatically created along with the Master user when you enable security for a database. See MicroKernel Engine Security Quick Start.

File Structure

All Zen databases use a common data format. The shared format allows different access methods to use the same data. All access methods operate through the Zen MicroKernel Engine.

Each Zen database table is a separate file with a default file extension of .mkd. However, you can use any file name extension. A MicroKernel file may contain both data and indexes and is organized into various types of pages. A MicroKernel file contains data in the shared data format.

Each Zen database also contains a set of data dictionary files with a file extension of .ddf. The .ddf files contain the database schema. DDFs for V1 and V2 metadata use different file names.

Note: The MicroKernel Engine is unconcerned with the schema of the data apart from key fields. However, referential integrity or access via SQL requires schema information.

The names and locations of Zen databases reside in a binary file named dbnames.cfg.

All files associated with a Zen database can be viewed from the operating system.

Zen File Types	Description
Data (common format)	For relational databases, the default file name is *tablename*.mkd. Each database table is a MicroKernel data file. For transactional data files, the application specifies the file name.
Data Dictionary	Files with an extension of .ddf. See System Tables in *SQL Engine Reference*.
Database Names Configuration	Dbnames.cfg, a binary file of Zen database names and locations. For default locations, see Where are the files installed? in *Getting Started with Zen*.

File Size

The file size limit depends on file version and page size, and for older file versions, the number of records per page. The following tables summarize and compare these limits.

File Version 16.0

The maximum size of a data file is 64 TB. The maximum key length is 1024 bytes instead of 255 bytes in previous versions. The longer key length enables indexing of longer data, such as

VARCHAR, without having to use PARTIAL INDEX. Note that keys longer than 255 bytes require a 16384-byte page size.

Maximum Pages (in millions)	File Size (TB) for Various Page Sizes (bytes)		
	4096	8192	16384
4096	16 TB	32 TB	64 TB

File Version 13.0

The maximum size of a data file is 64 TB. You must use a file format of 13.0 or newer to have a single file size larger than 256 GB.

Maximum Pages (in millions)	File Size (TB) for Various Page Sizes (bytes)		
	4096	8192	16384
4096	16 TB	32 TB	64 TB

File Version 9.5

The maximum size of a data file is 256 GB. You must use a file format of 9.5 or newer to have a single file size larger than 128 GB.

Note that the following table assumes no record compression on the file. If you use record compression, take into account that additional records are stored per page. See Choosing a Page Size and Estimating File Size, both in *Zen Programmer's Guide*.

Records per Page	Maximum Pages (in millions)	File Size (GB) for Various Page Sizes (bytes)				
		1024	2048	4096	8192	16384
1–15	256	256	256	256	256	256
16–31	128	128	256	256	256	256
32–63	64	64	128	256	256	256
64–127	32	32	64	128	256	256
128–255	16	16	32	64	128	256
256–511	8	—	16	32	64	128
512–1023	4	—	—	16	32	64

Records per Page	Maximum Pages (in millions)	File Size (GB) for Various Page Sizes (bytes)				
		1024	2048	4096	8192	16384
1024–2047	2	—	—	—	16	32
2048–4095	1	—	—	—	—	16

File Versions 9.0 or Older

The maximum size of a data file is 128 GB. You must use a file format of 9.0 or newer to have a single file size larger than 64 GB.

Note that the following table assumes no record compression on the file. If you use record compression, take into account that additional records are stored per page. See Choosing a Page Size and Estimating File Size, both in *Zen Programmer's Guide*.

File Version	Records per Page	Maximum Pages (in millions)	File Size (GB) for Various Page Sizes (bytes)								
			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	—	16	24	32	40	48	56	64	128
8.x	any	16	8	16	24	32	40	48	56	64	—
7.x	any	16	8	16	24	32	40	48	56	64	—
6.x	any	16	4	4	4	4	4	4	4	4	—
Pre-6.x	any	16	4	4	4	4	4	4	4	4	—

File Segmentation

By default, a data file is automatically divided into 2 GB operating system file segments as its size passes that boundary. The configuration property Limit Segment Size to 2 GB allows you to specify whether you want files divided into 2 GB segments or unified in a single, nonsegmented file. The advantage of using a larger nonsegmented file is more efficient disk I/O. Therefore, you

can expect increased performance. Note that files version 13.0 and later do not support segmentation.

The configuration option is part of the Performance Tuning properties for a database engine. See To set the properties in ZenCC for an engine, and Limit Segment Size to 2 GB.

The property is set to on by default, causing files to segment at 2 GB boundaries as with previous releases. If you set the property to off, files can increase past the 2 GB boundary. See also Automatic Upgrade of File Version for additional information relating to the configuration property.

Any non-segmented files are subject to the limit on file size specified by your operating system. For example, creating a large file on a FAT32 file system with Limit Segment Size to 2 GB turned off creates multiple 4 GB file extensions. If a previously created file is already segmented, that segmentation remains on the file.

Automatic Upgrade of File Version

If the configuration property Create File Version is set to 9.0 or higher, version 8.x files are automatically converted to version 9.0 files when they reach the file limits for version 8.x, which is 64 GB. The following table summarizes this behavior.

Configuration Property Setting for Create File Version	Configuration Property Setting for Limit Segment Size to 2 GB	File Size At Which Automatic Upgrade of File Version Occurs
9.0 (default)	Yes (default; option checkmarked)	64 GB (8.x maximum file size)
9.0 (default)	No (option not checkmarked)	2 GB (8.x segment size)

For example, a version 8.x file that is 5 GB in size has already passed the 2 GB segmentation boundary. Because the file is already segmented, the segmentation remains on the file. Such a file would continue to segment and grow in size until it reaches 64 GB, at which size the automatic upgrade would occur. This is true whether the configuration property is set to yes or no because the file is already segmented. As the file grows beyond 64 GB, it will continue to segment until it reaches the maximum size allowed for a version 9.0 file, 128 GB.

A version 8.x file that is 1.5 GB in size would continue to grow until it reaches 2 GB in size. At that point, the automatic upgrade occurs if the configuration property is set to no. The file can continue to grow as a non-segmented file up to the size limit for version 9.0 files, 128 GB. If the configuration setting is set to yes, the 2 GB file would continue to segment and grow until it reaches 64 GB in size. At that size, the maximum for a version 8.x file, the automatic upgrade to

version 9.0 occurs. As the file grows beyond 64 GB, it will continue to segment until it reaches the maximum size allowed for a version 9.0 file, 128 GB.

The Create File Version option is part of the Compatibility properties for a database engine. See To set the properties in ZenCC for an engine.

Note: Automatic upgrade of file version works only for an 8.x to a 9.0 file format.

Access Methods

The two primary methods in which data is accessed from Zen databases are transactional and relational.

With transactional, an application program navigates up and down along either physical or logical pathways through data records. Using a transactional API, an application program provides direct control and allows a developer to optimize data access based on knowledge of the underlying structure of the data. Btrieve is an example of a transactional database engine.

Relational is an access method in which data is represented as collections of tables, rows, and columns. The relational model insulates the developer from the underlying data structure and presents the data in a simple table format. ODBC is an example of a relational access method.

A single application program may include both types of access. For example, an application may use transactional access for adding and changing data, and relational access for querying the data and report writing.

You need to know the access methods used by the application programs that rely on your installation of Zen. The access methods may have different configurations. You may need to customize the configuration to optimize a particular access method.

Also, troubleshooting is easier when you are aware of the access methods used by a given application program. For example, if an application program uses relational access through ODBC, you may need to troubleshoot a problem at the ODBC level rather than at the database management system.

See Configuration Reference for the tasks and references pertaining to customizing configurations.

Client-Server Communications

The MicroKernel Engine supports two types of processing modes, local and client-server. An application accessing the database in local mode accesses a local copy of the engine. The local engine calls upon the operating system of the workstation which performs the I/O on a local or networked hard disk.

Client-server mode uses a server MicroKernel Engine executing on a shared file server. When an application program accesses the database engine in client-server mode, the requester connects to the remote engine. This requester passes transactional-level requests and data records between the application program and the server engine using the network protocol supported by the operating system. File I/O functions are completely handled by the server engine in client-server mode and the workstation has no operating system handles allocated to shared data files. Database manipulation is performed by the server-based engine on behalf of each workstation.

Note that the processing mode is determined by the configuration of the workstation and not the application program itself. This means that an application is capable of accessing both local and client-server database engines. The application program does not have to be recompiled to switch the application to client-server mode from local mode.

Both Workgroup and server engines can operate in either mode. When an application on the same computer as the database engine accesses the engine, it is operating in local mode. When an application on a different machine access the engine, it is operating in client-server mode.

The client-server configurations may be customized for the Workgroup and server versions of Zen. Configuration settings exists in the Zen Control Center (ZenCC) to facilitate the configuration of client-server configurations as well as stand-alone configurations.

See Configuration Reference for the tasks and references pertaining to configuring the client-server communications and database engine.

Database Code Page

Encoding is a standard for representing character sets. Character data must be put in a standard format, that is, encoded, so that a computer can process it digitally. An encoding must be established between the Zen database server and a Zen client application. A compatible encoding allows the server and client to interpret data correctly.

The encoding support is divided into database code page and client encoding. The two types of encoding are separate but interrelated. For ease of discussion, database code page and client encoding are discussed together. See Setting Up Network Communications for Clients in *Getting Started with Zen*.

ODBC DSN Creation Options

See DSN Setup and Connection Strings in *ODBC Guide*. This topic also discusses ODBC connection strings.

Using the idshosts File

Typically, an application provides its own file location information. As an alternative, you may provide file location mapping based on information in a text file, idshosts.

The idshosts file was one aspect of Zen (IDS). IDS has been removed from the core product but the idshosts file is still configurable.

If your applications do not use the mapping feature through idshosts, set the configuration setting **Use IDS** to off. If your applications already use idshosts, or if you prefer to use this alternative method to map file locations, set **Use IDS** to on. See Use IDS.

Note that performance is slower when the idshosts file is used because of the time required to access the file and read its contents.

An idshosts file may be used only with a Windows or Linux client requester. The client may communicate with a Zen server on Windows or Linux.

Note: Zen 8.5 or later is required if you set **Use IDS** to On. The requester uses database URIs to represent the IDS information. Database URIs were added with PSQL 8.5. See Database URIs in *Zen Programmer's Guide*.

If **Use IDS** is set to on, you must also set **Use Remote MicroKernel Engine** to on. **Use Remote MicroKernel Engine** is on by default.

See Use IDS and Use Remote MicroKernel Engine.

Format of idshosts Entries

Refer to the comments in the idshosts file itself for how to format entries in the file. The comments also provide example mappings. By default, for Windows platforms, the idshosts file is installed to the \bin directory under the database client installation directory. For Linux and Raspbian, idshosts is installed to the /etc directory under the database client installation directory (for example, /user/local/actianzen/etc).

Concepts of Database Maintenance

Zen is a comprehensive database management system built around the MicroKernel Database Engine (MKDE). Zen offers easy installation, uncomplicated maintenance, and high levels of performance and reliability. While Zen can run for months or years with practically no maintenance, you can get the most out of it by understanding some of its unique features and learning how to perform useful tasks. This manual describes how to tune, configure, and manage your Zen engine and associated databases.

- Configurations
- Database Security
- Data Archival and Restoration
- Troubleshooting
- Helpful Utilities

Configurations

You can configure separate settings for the server and client aspects of the database engine. The settings allow you to optimize the performance of the engine based on your business requirements.

The following figure illustrates the flow from an application program to the database files at the operating system. The database engine resides between the application program and the data files.
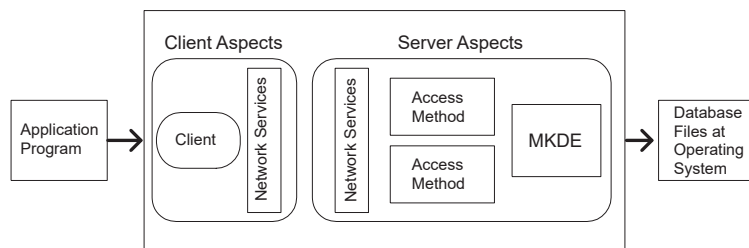
The types of settings that you can configure for server include the following:

- Access
- Communication protocols
- Compatibility with previous versions of the MicroKernel database engine (MKDE)
- Data integrity
- Debugging
- Directories
- Memory usage
- Performance

The types of settings that you can configure for client include the following:

- Access
- Communication protocols
- Performance
- Security
- Application characteristics

You configure these settings within the ZenCC. See Configuration Reference for the tasks and references pertaining to configuration.

Database Security

The access to a Zen database can be protected in several ways. Administrative-level security is set through the operating system. You can control who can administer a Zen database with the security mechanisms native to the operating system.

Zen also provides relational security at the user and group levels. You can control who can access the data and at what capability. For example, for each table within a Zen database, you can specify whether a user or group can create, select, update, insert into, delete, or alter the table.

You establish security by setting a password for the entire database. At that point, the only user authorized to access the database is a default user named Master. You can then add additional users and groups.

Security can be set within the ZenCC. Also supported are two SQL statements pertaining to security: GRANT and REVOKE. These two SQL statements also allow you to set security at both the table and the column level.

The GRANT syntax integrates with transactional owner names, allowing owner names to be enforced when using relational access.

See Zen Security for information pertaining to security, owner names, users, and groups.

Data Archival and Restoration

Backing up data is a routine part of protecting your databases and ensuring disaster recovery. You have several ways in which you can back up and restore your Zen databases.

If your business allows you to stop all applications that access a Zen database, you may use any of the operating system utilities, or third-party software, to backup or restore the database files.

Archival logging is another backup method that you can use to supplement operating system utilities. Archival logging allows you to keep a log of all database operations since your last backup. In case of a system failure, you can restore the data files from backup then roll forward the changes from the log file to return the database to the state it was in prior to the system failure.

Continuous operations allows you to backup database files while the database engine is running and users are connected. After starting Continuous Operations, the database engine closes the active data files and stores all changes in temporary data files (called *delta* files). When the backup is complete, you turn off Continuous Operations. The database engine then reads the delta file and applies all the changes to the original data files.

See Logging, Backup, and Restore for additional information about backing up and restoring databases.

Troubleshooting

Zen User's Guide and *Getting Started with Zen* both contain troubleshooting information. *Getting Started with Zen* contains troubleshooting information for installing the Zen products. The user's guide offers general troubleshooting information as well as a list of frequently asked questions.

Helpful Utilities

Zen comes with a variety of utilities designed to help you control and manage your databases. For a list of the primary utilities, see *Zen User's Guide*. Note that some utilities may be excluded in a custom installation.

The Zen Component Architecture

The following topics cover features designed to offer a trouble-free environment for installing and running critical applications:

- Zen Database Management System
- Relational Architectural Overview
- Error Codes
- Auto Reconnect

Zen Database Management System

The Zen database management system supports data processing applications in two ways:

- The MicroKernel Engine, which provides access to database transactions through the Btrieve API

- The Relational Engine, which provides a SQL relational interface through ODBC

The SQL interface of the Relational Engine uses the Btrieve API to access data files.

The rest of this topic describes the shared features of these two engines, as well as where they differ in configuration and behavior, generally based on the operational environment.

Common Address Space

Zen uses an optimized memory architecture that provides high performance for both transactional and relational data access methods. Both the MicroKernel Engine and the Relational Engine load and operate in the same process address space, minimizing the CPU time to communicate between them.

Row Level Locking

The Btrieve v6 data file format introduced row level locking to improve database engine performance in multiuser environments, where many updates and writes occur at once or where transactions remain open for an extended time.

A transaction locks only the rows that it affects directly, not the entire page. One client can update records on a given page at the same time as another client updates different records on the same page. Waiting is necessary only when a second application attempts to modify the exact same records currently locked by the first application. Row level locking decreases overall wait time and improves performance in a multiuser environment.

Row level locking is implemented for data pages and partially implemented for key pages. It does not apply to variable pages. A small percentage of key page changes may cause key entries to move from one page to another, for example when a key page is split or combined. These changes retain a full page lock until the transaction completes.

MicroKernel Engine

In all installations, the MicroKernel Engine provides Btrieve API support for data files and local applications running on the same system as the engine. In a Zen client-server environment, the

MicroKernel Engine supports both local and remote applications. In a Workgroup environment, the MicroKernel Engine uses a gateway configuration to connect to remote applications and can service requests sent by a remote Workgroup engine.

In client-server environments on Windows, the MicroKernel Engine is by default installed to run as a Windows Service. In a Workgroup environment, it can be installed to run as an application or as a service. If installed as an application, a tray icon is displayed to provide indication that the engine is running. The tray icon does not appear for a server engine or when the Workgroup engine is installed as a service. See also Technical Differences Between Server and Workgroup.

The Zen Btrieve and ODBC APIs support distributed database applications while hiding the details of connecting to a local or remote database engine from those application. Using this architecture, an application can access data that locally while also accessing data on a remote computer. Moreover, a SQL database can be distributed by having data dictionary files (DDFs) serviced by a local MicroKernel Engine and data files (tables) serviced by a remote MicroKernel Engine. Such a SQL database, which is not serviced exclusively by a local MicroKernel Engine, is referred to as a *mixed-access database*.

Mixed-access databases are subject to the following constraints:

* The following features are not supported: referential integrity (RI), bound databases, triggers, distributed transaction atomicity (requires two-phase commit).

* The Relational Engine and the MicroKernel Engine must both be running on the same computer to access DDFs.

* Data files for tables that are involved in referential integrity relationship, or those that have any triggers defined for them, or are in a bound named database, cannot be opened by a remote MicroKernel Engine.

* When opening a file, the Relational Engine does not verify the version of the MicroKernel Engine servicing the request. If an operation that requires v6.30 or higher MicroKernel Engine API support (for example, shared locking) is issued to an engine with a version earlier than v6.30, then an error code is returned. When opening DDFs or when attempting to bind a DDF or data file, the Relational Engine verifies that the local MicroKernel Engine is servicing the request.

Asynchronous I/O

On a Windows server, the MicroKernel Engine uses asynchronous I/O to write pages to disk to improve performance. The engine writes pages to the Windows system cache or its own cache. In turn, Windows signals when the pages are on disk, helping the MicroKernel to perform write operations efficiently.

Read performance is also enhanced when there are many concurrent operations being done in the MicroKernel Engine at the same time, especially if you access your data set on a striped set of disk drives. Each read causes a worker thread to wait until the page is available. With asynchronous I/O, the operating system can pool the work of multiple readers to make the read operations more efficient.

Relational Engine

The Zen Relational Engine provides ODBC support for Zen applications. ODBC client platforms include Windows platforms. Remote ODBC application access to the Relational Engine requires installation of the Zen ODBC Client, which is a specialized ODBC driver that routes client-side ODBC calls to the ODBC communications server over the network.

Some of the features of the Relational Engine include:

- Atomic statements
- Bidirectional cursors (using the ODBC Cursor Library)
- Outer join support
- Updatable views
- ODBC data type support
- Multiple variable length columns in a table

The ODBC communications server performs the following functions:

- Support network communication for ODBC clients
- Route ODBC calls to the server-side ODBC Driver Manager (which routes calls to the engine)

For details on SQL and ODBC, see SQL Overview in *SQL Engine Reference* and DSN Setup and Connection Strings in *ODBC Guide*.

Relational Architectural Overview

The following diagram illustrates the architectural components of the Zen Relational Engine for the server version. The SQL Connection Manager starts and runs in the same process address space as the MicroKernel Engine and the Relational Engine.

Zen Relational Architecture for Servers and Workgroups

The SQL Connection Manager supports up to 2000 simultaneous connections and uses the ODBC Driver Manager to make calls to the Relational Engine, a SQL relational database engine (SRDE), which in turn rests on top of the MicroKernel.

The following figure illustrates the Zen client-server relational architecture. The client talks to the SQL Connection Manager on the server through TCP/IP. This architecture applies to server engines and also to the Workgroup engine when a client DSN is used to connect from the local Workgroup engine to the remote Workgroup engine.
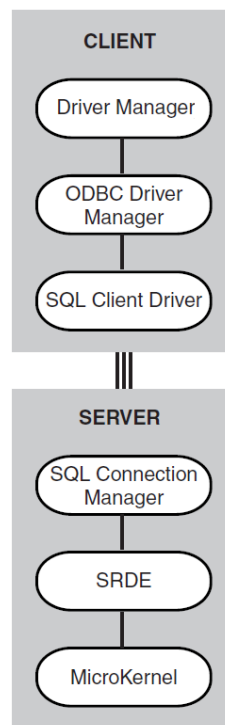
The next figure illustrates the Workgroup relational architecture when a DSN is used to connect from the local Workgroup engine to the remote database, assuming that a remote Workgroup engine is acting as a Gateway to the remote data.

WORKGROUP

ODBC Application

ODBC Driver Manager

SQL Client Driver

SQL Connection Manager

SRDE

MIF

MicroKernel

Common Format Data Storage
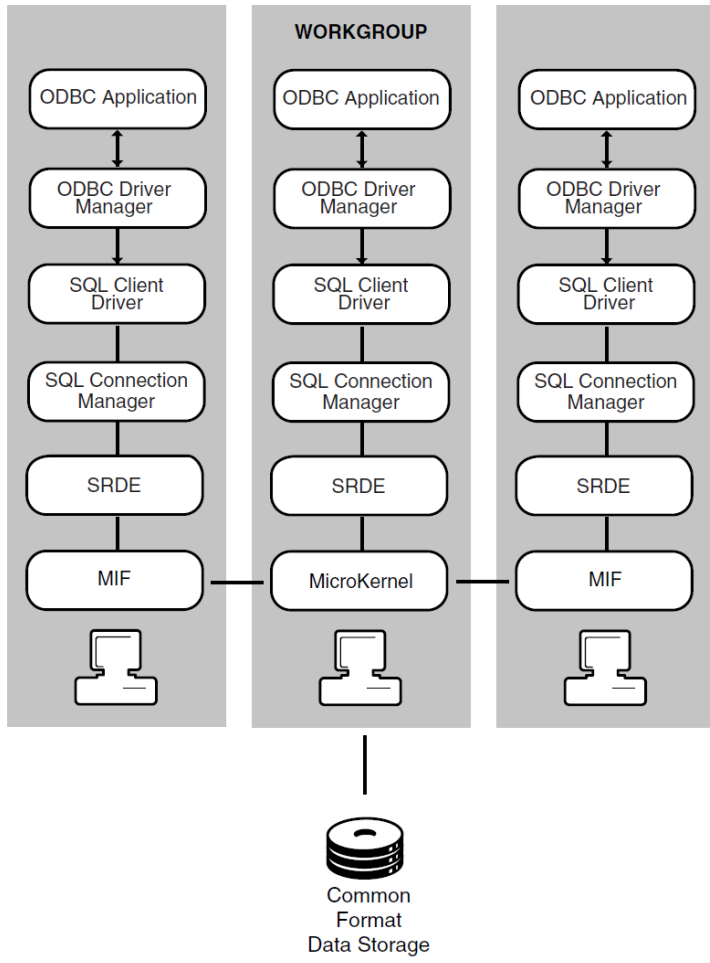
Error Codes

Most Zen top-level components pass through error codes from underlying components so that the actual source of the error is clearly identified to the calling application or in the log file. In situations where an error code may apply to various situations, specific information in the Zen event log should identify the root cause of the error. See Reviewing Message Logs.

Auto Reconnect

The Zen Auto Reconnect capability enables client-server or workgroup applications to endure temporary network interruptions without canceling the current database operation. When Zen detects a network interruption, it automatically attempts to reconnect at specific intervals for a configurable amount of time. This feature also preserves the client context so that when communications are reestablished, database access continues exactly where it left off when the network interruption occurred.

This feature preserves the application context and attempts to reconnect regardless of whether the client or server was attempting to send data at the moment when the network communications were interrupted.

When a network interruption occurs, the reconnect attempts occur at specific intervals. For all connections, successive attempts are made at 0.5, 1, 2, 4, and 8 seconds, continuing every 8 seconds thereafter until the Auto Reconnect Timeout value is reached. If no attempt is successful before the maximum wait time is reached, then the current operation fails and the client connection is reset. The maximum wait time is configurable between 45 seconds and 65,535 seconds.

This feature is disabled by default. For this feature to operate, you must select Enable Auto Reconnect (Windows only) for both client and server configurations. You can specify the time-out value using the server setting Auto Reconnect Timeout.

Remarks

This feature is supported for Btrieve, ODBC, and DTI connections.

The Btrieve communication servers may write out .par or .sar files to the Transaction Log Directory. These are temporary files that contain the context for the last item that the server tried to send to the client. When a reconnection occurs, the client may ask for data to be sent again. The server reads these files to obtain the appropriate data. These files are normally deleted by the server after the data is read or later when the connection is finally terminated.

Configuration Reference

The following topics cover various ways to configure your database server engines and clients:

- Configuration Overview
- Configuration Using ZenCC
- Configuration Using Bcfg
- Service Configuration Properties
- Server Configuration Properties on All Platforms
- Windows Client Configuration Properties
- Linux and Raspbian Client Configuration Properties
- Reporting Engine Configuration Properties

Configuration Overview

Zen is configured using settings in its database engines and clients. You can set configuration properties in Zen Control Center (ZenCC) or at a command prompt.

In ZenCC, the configuration settings are properties of the engine or client. See To set the properties in ZenCC for an engine and To set the properties in ZenCC for a local client.

Configuring any components is optional. If you do not configure them, each component loads with default configuration settings. For best results, you should use only the version of ZenCC that is the same version as your client and engine components.

You can use configuration for the following reasons:

- Your system or your Zen application requires you to adjust the settings. See your application documentation for recommended values. If you are running multiple applications concurrently, add the recommended values together. If you are running multiple applications sequentially, use the highest recommended value.

- You want to optimize the settings so that Zen provides the services you need without using more memory than necessary.

The configuration settings themselves are discussed in Configuration Reference.

Ensuring Configuration Changes Take Effect

Some engine configuration settings require that you restart the database engines after the setting is changed. Restarting the engines ensures that the setting takes effect. Each setting in this chapter lists whether a database engine restart is required. Also, ZenCC informs you with a message if a changed setting requires an engine restart.

The CLI tool also informs you that you changed the setting from the command line rather by using an input file. Use of an input file always requires that you restart the engines. See Configuration Using Bcfg.

To stop and start a server database engine from the command line, see the following topics in *Zen User's Guide*:

- Starting and Stopping the Enterprise Server Engine on a Windows Server

- Starting and Stopping the Database Engine on Linux and Raspbian

To stop and start a Workgroup engine, see Starting and Stopping the Workgroup Engine on Windows in *Zen User's Guide*.

In addition, changing client properties often requires the client to be restarted. To reload the client, simply exit all applications that depend on Zen and restart them.

Connecting to Different Machines

You can configure both local and remote engines as well as local client components; however, each engine must be configured separately. See Configuration Using ZenCC and Configuration Using Bcfg.

When you are connected to a remote machine with ZenCC, you can view and change only engine components. Client components, such as on workgroup and workstation engines and client machines, can be configured only locally on each machine.

Configuration Using ZenCC

In ZenCC, the configuration settings are properties of either an engine or a client. All registered engines appear in Zen Explorer, and you can configure their properties by selecting their nodes, but only the local client is shown. To configure a remote client, locate it in Zen Explorer on the client machine itself.

To configure a remote server, you can open ZenCC on a client to the server or on any server accessible to the network, locate or register the remote server under the Engines node, then configure its properties. This method is useful with operating systems where ZenCC is not supported, such as Windows Nano Server and IoT Core and Raspbian.

The following steps show how to access the properties of an engine and a client.

To set the properties in ZenCC for an engine

1. In Zen Explorer, expand the **Engines** node.

2. Right-click the database engine for which you want to specify configuration settings.

3. Select **Properties**.

4. Click a category of properties to display that group of settings.

To open a help topic on a selected setting, you can press F1 or click the question mark icon at lower left. The general documentation for settings is found under Server Configuration Properties on All Platforms.

To set the properties in ZenCC for a local client

1. In Zen Explorer, expand the **Local Client** node.

2. Right-click **MicroKernel Router**.

3. Select **Properties**.

4. Click the desired option category in the tree to display the settings for that category of options.

To open a help topic on a selected setting, you can press F1 or click the question icon at lower left. The general documentation for property settings is found under the following topics:

*   Windows Client Configuration Properties

*   Linux and Raspbian Client Configuration Properties

*   Reporting Engine Configuration Properties

You may also set properties from the command line in Windows systems. This method is also normally used on Linux and Raspbian systems. See Configuration Using Bcfg.

Configuration Using Bcfg

The **bcfg** command line tool provides the same settings as property dialogs in ZenCC. The tool runs on Windows, Linux, and Raspbian platforms supported by Zen. Its executable program is **bcfg.exe** on Windows and **bcfg** on Linux-based systems. On Windows systems, **bcfg** is installed in the bin directory of the Zen installation. On Linux systems, it is in /usr/local/actianzen/bin.

The following topics cover use of **bcfg**:

- Command Syntax
- Example Scenario: Configuring a Single Setting from a Command Prompt
- Editing an Input File
- Restarting the Engines After Applying a New Setting
- Troubleshooting

Command Syntax

Bcfg offers several of commands for managing configuration settings:

- To read the value of a single setting

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

- To change the value of a single setting

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

- To write an output file of all current settings (can be edited and then used for input)

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

- To read an input file of new settings and apply them (text input only)

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

- To search for the settings by keyword

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

Options

-E   Ignore errors when reading *inputfile* or writing to *outputfile*.

-F   Force overwrite of *outputfile* without prompting. Without this flag, the combination of a preexisting *outputfile* and a lack of stdin, such as found in remote PowerShell sessions, causes **bcfg** to fail with the message "Interactive input failure detected."

-H   The help search option for a *keyword*. The tool searches for configuration settings that contain *keyword* and returns the ID and setting name, or returns "no matches found." See also the next item in this list.

-I   Used to read tool input from a file. The file is named in the *inputfile* parameter.

ID   A two- or three-digit integer that identifies a configuration setting. Some configuration settings require that you restart the database engines for them to take effect. **Bcfg** prompts you if a restart is required. See Restarting the Engines After Applying a New Setting.

inputfile   A text file that contains one or more configuration setting records for a specified server and the value assigned to each setting. Used with the -I option. A convenient way to create an input file is first to create an output file. You can then edit the setting values as required and use the edited version as an input file. See Editing an Input File for the types of edits allowed.

-JSON   Optional parameter to use JSON format instead of text for output, including error messages.

keyword   The name of the configuration setting, such as "allow client-stored credentials" or "supported protocols".

      Keyword is not case-sensitive. If you use more than one keyword with spaces, double-quote the string.

      The tool can provide help based on partial keywords. For example, if you specify `-H client`, the tool returns all settings with the word *client* as part of the setting name. If you specify `-H a`, the tool returns all settings with an "a" in the name.

-O   Used to write tool output to a file. The file is named in the *outputfile* parameter.

outputfile   A text or JSON file where one or more configuration settings for a specified server are written. Used with the -O option.

-P   Required if a password is required to access *server*. Provided in the *password* parameter.

password   Password used with *username* to connect to the *server*. Used with the -P option. See *username*. See also Zen Security.

-S   Required if the configuration settings apply to a remote server (a server other than the local one). Provided in the *server* parameter.

server	The name or IP address of the remote server that contains the database engine. Used with the -S option.
-U	Required if a user name is required to access *server*. Provided in the *username* parameter.
username	User name with which you will connect to *server*. See also Zen Security. Used with the -U option.
	If *server* is a local machine, the *username* and *password* are not required if the following are true:
	• You are logged in to the local machine as administrator or a member of the Zen_Admin group.
	• The local machine is not running Terminal Services.
value	A value assigned to the configuration setting. The valid values are given in a configuration setting in Options or Range.
	If *value* is omitted, the tool returns the current setting.
	If *value* is included, the tool changes the setting to the value.
	See Example Scenario: Configuring a Single Setting from a Command Prompt.

Example Scenario: Configuring a Single Setting from a Command Prompt

Suppose that you want to turn on a configuration setting having to do with reconnecting a client to the server in the event of a network outage. You are not certain about the name of the configuration setting. The steps in the following example show how to find the setting and configure it at a command prompt.

1. At a prompt, enter `bcfg -H reconnect` then press **Enter**.

   The tool reports all settings that contain the string `reconnect`:

   ```
   ID                Setting Name
   -----------------------------
   29                Enable Auto Reconnect
   148               Enable Auto Reconnect
   149               Auto Reconnect Timeout
   ```

   Two of the settings, 29 and 148, look like what you want, but which is which?

2. Enter `bcfg 29` then press **Enter**.

   The tool reports the following for setting ID 29:

   ```
   ====================
   ENABLE AUTO RECONNECT
   ```

```
======================
ID: 29
Value: Off
Options: On     Off
Default Value: Off
Description: <Client setting> Specifies if the Client will attempt to reconnect to
the Server in the event of a network outage. The reconnected Client will continue
processing as if no errors were encountered.
```

The description tells you that this setting applies to a client and that the setting is currently off.

3. Enter `bcfg 29 on` then press **Enter**.

    The tool informs you that system setting 29 has been updated.

4. If you want to verify that the setting is now on, enter `bcfg 29` then press **Enter**.

    The tool reports the following for setting ID 29, showing that the value is now set to on:

    ```
    ======================

    ENABLE AUTO RECONNECT

    ======================

    ID: 29

    Value: On

    Options: On     Off

    Default Value: Off

    Description: <Client setting> Specifies if the Client will attempt to reconnect to
    the Server in the event of a network outage. The reconnected Client will continue
    processing as if no errors were encountered.
    ```

Editing an Input File

Instead of configuring settings one at a time from the command line, you can provide one or more settings in an input file. In the current release, only text is supported for this purpose.

A convenient way to create an input file is to edit an output file and use it as an input file. The input file can contain all settings or only the ones you need to change. It must contain at least one complete record for one setting. If you create an input file from an output file and remove settings, be sure that the ones remaining are complete records.

You can change configuration values in the lines for ID, Value, Options, or Range. Changes to other lines may result in errors, failed changes to settings, or other unexpected behavior.

At a minimum, a complete record includes an ID and value pair. We recommend that you include all lines from the header through the description. For example, in the last step of the example scenario under this topic, the setting for Enable Auto Reconnect is complete and could be used in an input file.

Restarting the Engines After Applying a New Setting

If you use an input file, **bcfg** prompts you to restart the database engines to ensure that the configuration settings take effect, no matter what settings were included in the file.

If you configure a setting from the command line, **bcfg** prompts you to restart the database engines only if the setting requires a restart.

See Ensuring Configuration Changes Take Effect for steps to restart the engines.

Troubleshooting

If you have difficulty running the tool, the following table gives suggestions for troubleshooting.

Troubleshooting Condition	Discussion
You receive the error: Unable to connect to database engine. Make sure the target machine is accessible and an engine is running on the target machine.	This error occurs when you attempt to configure the local server. To configure the local server, you must be a member of the **zen-data** group or be the root user. See also Zen Account Management on Linux and Raspbian in *Getting Started with Zen*.
You receive the error: Interactive input failure detected. Exiting without overwriting file.	When you run the command, add the **-F** option to suppress the prompt, make no attempt to read from stdin, and always overwrite the file.

Service Configuration Properties

On Windows server environments, Zen servers run as a service. The service is loaded as part of the installation process and is set to be always available if you applied the complete installation.

You can configure the startup policy for the service from within ZenCC.

To use ZenCC to set service startup policy

1. In ZenCC expand the **Services** node.

2. Right-click **Actian Zen Server Engine**.

3. Click **Properties**.

4. For **Startup type**, select a startup policy.

Policy	Meaning
Manual	The service is not started automatically when the operating system starts. You must manually start the service after the operating system starts or restarts.
Automatic	The service is started automatically when the operating system starts or restarts.
Disabled	The service is rendered inoperative until you reset the startup policy to Manual or Automatic.

5. Click **OK**.

Server Configuration Properties on All Platforms

Each Zen database engine has its own server configuration. This topic describes the options for each engine, which are independent of those for other engines.

You can configure Zen servers on Windows, Linux, and Raspbian platforms using ZenCC or the command line tool **bcfg**. For ZenCC, open the properties window by right-clicking a database engine node. For the command line, see Configuration Using Bcfg.

The following table lists all server configuration properties and their settings in alphabetical order. The settings are linked to topics with more information.

Configuration Option	Setting Name
Access	Accept Remote Request
	Allow Cache Engine Connections
	Allow Client-Stored Credentials
	Authentication (Linux-Based Engines Only)
	Configuration File (Linux and Raspbian Engines Only)
	Prompt for Client Credentials
	Storage Server
	Wire Encryption
	Wire Encryption Level
Communication Protocols	Auto Reconnect Timeout
	Enable Auto Reconnect (Windows only)
	Listen IP Address
	Supported Protocols
	TCP/IP Multihomed
	TCP/IP Port
Compatibility	Limit File Size
	Create File Version
	System Data

Configuration Option	Setting Name
Data Integrity	Archival Logging Selected Files
	Initiation Time Limit
	Operation Bundle Limit
	Transaction Durability
	Transaction Logging
	Wait Lock Timeout
Debugging	Number of Bytes from Data Buffer
	Number of Bytes from Key Buffer
	Select Operations
	Trace File Location
	Trace Operation
Directories	Temporary File Location
	Transaction Log Directory
	Working Directory
	DBNames Configuration Location
	TEMPDB Directory (Client Reporting Engine only)
Information	Server name (display-only)
	Engine version (display-only)
	Engine type (display-only)
Memory Usage	Allocate Resources at Startup
	Back to Minimal State if Inactive
	Minimal State Delay
	Sort Buffer Size
	System Cache

Configuration Option	Setting Name
Performance Tuning	Automatic Defragmentation
	Cache Allocation Size
	Communications Threads
	File Close Delay
	File Growth Factor
	Index Balancing
	Limit Segment Size to 2 GB
	Transaction Log Buffer Size
	Max MicroKernel Memory Usage
	Number of Input/Output Threads
	Transaction Log Size

Access

Right-click a server and select Properties > Access to see the following configuration settings:

- Accept Remote Request
- Allow Cache Engine Connections
- Allow Client-Stored Credentials
- Authentication (Linux-Based Engines Only)
- Configuration File (Linux and Raspbian Engines Only)
- Prompt for Client Credentials
- Wire Encryption
- Wire Encryption Level

Accept Remote Request

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Yes

This setting specifies whether the Communications Manager accepts requests from remote servers and client workstations. If you turn this option to **On**, the Communications Manager advertises its presence on the network.

Allow Cache Engine Connections

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Yes

Specifies if the server will support clients that will attempt to connect to the server with the Cache engine. When set to **Off**, clients will still connect to the server but will not use the Cache engine.

Allow Client-Stored Credentials

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	No

When this setting is **On**, the database engine accepts user credentials stored on the client. The method and location of storage depends on the operating system of the client:

- Windows clients: these credentials are stored in the Windows registry. When the Prompt for Client Credentials is set to **On**, then a popup dialog allows you to save the credentials by selecting the **Save User name and Password** check box. Alternatively, you can use the **pvnetpass** command line tool to manage stored credentials.

- Linux and Raspbian clients: Credentials are stored in the Zen registry by **pvnetpass**.

When this setting is **Off**, the database engine forces the client to omit stored credentials from any database operation that requires credentials. Such credentials must be supplied by the application or through the login dialog. The login dialog still writes client-stored credentials if specified using the login dialog, even if this setting is Off. However, they will not be accepted.

When client-stored credentials are allowed, anyone can sit at that particular client computer and log in to the database using the stored credentials without knowing those credentials. This behavior can be convenient for environments in which strict authentication of individual users is not a concern, such as a physically secured work area where all users have the same level of access permissions. On the other hand, in environments where unauthorized personnel are present or authorized users have varying levels of access permissions, this setting must be **Off**.

See also Prompt for Client Credentials.

Summary Chart of Login Behavior

Prompt for Credentials	Allow Client-Stored Credentials	Behavior
Off	Off	Zen client does not prompt the user or use stored credentials, thus credentials must be supplied by the client application during a Btrieve operation.
Off	On	If credentials are not supplied by the client application during the Btrieve operation, the client uses credentials stored by the login dialog or by pvnetpass, if such credentials are available. If no credentials are supplied by either method, the connection attempt fails. No login dialog is displayed.
On	Off	If credentials are not supplied by the client application during the Btrieve operation, the client displays a login dialog to the user, and the Linux or Raspbian client returns a status code for permissions error. Credentials stored by the login dialog or by pvnetpass are not used.
On	On	If credentials are not supplied by the client application during the Btrieve operation, stored credentials are used. If no stored credentials are available, then the client displays a login dialog to the user, and the Linux or Raspbian client returns a status code for permissions error.

Authentication (Linux-Based Engines Only)

Type	Range	Default	Units	Requires Engine Restart
SelectOne	Three options. See below	Emulate Workgroup Engine	None	Yes

The following options set which type of authentication to use for access to the server engine:

- **Emulate Workgroup Engine**. Use this value when Samba is used to authenticate user access on the system. If you want to bypass security provided by the operating system and do not want to store RTSS passwords in the registry, use **Emulate Workgroup Engine**.

- **Proprietary Authentication (using btpasswd)**. Use this value when not using Samba for Linux to authenticate and the user has no account on the server. This option allows you to maintain a separate password file for connecting to the Linux or Raspbian system.

  If you are using BTPASSWD authentication on your server, user names and passwords must be set from clients connecting to this server. Use Zen Control Center or **pvnetpass** at a command prompt. See Groups, Users, and Security and pvnetpass, both topics in *Zen User's Guide*.

  Use Proprietary Authentication if stronger security is needed for the server and you want user names and passwords to be different from any user authentication scheme employed on the server.

- **Standard Linux Authentication**. Use this value when not using Samba to authenticate but users have accounts on the Linux or Raspbian system.

  Standard Linux authentication is used with PAM. Use PAM if you want to use existing user names and passwords on the Linux or Raspbian server. You can specify user names and passwords from the client using **pvnetpass**. PAM is also very flexible and offers many custom modules for Linux and Raspbian. Check the PAM website for more information.

  If the Zen installation detects PAM, the installation completes its configuration so that PAM can be used. If you install PAM after installing Zen and want to use standard authentication with PAM, you must reinstall Zen. The reason is that the PAM installation copies files, creates configuration files, sets permissions, and creates links. Zen needs to be reinstalled to detect PAM and correctly complete its PAM configuration.

  You reinstall Zen by uninstalling and then installing again. See Installing Zen for Linux-based Systems in *Getting Started with Zen* for the steps to do this.

Samba and Authentication Using PVPIPE$ (Linux Only)

You may use Samba, if available, in addition to any of the three authentication methods described above. If PVPIPE$ is shared as described under Configuration File (Linux and Raspbian Engines Only), the Zen engine creates the FIFO in `$ACTIANZEN_ROOT/etc/pipe/mkde.pip`. PVPIPE$ is supported only on Linux.

Note: The trailing dollar sign ($) means this share will be hidden. The Zen client components automatically take care of accessing this pipe as `\\<server>\PVPIPE$\mkde.pip` (case-insensitive). You do not need to perform any explicit actions or modify your application to access

this pipe. The only exception to this is if you are troubleshooting your Samba or Zen configurations.

When a client connects to the remote engine and discovers the engine returns Unix in the version block, it first looks in the registry (RTSS setting) for authentication information. If it finds no user name and password there, the client connects to the above pipe and receives client authentication information from the server, which will be validated later.

To be authenticated, you must be able to connect to the share and read the pipe. This is one way of specifying who can use the engine and who cannot. The easiest way to do this is to set the valid users value in the smb.conf configuration file. If the client is unable to get authentication, status 3119 is returned.

Caution! By allowing a client read access to PVPIPE$, that client is authorized to access the engine remotely.

A simple way to ensure the client gets authentication is to enter `\\<yourserver>\pvpipe$\mkde.pip` at a command prompt. You should see a lot of question marks (unprintable symbols), occasional printable characters, and hear beeps. If you do not, check your Samba configuration file to be sure you have rights to read this pipe. If you do but still see error 94 or 3119, validate your RTSS setting using the engine configuration properties in Zen Control Center or with pvnetpass.

To learn more about access to files shared through Samba, read the Samba documentation.

Configuration File (Linux and Raspbian Engines Only)

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	/etc/smb.conf	None	Yes

This setting gives the location of the smb.conf file used to export local file systems to Windows clients. The engine requires this file to translate UNC paths on remote systems into local calls to the correct database file. Note that on Raspbian systems where native SMB file sharing is used instead of a third-party Samba package, this setting does not apply.

The default value is **/etc/smb.conf**. If you installed the Samba configuration file in a different location, enter the correct path name.

Zen checks for the smb.conf configuration file in the following locations, in this order:

- /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

The first smb.conf found is the one used. If no smb.conf is found, Zen logs an entry in the system log file, and no Samba sharing is enabled.

On Linux, if you want to use the PVPIPE$ FIFO share, and it is not already present in the smb.conf file, you must be set in the file as follows. Note that Zen server installation creates the zen-svc user and zen-data group.

```
[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
```

To enable the user named **zen-svc** to access this share, you must use the **smbpasswd** command with certain parameters. See the Samba documentation for information about the -n option. When you are ready to enable access, do the following on the system where Zen Enterprise Server or Cloud Server is installed:

1. Log in as root.

2. Run the following command to add user zen-svc to the local smbpasswd file:

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

3. Restart smbd for the changes to take effect.

Prompt for Client Credentials

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	No

This setting determines whether the Windows Zen client prompts the user for login credentials if no other credentials are available during a database operation that requires user authentication.

When this setting is **On**, in the absence of other authentication credentials, the engine requires the Windows client to present a login dialog to the user. This setting applies only when Mixed or Database security is in effect and does not apply to a Linux or Raspbian client under any circumstances. If valid credentials are supplied via another method (for example, explicit Btrieve Login (78) operation or credentials stored on the client), the login dialog does not appear.

If no database context is specified to the engine within the operation requiring user credentials, the engine assumes the user is attempting to log in to the current database.

When this setting is **Off** and one of the new security models is in use, user credentials must be provided programmatically (credentials stored on the client or provided with a Btrieve Login (78), Open (0), or Create (14) operation), or else the login attempt fails with an authentication error.

See Also

Allow Client-Stored Credentials

Storage Server

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	blank	None	Yes

This property gives the name of the Zen server that the Client Reporting Engine is supporting. It is set on the system where the Client Reporting Engine is installed. On Windows, the string is not case-sensitive. This setting is provided in Zen only for Client Reporting Engine.

Wire Encryption

Type	Range	Default	Units	Requires Engine Restart
Single select	Never If Needed Always	If Needed	None	Yes

This property specifies whether the given client or server should use encryption for its network communications. The default value of **If Needed** means that the client or server only uses encryption if the other end of the communication stream requires it. For example, assume that Server A has its **Wire Encryption** value set to **Always**. Server B has its value set to **Never**. Your client has its value set to **If Needed**. In this case, the client will use encryption when communicating with Server A, but it will not use encryption when communicating with Server B.

The following table summarizes behavior for each combination of client and server values.

Client Setting	Server Setting "Never"	Server Setting "Always"	Server Setting "If Needed"
Never	Encryption not used	Status Code 5001	Encryption not used
Always	Status Code 5000	Encryption used. Level determined by highest Wire Encryption Level setting between client and server	Encryption used. Level determined by client Wire Encryption Level setting.
If Needed	Encryption not used	Encryption used. Level determined by server's Wire Encryption Level setting	Encryption not used

Wire Encryption Level

Type	Range	Default	Units	Requires Engine Restart
Single select	Low Medium High	Medium	None	Yes

This setting specifies the strength of the encryption key that should be used for encrypted communications. The following table shows the levels available.

Value	Meaning
Low	40-bit encryption key used
Medium	56-bit encryption key used
High	128-bit encryption key used

Encryption using a key 128 bits long is generally accepted as strong encryption. The other settings provide progressively less protection but higher performance, in the event that you require some level of encryption but are willing to accept a lower level of deterrence to gain better performance.

When a client and a server both require encryption and one specifies a stronger encryption level than the other, the two entities use the stronger level to communicate.

Communication Protocols

Communication Protocols contains the following configuration settings:

- Auto Reconnect Timeout
- Enable Auto Reconnect (Windows only)
- Listen IP Address
- Supported Protocols
- TCP/IP Multihomed
- TCP/IP Port

Auto Reconnect Timeout

Type	Range	Default	Units	Requires Engine Restart
Numeric	45–65535	180	seconds	Yes

This setting specifies how long the client will attempt to connect to the server before giving up. When an Auto Reconnect-enabled client first connects to a Auto Reconnect-enabled server, the server communicates this value to the client so that both components know how long to attempt to reconnect in the event of a network interruption.

Enable Auto Reconnect (Windows only)

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Yes

This setting specifies whether you want the server to support clients attempting to automatically reconnect during a network outage. A setting of **On** means Auto Reconnect is enabled.

Auto Reconnect is not in effect for a given client connection unless this setting is also enabled in that client's configuration.

To specify how long a client will attempt to reconnect to the server before giving up, see Auto Reconnect Timeout, above.

Listen IP Address

Type	Range	Default	Units	Requires Engine Restart
String	Valid IP address or multiple addresses separated by a comma between each address	0.0.0.0	None	Yes

This option specifies the IP address or addresses the database engine listens on when TCP/IP Multihomed is **Off**. This option is ignored when TCP/IP Multihomed is **On**.

Multiple IP addresses may be specified but must be separated by a comma between each address. The string can be a combination of IPv4 and IPv6 addresses. Any of the IPv6 address formats supported by Zen can be used. See Drive-based Formats in *Getting Started with Zen*.

Supported Protocols

Type	Range	Default	Units	Requires Engine Restart
Multiple select	One or more	TCP/IP	None	Yes

This setting specifies the protocols by which the database engine listens for client connections. If more than one protocol is selected, the engine begins the session by listening on all of them. The first protocol that succeeds is used for the remainder of the session. You must have at least one protocol enabled on both the client and the server or they cannot communicate.

Note: The default is TCP/IP, which is the only option currently supported.

Linux and Raspbian

TCP/IP is the only supported protocol for Zen on Linux and Raspbian. Therefore, the Supported Protocols setting is not available for those environments.

TCP/IP Multihomed

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Yes

This option specifies whether the database engine should listen for client connections on all network interfaces. If it is set to **On**, the database engine listens on all network interfaces, and the IP addresses listed in the Listen IP Address option is ignored. If this setting is **Off**, you must specify in Listen IP Address which addresses) for the database engine to use for client communications.

TCP/IP Port

Type	Range	Default	Units	Requires Engine Restart
Numeric	256–65535	1583	None	Yes

This setting configures the port number used by the Relational Engine.

This port number must be the same as that defined in any Client DSNs pointing to this server. For information on how to change the port number in a Client DSN, see Advanced Connection Attributes in *ODBC Guide*.

For additional information on ports, see Changing the Default Communication Ports in *Getting Started with Zen*.

Compatibility

Compatibility contains the following configuration settings:

- Limit File Size
- Create File Version
- System Data

Limit File Size

This check box can be selected so that files version 13.0 or later are prevented from exceeding the version 9.5 maximum file size of 256 GB.

Create File Version

Type	Range	Default	Units	Requires Engine Restart
SelectOne	6.x, 7.x, 8.x, 9.0, 9.5, 13.0, 16.0	9.5	None	No

This setting specifies the format used to create new files, but it does not affect files already created. Existing files from version 6.x onward can be read from and written to without changing their file format.

More recent Zen file formats offer more features. We recommend that you choose 6.x, 7.x, or 8.x only if you need backward compatibility with a previous MicroKernel version.

Note: Dictionary files (DDFs) must be created with a file format of 6.x or later. The **New Database** wizard uses the setting for create file version. The data files can be in any of the previous file formats supported. Only the DDFs must use a file format of 6.x or later.

System Data

Type	Range	Default	Units	Requires Engine Restart
Single select	See below	If needed	None	No

System data refers to a hidden unique key in each record. Because the MicroKernel relies on uniquely identifying rows in order to ensure transaction durability, a file must either have a unique key defined or have system data included in the file. The default value is **If needed**; the available options are:

- **None**. By default, system data is not included on file creation. Application developers using the Create operation can override this setting.

- **If needed**. System data is added to the file on file creation if the file does not have a unique key.

- **Always**. System data is always added on file creation, regardless of whether the file has a unique key.

Note: A change to the System Data setting does not affect existing files. This setting only affects how new files are created.

If you want to use transaction durability with a file that was not created with System Data turned on and does not have a unique key, you must recreate the file after setting System Data to **Yes** or **If needed**.

The Relational Engine always creates files with system data. This information applies to files created through SQL, JDBC, or any method other than the Btrieve API.

This setting introduced standard Zen system data for logging. It does not currently create system data v2.

Even if a file has a unique key, you may want to include system data, because users can drop indexes.

Data Integrity

Data Integrity contains the following configuration settings:

- Archival Logging Selected Files
- Initiation Time Limit

- Operation Bundle Limit

- Transaction Durability

- Transaction Logging

- Wait Lock Timeout

Archival Logging Selected Files

Type	Range	Default	Units	Requires Engine Restart
Boolean	On, Off	Off	None	Yes

This setting controls whether the MicroKernel performs archival logging, which can facilitate your file backup activities. If a system failure occurs, you can use the archival log files and the **butil -rollfwd** command to recover changes made to a file between the time of the last backup and a system failure.

To direct the MicroKernel to perform archival logging, you must specify the files for which the MicroKernel is to perform archival logging by adding entries to an archival log configuration file that you create on the volume that contains the files. For more information about archival logging, refer to Understanding Archival Logging and Continuous Operations.

Initiation Time Limit

Type	Range	Default	Units	Requires Engine Restart
Numeric	1–1800000	10000	milliseconds	No

This setting specifies the time limit that triggers a system transaction. The MicroKernel initiates a system transaction when it reaches the Operation Bundle Limit or the time limit, whichever comes first, or when it needs to reuse cache.

Operation Bundle Limit

Type	Range	Default	Units	Requires Engine Restart
Numeric	1–65535	65535	None	No

This option specifies the maximum number of operations (performed on any one file) required to trigger a system transaction. The MicroKernel initiates a system transaction when it reaches the bundle limit or the Initiation Time Limit, whichever comes first, or when it needs to reuse cache.

The MicroKernel Database Engine treats each user transaction (starting with Begin Transaction until End Transaction or Abort Transaction) as one operation. For example, if there are 100 Btrieve operations between the Begin Transaction and the End Transaction operation, then all the 102 Btrieve operations together are treated as a single operation.

Transaction Durability

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Yes

Transaction Durability is the same as Transaction Logging except that Transaction Durability guarantees that all successfully completed transactions are committed to the data files in the event of a system crash.

For a full discussion of transaction logging and durability, see Transaction Logging and Durability.

Note: When you turn Transaction Durability on, some files may not be able to support the feature. A file must contain at least one unique key, or when it was created, the configuration setting System Data must have been set to "Yes" or "If Needed." Otherwise, any changes to the file are not written to the transaction log. For more information about transaction durability and system data, see *Zen Programmer's Guide*.

Because a change to the System Data setting does not affect existing files, you may need to recreate files that do not have a unique key and were not created with System Data turned on. Be sure to turn on System Data before recreating these files.

Caution! Gateway locator files allow different engines to manage files in different directories on the same file server. If your database contains data files in different directories, you must be sure

that the same database engine manages all the data files in the database. If you have more than one database engine managing files within the same database, database integrity and transaction atomicity are not guaranteed. For more information on how to avoid this potential problem, see Redirecting Locator Files.

Related Settings

See more information on similar and related settings under **Transaction Logging**.

Transaction Logging

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Yes

This setting controls whether the MicroKernel ensures atomicity of transactions by logging all operations that affect the data files.

If the related setting, Transaction Durability, is turned on, then logging takes place automatically, and the **Transaction Logging** setting is ignored.

For a full discussion of transaction logging and durability, see Transaction Logging and Durability.

Note: When you turn Transaction Logging on, some files may not be able to support the feature. A file must contain at least one unique key, or when it was created, the configuration setting System Data must have been set to "Yes" or "If Needed." Otherwise, any changes to the file are not written to the transaction log. For more information about transaction durability and system data, see *Zen Programmer's Guide*.

Because a change to the System Data setting does not affect existing files, you may need to recreate files that do not have a unique key and were not created with System Data turned on. Be sure to turn on System Data before recreating these files.

Caution! Do not turn off Transaction Logging unless your database does not require transaction atomicity among data files. Database integrity for multifile databases cannot be guaranteed if Transaction Logging is turned off.

Do not turn off Transaction Logging unless doing so is supported by your application vendor.

Related Settings

The server configuration setting **Transaction Durability** is similar to Transaction Logging, but provides a higher level of data safety along with a lower level of performance. The server configuration settings **Transaction Log Buffer Size** and **Transaction Log Size** are related to Transaction Logging. Transaction Log Buffer Size allows you to configure the balance between transaction recoverability and performance. The larger the log buffer, the fewer times it is written to disk, and thus the greater the performance. However, database changes that are in the log buffer are not durable through a system failure.

Transaction Log Size controls how large each log file segment gets before a new segment is started.

Note that all of these settings are ignored if Btrieve or SQL transactions are not being used.

Wait Lock Timeout

Type	Range	Default	Units	Requires Engine Restart
Numeric	0–2147483647	30000	milliseconds	Yes

The database engine and its clients use a coordinated retry mechanism when a record lock conflict occurs. If the engine cannot obtain a lock on every requested record within the duration for wait lock timeout, the engine returns control to the application with an appropriate status code.

Wait Lock Timeout Benefits

Wait lock timeout provides the following benefits if a lock conflict occurs:

- Allows the database engine to continue processing requests while waiting for the lock to release.

- Improves thread queuing if multiple threads are waiting for the locked resource.

- Improves network performance by reducing network traffic.

When Wait Lock Timeouts Apply

Wait lock timeouts apply to only two kinds of applications:

- Any application that uses the Relational Engine

- Btrieve applications performing a change operation that does not need to be retried. Such applications receive a lock error within either one second or the wait lock timeout value, whichever is less.

Wait lock timeouts do not usually apply to Btrieve applications that use the MicroKernel Engine through a Zen client on Windows, Linux, or Raspbian. Instead, such applications do one of the following:

- Receive a page or lock conflict error immediately for read operations with "no wait" lock bias (200, 400) or a write operation to which a "no write wait" lock bias (500) has been applied.

- Receive a page or lock conflict error within the lesser of either 1 second or the wait lock timeout value for nontransactional write operations where a "no write wait" lock bias (500) has **not** been applied.

- Wait indefinitely if the operation involved is a read operation with a "wait" lock bias (100, 300).

- Wait indefinitely if the operation involved is a write operation inside of a transaction and a "no write wait" lock bias (500) has **not** been applied to either the operation or the transaction.

On receiving a page or lock conflict error, the application may determine how to handle the conflict by retrying, waiting, or other options.

Handling of Page Locks

The MicroKernel Engine API provides controls to handle record lock situations. In brief, here are the control mechanisms:

- Explicit record locks – You can add lock biases (100, 200, 300, or 400) to read operations when reading records to specify whether to wait. You may also apply these biases to the Begin Transaction operation.

- Implicit page locks during a transaction – Most page lock conflicts are avoided because of row-level locking, but you can add lock bias 500 to your transactions to avoid waiting when a page lock occurs.

- Exclusive file locks – Use concurrent transactions to avoid explicit file locks. If a file cannot be opened exclusively, the request always returns immediately.

For more information about transaction locking, see *Zen Programmer's Guide*, as well as lock bias steps for various operations in *Btrieve API Guide*.

Debugging

Debugging contains the following configuration settings:

- Number of Bytes from Data Buffer
- Number of Bytes from Key Buffer

- Query Log Fetches

- Query Logging

- Select Operations

- Trace File Location

- Trace Operation

Number of Bytes from Data Buffer

Type	Range	Default	Units	Requires Engine Restart
Numeric	0–65535	128	bytes	No

This setting specifies the size of the data buffer that the MicroKernel writes to the trace file. The Trace Operation setting must be set to **On** to use this setting. The size you specify depends on the nature of your tracing needs, whether you need to see the entire data buffer contents or just enough of the buffer contents to identify a record.

Number of Bytes from Key Buffer

Type	Range	Default	Units	Requires Engine Restart
Numeric	0–255	128	bytes	No

This setting specifies the size of the key buffer that the writes to the trace file. The Trace Operation setting must be set to **On** to use this setting. The size you specify depends on the nature of your tracing needs, whether you need to see the entire key buffer contents or just enough of the buffer contents to identify a key.

Query Log Fetches

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	No

This setting determines whether executions of FETCH statements are included query log file. It is a supplement to the Query Logging setting, which must be enabled before enabling this setting.

Query Logging

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	No

This setting enables or disables the logging of SQL queries.

Select Operations

Type	Range	Default	Units	Requires Engine Restart
Multiselect	See below	All	None	No

The **Selected** list displays the available Btrieve API operation codes that are traced. Select from the list the desired operations to trace.

- 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 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)
- 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)
- Step Next (24)
- Step Next Extended (38)
- Step Previous
- Stop (25)
- Step Previous Extended (39)
- Unlock (27)
- Update (3)
- Update Chunk (53)
- Version (26)

Trace File Location

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	*file_path*\Zen\bin\mkde.tra	None	No

This setting specifies the trace file to which the MicroKernel writes trace information. The file name must include a drive or volume specification and path or use a UNC path. If you do not want the trace file in the default location, enter a different path or file name.

For default locations of Zen files, see Where are the files installed? in *Getting Started with Zen*.

Note: Do not use the same trace file name for ODBC tracing and MicroKernel tracing.

Trace Operation

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	No

This setting enables or disables the trace feature, which allows you to trace each Btrieve API call and save the results to a file. Developers can use tracing to debug applications. The MicroKernel writes to the trace file using forced write mode, which ensures that data gets written to the file even if the MicroKernel unloads abnormally. The MicroKernel's performance can be severely impacted, depending on the frequency of incoming requests. If you enable this option, you must specify a Trace File.

Note: You do not need to restart the engine in order to start and stop tracing. You can turn tracing on or off during runtime and apply the changes directly to the engine. If you receive a message from Configuration indicating that you must restart the engine for changes to take effect, you may safely ignore the message for this setting.

Directories

Directories contains the following configuration settings:

- Temporary File Location
- Transaction Log Directory
- Working Directory

- DBNames Configuration Location

- TEMPDB Directory (Client Reporting Engine only)

Temporary File Location

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	Varies	None	No

This property sets the directory where Zen can write a temporary file while processing a query. For more information, see Temporary Files in *SQL Engine Reference*.

Transaction Log Directory

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	Varies	None	Yes

This setting specifies the location the MicroKernel uses to store the transaction log. It must be a valid path and include a drive or volume specification or UNC path. Default varies with the operating system.

The engine ignores this setting unless Transaction Durability or Transaction Logging is turned on.

Caution! Do not use the same directory for multiple database engines. For example, it may seem convenient to set a remote server directory as the transaction log directory for more than one engine. However, the engines will be unable to determine which transaction log segments belong to which engine if it becomes necessary to do a log roll forward

If your database engine sees heavy use, you should configure your system to maintain the transaction logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the log writes and data file writes are split across different drives instead of competing for I/O bandwidth on a single drive. For a full discussion of transaction logging, see Transaction Logging and Durability.

Working Directory

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	Same directory as data file	None	Yes

This setting specifies the location of the MicroKernel working directory, which is used to store temporary files in operations such as building large indexes. If disk space is limited on certain volumes, you can use this option to specify a working directory on a volume with adequate space.

There is no default value specified, but if you do not set a working directory, then the default is the location of the data file. To specify a fixed working directory, enter a path in the **Value** text box. The path must include a drive or volume specification or a UNC path.

DBNames Configuration Location

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	Varies	None	Yes

This value sets the path to an alternate location for the DBNames configuration file.

For Zen server engines, this is a local file path, not a directory path. For Workgroup engines, it can be a remote path accessible to the Workgroup MicroKernel. The default value varies by operating system.

- Windows platforms: *application_data_directory*

- Linux or Raspbian server: /usr/local/actianzen/etc

If you do not want the configuration file in the default location, enter a valid path.

TEMPDB Directory (Client Reporting Engine only)

Type	Range	Default	Units	Requires Engine Restart
String	not applicable	Varies	None	Yes

This property sets the location where Client Reporting Engine creates a temporary database to cache data from its storage server. It must be a valid path and include a drive or volume specification or UNC path. Default varies with the operating system. This setting is provided only for Client Reporting Engine.

Information

Information lists the following display-only items:

Display-only Item	Discussion
Server name	The name of the machine on which the database engine is running
Engine version	The release version of the database engine
Engine type	The product category of the database engine

Memory Usage

Memory Usage contains the following configuration settings:

- Allocate Resources at Startup
- Back to Minimal State if Inactive
- Minimal State Delay
- Sort Buffer Size
- System Cache

Allocate Resources at Startup

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Yes

This setting instructs the MicroKernel to allocate resources, including threads and memory buffers, when the MicroKernel is started. The resources allocated at startup includes the background threads in addition to the L1 cache. Zen components automatically allocate resources as needed. Therefore, in most cases, this setting can be off, which is the default.

With the setting off, the MicroKernel does not allocate any resources until the first operation request. If your server system supports a large number of users, you may prefer to have this setting turned on.

When first set to on, the setting may not produce any noticeable difference in the memory allocated because of how Windows behaves. When the MicroKernel allocates its L1 cache, the Windows operating system reserves pages of memory but does not commit them to the

MicroKernel. Later, when the MicroKernel actually accesses cache memory, Windows then commits physical pages and memory usage of Zen components increases.

If you look at the VM Size column in Windows Task Manager, you can see the memory value change as L1 cache is accessed. You should also be able to see the number of threads change when background threads are accessed.

Back to Minimal State if Inactive

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Yes

This setting causes the MicroKernel to free considerable memory and thread resources to the system and return to a minimal state after a certain amount of time without any active clients. The time interval is specified by the value of Minimal State Delay. The MicroKernel reallocates resources when another client becomes active.

Minimal State Delay

Type	Range	Default	Units	Requires Engine Restart
Numeric	0–2147483647	300	milliseconds	Yes

This value sets how long the MicroKernel waits during a period of inactivity before returning to a minimal state, the initial state in which the MicroKernel begins. By returning to this state, the MicroKernel frees memory and thread resources to the system. In some cases, you may not want the MicroKernel to return to a minimal state. For example, you may be running a batch file that uses the MicroKernel repeatedly. The MicroKernel reallocates resources when another client becomes active.

This setting is ignored if the value of Back to Minimal State if Inactive is set to **Off** (the default).

Sort Buffer Size

Type	Range	Default	Units	Requires Engine Restart
Numeric	0–limited by memory	0	bytes	Yes

This setting specifies the maximum amount of memory (in kilobytes) that the MicroKernel dynamically allocates and releases for sorting purposes during run-time creation of indexes.

If the memory required for sorting exceeds the size specified or is greater than 60 percent of the available process memory, the MicroKernel creates a temporary file. The amount of available memory for a process is a dynamic value and varies according to system configuration and load. If you specify zero kilobytes, the MicroKernel allocates as much memory as needed, up to 60 percent of the available memory.

System Cache

Type	Range	Default	Units	Requires Engine Restart
Boolean	On, Off	Off (Server) On (Workgroup)	None	Yes

This option specifies whether the MicroKernel should use the system cache in addition to the MicroKernel's own database cache, as set using the configuration parameter Cache Allocation Size.

If you are using the L2 cache, you should set **System Cache** to Off. Check the setting of Max MicroKernel Memory Usage. When Max MicroKernel Memory Usage is set to a value greater than zero, you are using L2 cache.

If you are not using L2 cache, performance can be enhanced by turning on **System Cache**. The MicroKernel relies on the system cache to organize and group pages to be written. It delays a flush long enough to allow the system cache to write the pages to disk in a more efficient manner. However, if your server has an advanced self-cached disk array, you might achieve better performance by setting **System Cache** to **Off**.

For Windows Server only, you can use the Paging File and Process objects in the Windows Performance Monitor tool to determine whether the Windows system cache is being used effectively. For the zenengnsvc.exe instance, monitor the % Usage and % Usage Peak in the Page File object and the Page Faults/Second and Page File Bytes in the Process object.

Performance Tuning

Performance tuning involves the following configuration settings:

- Automatic Defragmentation
- Cache Allocation Size
- Communications Threads
- File Close Delay
- File Growth Factor
- Index Balancing
- Limit Segment Size to 2 GB
- Transaction Log Buffer Size
- Max MicroKernel Memory Usage
- Number of Input/Output Threads
- Transaction Log Size

Automatic Defragmentation

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	No

This setting turns automatic defragmentation on or off. When it is on, the feature works as follows:

- One hour after the setting is turned on, the engine begins discovery of files to defragment.
- The engine first identifies files active since the last engine restart.
- The engine then tests those files for any of the following criteria.

Analysis	Greater Than or Equal to
Fragmented	15%
Unused	15%
Out of Order	5%

These criteria are the same as the ones defined under the Watch List topic. The advantage of automatic defragmentation is that you do not have to manually select files, add them to the Watch List, and then manually check and defragment them.

- Defragmentation begins immediately on the first file that meets any of the criteria.

- The engine defragments one file at a time. When it finishes with a file, it waits one minute before starting to defragment the next one.

- For efficiency, files matching the following criteria are ignored:

  - Already defragmented in the last 24 hours

  - Smaller than 10 MB

  If you want to defragment such files, add them to the Watch List for manual monitoring or use the **dbdefrag** command line tool.

- When all files meeting automatic defragmentation criteria are done, the engine waits 60 minutes before beginning the next round of file discovery.

- Messages in zen.log state whether a file was defragmented automatically or manually.

Note: If an active file is closed and its pages flushed from cache, the engine no longer considers it a candidate for automatic defragmenting. When the file is next opened, the engine discovers it again.

For more information about defragmenting data, see Monitoring Data File Fragmentation.

Cache Allocation Size

Type	Range	Default	Units	Requires Engine Restart
Numeric	1 MB to the amount limited by memory	Server value calculated on first startup, 64 MB for other editions	MB	Yes

This property sets the size of the Level 1 cache that the MicroKernel allocates. The MicroKernel uses this cache when accessing data files.

Generally speaking, overall performance is best when the cache allocation size is a value less than 40% of the physical memory on the system, and the configuration property Max MicroKernel Memory Usage is set to a value greater than 40%. Your optimal settings depend on the size of

your data files, the number of other applications running on the system, and the amount of physical memory.

Setting for a Server Engine

The initial setting is 20% of physical memory. The database engine calculates this value the first time it starts and writes it to the registry. From then on, whenever the engine starts, it reads the value in the registry. The engine never recalculates the value. Changing it manually updates the registry.

Setting for Workgroup, Client Cache, and Client Reporting Engines

For Zen installations other than servers, cache allocation size is set to a default value of 64 MB. As for servers, changing it manually updates the registry.

Note: If you use Zen clients before PSQL v10, the value for Fcache allocation size must be specified in bytes, with a minimum of 64 KB (65,536 bytes).

Using Cache Allocation Size to Optimize Performance

To optimize performance, set a larger cache allocation size. The value should be no larger than the sum of the sizes of the files that the engine is using. Setting a higher value provides no benefit and may waste memory needed by the system for other tasks. Taking all available memory, especially if the system is running other applications, may lead to undesirable behavior.

If you add or remove memory from the system, you also should reset this property to take account of the new amount of memory.

Communications Threads

Type	Range	Default	Units	Requires Engine Restart
Numeric	*num_cores* to 256	*num_cores*, where *num_cores* is the number of processors in the machine on which the database engine is running	None	Yes

This setting specifies how many threads the MicroKernel initially spawns to handle requests from remote clients. Communication threads are the elements that actually perform Btrieve operations on behalf of the requesting remote client process. In this way they are very similar to worker threads. Communications threads increase dynamically as needed up the maximum range allowed. The maximum is 256.

The Communications Threads setting can help improve scaling under certain conditions. For example, if you have many clients performing operations (typically writes) on one file, a lower setting should improve scalability. The lower number of threads prevents context switching on system resources. Another condition that this setting may improve is a slowdown caused by thrashing among large numbers of worker threads. Worker threads are dynamically created only if all existing threads are waiting on record or file locks.

File Close Delay

Type	Range	Default	Units	Requires Engine Restart
Numeric	0-5000	50	Milliseconds	No

This setting gives control over how long the engine holds files open after their client connections have closed. Because opening files repeatedly can slow engine performance, increasing the time that files remain open can offer improvement. The delay can be as high as 5 seconds. A value of 0 means the setting is turned off and the file is closed immediately after the last user issues a Btrieve Close operation.

You can use **butil -close** to have the engine check the specified files and immediately close any with remaining delay times. For more information, see the Close command.

File Growth Factor

Type	Range	Default	Units	Requires Engine Restart
Numeric	0-100	15	Percent	Yes

This value specifies the approximate percentage of free pages to maintain in version 8.x and later format data files. The setting does not apply to any previous file format versions. The MicroKernel uses this value to decide whether to extend a file or use free pages first. The database engine has the ability to write multiple contiguous file pages on disk. The goal is to improve the performance of disk writes as the number of free pages increases within a file. However, because disk write performance is a trade-off against file size, allowing too many free pages in a file can reduce overall performance.

To maintain a certain number of contiguous free pages in a data file, the database engine must periodically expand the file. Keep in mind that file growth factors are exponential. For example, if you start with a file that has no free pages and you set a File Growth Factor value of 50%, then the file will eventually double in size. If you set a File Growth Factor value of 75%, the file will quadruple in size. A value of 90% will allow the file size to grow by as much as 10 times.

Note that only completely unused pages are counted as empty, so 15% empty pages does not mean that 15% of the file is unused, since even mostly unused pages are not counted as empty.

Depending on how heavily a given file is being updated, the actual percentage of empty pages may be much less at any given moment.

This setting is not applicable to pre-8.x format files. These older files also have empty pages, but the percentage of empty pages varies with the activity on the file.

Index Balancing

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Yes

This setting controls whether the MicroKernel performs index balancing. Index balancing increases performance on read operations. However, when you enable this option, the MicroKernel requires extra time and may require more disk I/O during insert, update, and delete operations. For more information about index balancing, see *Zen Programmer's Guide*.

Limit Segment Size to 2 GB

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Yes

This setting specifies that a data file is to be automatically broken into 2GB operating system file segments as its size passes that boundary. If set to **On**, this setting specifies that you want data files divided into 2GB segments. If set to **Off**, this setting specifies that data files will remain a single, nonsegmented file. The advantage of using a larger nonsegmented file is more efficient disk I/O. Therefore, you can expect increased performance.

Any nonsegmented files are subject to the limit on file size specified by your operating system. If a previously created file is already segmented, that segmentation remains on the file.

See also Automatic Upgrade of File Version for related information.

Note: This setting does not affect files version 13.0 or later, which are never segmented.

Transaction Log Buffer Size

Type	Range	Default	Units	Requires Engine Restart
Numeric	262144 (0.25 MB) to limited by memory	The smaller of the following values: • Physical memory in MB / 256 • 33549312 (32 MB)	bytes	Yes

This setting specifies the size of both the transaction log buffer and the archival log buffer that the MicroKernel uses. You can enhance performance by increasing the transaction log buffer size, because the MicroKernel writes the log information to disk less frequently.

Note: If you set **Transaction Log Buffer Size** to a value greater than Transaction Log Size, then the MicroKernel automatically increments **Transaction Log Size** to the value you specified for **Transaction Log Buffer Size**.

Max MicroKernel Memory Usage

Type	Range	Default	Units	Requires Engine Restart
Numeric	0-100	60	Percent	No

This property dynamically controls the size of the secondary (L2) cache so that all data caching (L1 + L2), plus additional internal memory usage needed by the MicroKernel Engine, does not exceed the proportion of total physical memory specified. The engine uses less than the specified proportion if it is not needed or not available. It may use additional memory for internal processing beyond the limit specified by this configuration if necessary to handle certain scenarios, such as massive data insert, update, and delete operations in transactions. Note that memory for the Relational Engine is not included in this discussion.

Entering a value of zero turns off dynamic caching. In this case, the only cache available is L1, the size of which is specified by Cache Allocation Size. Also in this case, the maximum amount of memory used will not be limited, and additional memory for the MicroKernel will be allocated and released as needed.

If you have a dedicated database server machine, then you should set Max MicroKernel Memory Usage to the maximum value, or whatever proportion of memory is not occupied by the operating system. If you run other applications on your database server and you need to balance

performance among all of these, then you should set this value lower so that the database cache does not compete as much with the other applications when using available memory.

Note: If Cache Allocation Size is set to a higher amount of physical memory than **Max MicroKernel Memory Usage**, then Cache Allocation Size takes precedence. For example, a machine has 1 GB of physical memory and you set Cache Allocation Size to 600 MB and Max MicroKernel Memory Usage to 40%. The L1 cache is allocated 600 MB of memory. Since Cache Allocation Size is higher, it takes precedence and the amount of memory allocated to the L1 cache exceeds the value specified for Max MicroKernel Memory Usage.

Use the following equation to determine the approximate value for **Max MicroKernel Memory Usage.**

$(L1\_cache\_size + internal\_allocations + L2\_cache\_size / size\_of\_physical\_memory) * 100$

where:

$L1\_cache\_size$ is the Cache Allocation Size

$internal\_allocations$ is approximately 25% of the size of the L1 cache

$L2\_cache\_size$ is the amount of memory that expands and contracts based on memory load of the system

$size\_of\_physical\_memory$ is the amount of memory installed in the machine

For more information on tuning performance, see Tuning Performance.

Number of Input/Output Threads

Type	Range	Default	Units	Requires Engine Restart
Numeric	1 to 1024	32	None	Yes

This setting specifies how many background I/O threads the MicroKernel spawns. These threads are responsible for writing all pages from the MicroKernel cache to the file on disk in an atomic and consistent manner. They also are responsible for initially opening a file and reading the File Control Record. Most of the other reads are done by local worker threads and communication threads. When the MicroKernel updates or writes to data files, it assigns each file to a particular I/O thread sequentially. When it reaches the last thread, the MicroKernel starts over until all data files have been assigned to a background thread. Because the MicroKernel does not spawn additional I/O threads as needed, specify the maximum number of I/O threads you anticipate needing.

For best performance, set this value based on the average number of open files. Monitor shows the current and peak number of files open. If your database has an average of 256 files open, then the default of 32 I/O threads makes each thread responsible for 8 files. A good rule of thumb is to have about 8 files per I/O thread. For example, if your average number of open files is 400, you should use about 50 I/O threads. Specifying a value higher than 64 may degrade performance, but that depends on the capabilities of the system.

Note: No accurate way is available to calculate the appropriate number of I/O threads because this setting depends on the machine's characteristics, OS configuration, and the database engine's planned work load.

Transaction Log Size

Type	Range	Default	Units	Requires Engine Restart
Numeric	65536 to limited by disk space	2 times the Transaction Log Buffer Size	bytes	Yes

This setting specifies the maximum size of a transaction log segment. When the log file reaches its size limit, the MicroKernel closes the old log segment file and starts a new one. You might want to limit the size of your transaction log segments, because this reduces the disk space that the MicroKernel uses temporarily. However, limiting the size of the transaction log segments does require more processing by the MicroKernel and can decrease performance, because it has to close and create log segments more frequently.

Note: If you set the value for this option less than the value you specified for Transaction Log Buffer Size, the Database Engine automatically adjusts **Transaction Log Size** by setting it to the value of **Transaction Log Buffer Size**.

Windows Client Configuration Properties

All Zen editions can be configured as clients. The client properties must be set independently for each client, including servers acting as clients. You can configure a client on Windows platforms using Zen Control Center or the command line tool **bcfg**. For ZenCC, open the properties window by right-clicking MicroKernel Router node in Zen Explorer. For **bcfg**, see Configuration Using Bcfg.

The following table lists all client properties and their settings in alphabetical order. The settings are linked to topics with more information.

Configuration Option	Parameter Name
Access	Gateway Durability
	Number of Load Retries
	Use IDS
	Use Local MicroKernel Engine
	Use Remote MicroKernel Engine
	Wire Encryption
	Wire Encryption Level
Application Characteristics	Embedded Spaces
	Verify Key Length
	Splash Screen
Cache Engine	Allocate Resources at Startup
	Back to Minimal State if Inactive
	Cache Allocation Size
	Max MicroKernel Memory Usage
	Minimal State Delay
Cache Engine Debugging	The settings available under Cache Engine Debugging perform the same functions for the client cache as similar settings for a server. See the related server settings for Debugging.

Configuration Option	Parameter Name
Communication Protocols	Enable Auto Reconnect
	Supported Protocols
	Connection Timeout
Performance Tuning	Use Cache Engine
Security	Runtime Server Support

Access

Access contains the following configuration settings:

- Gateway Durability
- Number of Load Retries
- Use IDS
- Use Local MicroKernel Engine
- Use Remote MicroKernel Engine

Gateway Durability

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Not applicable

This option specifies whether the MicroKernel Router should store in the registry a list of computers that do not have Zen running on them. This decreases the time it takes to find a gateway engine. You must set this option to **Off** when new engines are added to the workgroup.

Number of Load Retries

Type	Range	Default	Units	Requires Engine Restart
Numeric	0 to 65536	5	None	Not applicable

This setting specifies the number of times the MicroKernel Router attempts to connect to the target engine.

Use IDS

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting is primarily for use with legacy applications that used Zen (IDS). IDS functionality is now integrated into the core product and IDS is no longer available as separately installed components. The integration of IDS requires that you reconfigure your client-server environment.

Typically, an application provides its own file location information. As an alternative, IDS provided file location mapping based on information in a text file, idshosts.

If your applications do not use the mapping feature through idshosts, set **Use IDS** to **Off** to improve performance.

If your applications already use idshosts, or if you prefer to use this alternative method to map file locations, set **Use IDS** to **On**. See Using the idshosts File.

Note: PSQL 8.5 or later is required if you set **Use IDS** to **On** or if your legacy applications pass file location information in the format of a PIDS URL. The requester uses database URIs to represent the IDS information. Database URIs were added with PSQL 8.5. See Database URIs in *Zen Programmer's Guide*.

Use Local MicroKernel Engine

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Not applicable

This setting determines whether a local application tries to connect to a local engine. If turned off, no attempt is made to connect to a local engine.

Use Remote MicroKernel Engine

Type	Range	Default	Units	Requires Engine Restart
Boolean	On, Off	On	None	Not applicable

This setting specifies whether the MicroKernel Router allows access to an engine running on a remote server. If this value is set to **On**, and **Use Local MicroKernel Engine** is set to **On**, the remote server is tried first.

Wire Encryption

See Wire Encryption.

Note: Client-side wire encryption settings are not used by the Zen JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in *JDBC Driver Guide* and Defining Basic Connection Strings in *Data Providers for ADO.NET Guide*.

Wire Encryption Level

See Wire Encryption Level.

Note: Client-side wire encryption settings are not used by the Zen JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in *JDBC Driver Guide* and Defining Basic Connection Strings in *Data Providers for ADO.NET Guide*.

Application Characteristics

Application Characteristics includes the following configuration settings:

- Embedded Spaces
- Splash Screen
- Verify Key Length

Embedded Spaces

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Not applicable

This option instructs the MicroKernel Engine to allow embedded spaces in file names for MicroKernel Engine operations.

Splash Screen

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting controls whether or not the MicroKernel Engine splash screen displays. The splash screen displays the first time a client requester loads.

Verify Key Length

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	On	None	Not applicable

This option can be used for legacy Btrieve applications to prevent the requester from verifying that the index key length passed to the client requester is large enough to hold the key. Setting this option to off may allow such applications to avoid status 21 errors.

Caution! If set to off, this option disables the check by the Zen requester to prevent memory overwrites. A memory overwrite can cause a general protection fault (GPF) among other undesirable conditions.

Cache Engine

These settings apply only when the cache engine is running. The Workgroup engine doubles as a cache engine. Note, however, that the cache engine is **not** used if a database server engine is running.

Cache Engine contains the following configuration settings:

- Allocate Resources at Startup
- Back to Minimal State if Inactive
- Cache Allocation Size
- Max MicroKernel Memory Usage
- Minimal State Delay

Allocate Resources at Startup

Type	Range	Default	Units	Requires Database Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting instructs the cache engine to allocate resources, including threads and memory buffers, when the cache engine is started.

If you turn this option off, the cache engine does not allocate any resources until the first operation request. Zen components automatically allocate resources as needed. Therefore, in most cases you do not need to do so explicitly.

Back to Minimal State if Inactive

This setting displays only if the Workgroup engine is running.

Type	Range	Default	Units	Requires Database Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting causes the cache engine to free considerable memory and thread resources to the system and return to a minimal state after a certain amount of time without any active clients. The time interval is specified by the value of Minimal State Delay. The cache engine reallocates resources when another client becomes active.

Cache Allocation Size

Type	Range	Default	Units	Requires Database Engine Restart
Numeric	1 MB to the amount limited by memory	Initialized dynamically at first startup	MB	Not applicable

This setting specifies the size of the Level 1 cache that the MicroKernel allocates; the MicroKernel uses this cache when accessing any data files.

Generally speaking, overall performance is best when the Cache Allocation Size is a value less than 40% of the physical memory on the system, and the configuration property Max MicroKernel Memory Usage is set to a value greater than 40 percent. Your optimal settings depend on the size of your data files, the number of other applications running on the system, and the amount of physical memory.

The database engine sets this value the first time it starts and writes it to the registry. The initial value is 20% of physical memory. From then on whenever the engine starts, it reads the value in the registry. Changing the value of this property updates it in the registry. If you add or remove memory from the system, you should reset this property to take best advantage of the new amount of memory.

Note: If you use Zen Clients earlier than PSQL v10, the value for cache allocation size must be specified in bytes, with a minimum of 64 KB (65,536 bytes).

Max MicroKernel Memory Usage

Type	Range	Default	Units	Requires Database Engine Restart
Numeric	0-100	60	Percent	Not applicable

This setting specifies the maximum percentage of total physical memory that the cache engine is allowed to consume. The percentage applies to L1, L2, and all miscellaneous memory used by the cache engine. The database engine uses less if the memory is not needed or not available.

If a value of zero (0) is entered, then dynamic caching is turned off. In this case, the only cache available is L1, the available amount for which is set by Cache Allocation Size.

For more information on tuning performance, see Tuning Performance.

Minimal State Delay

This setting displays only if the Workgroup engine is running.

Type	Range	Default	Units	Requires Database Engine Restart
Numeric	0–2147483647	300	milliseconds	Not applicable

This setting specifies how long the cache engine waits during a period of inactivity before returning to a minimal state. (This is the initial state in which the cache engine begins.) By returning to a minimal state, the cache engine frees considerable memory and thread resources to the system. In some cases, you may not want the cache engine to return to a minimal state. For example, you may be running a batch file that uses the cache engine repeatedly. The cache engine reallocates resources when another client becomes active.

This setting is ignored if the value of Back to Minimal State if Inactive is set to **Off** (the default).

Cache Engine Debugging

These settings apply only when the cache engine is running. The Workgroup engine doubles as a cache engine. Note, however, that the cache engine is **not** used if a database engine is running.

The settings available under **Cache Engine Debugging** perform the same functions for the Client cache as similar settings under **Server Debugging** perform for the main database engine. For more information about each setting, see the related server setting:

- Number of Bytes from Data Buffer
- Number of Bytes from Key Buffer
- Select Operations
- Trace File Location
- Trace Operation

Communication Protocols

Communication Protocols contains the following configuration settings:

- Enable Auto Reconnect
- Supported Protocols
- Connection Timeout

Enable Auto Reconnect

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting specifies whether you want the client to attempt to automatically reconnect during a network outage. A setting of **On** means Auto Reconnect is enabled.

Auto Reconnect is not in effect unless this setting is also enabled in the server configuration.

Supported Protocols

Type	Range	Default	Units	Requires Engine Restart
Multiple select	One or more	TCP/IP	None	Yes

This setting specifies the protocols by which the database engine listens for client connections. If more than one protocol is selected, the engine begins the session by listening on all of them. The first protocol that succeeds is used for the remainder of the session. You must have at least one protocol enabled on both the client and the server or they cannot communicate.

Note: The default is TCP/IP, which is the only option currently supported.

Linux and Raspbian

TCP/IP is the only supported protocol for Zen on Linux and Raspbian. Therefore, the Supported Protocols setting is not available for those environments.

Connection Timeout

Type	Range	Default	Units	Requires Engine Restart
Numeric	1 to 2147483647	15	seconds	Not applicable

The value of this setting specifies how long the client waits while searching for or connecting to a remote database engine. If this value is set too low, the client may return spurious "server not found" errors, because it is timing out before it has a chance to complete the connection. If the value is set too high, when the client attempts to connect to a server that is not reachable, you may encounter lengthy delays before receiving an error. Generally speaking, a value between 15 and

30 seconds is adequate for most networks. If you receive many errors of server not found, try a higher setting.

This setting was previously named: **TCP/IP Timeout for Communication Requester**.

Performance Tuning

Performance Tuning contains the following configuration property:

- Use Cache Engine

Use Cache Engine

Type	Range	Default	Units	Requires Database Engine Restart
Boolean	On Off	Off	None	Not applicable

This property sets whether to use the Client cache engine. This engine is a subset of the MicroKernel engine that caches data only for reading and runs as a separate process. The Client cache engine is also referred to as the *Client cache* or *cache engine*.

If **Use Cache Engine** is off, nothing is cached on the Client side. Read requests from an application retrieve data directly from the remote database engine.

If **Use Cache Engine** is on, the Client cache engine acts as an intermediary between the Client and the remote database Engine to cache data. The first time an application issues a read request, the Client cache engine caches the data. It then answers additional read requests for the same record using the cached data. The read operation does not have to access the remote database engine.

The Client cache is similar to the Workgroup Engine. By default, it automatically loads in memory when an application first accesses the database, and it unloads shortly after that application exits.

You may wish to keep the Client cache in memory to avoid the performance cost of repopulating it with each usage session. To keep the Client cache loaded, run the following from a command prompt: *install_path*`\Zen\bin\zenengnapp`

After the Client cache starts, an icon appears to the far right of the taskbar in the notification area. The icon allows you to control the Client cache engine. To stop the Client cache, right-click the icon and select **Stop Engines and Exit**.

If the Client is installed as a service, the Client cache engine is set by default to start automatically. However, even though the Client cache service is running, an application does not use the Client cache unless **Use Cache Engine** is turned on.

Note: Zen synchronizes the Client cache with the database engine cache and other Client cache locations. This behavior is fully automatic and entirely transparent. However, there is a maximum delay of 5 seconds between any database change happening in the database engine cache and it being reflected in all Client cache locations. If the possibility of such stale pages existing in the cache for a maximum of 5 seconds is unacceptable in your system, do not use the Client cache.

The following operations are **not** stored in the Client cache:

- Everything inside a transaction

- Operations with a lock bias

- Write operations such as INSERT, UPDATE, and DELETE

- Open and close operations

See also the discussion of Client cache for the Cache Allocation Size property.

Security

Security contains the following configuration setting:

- Runtime Server Support

Runtime Server Support

Type	Range	Default	Units	Requires Engine Restart
String	Yes No user_name,password	Yes	None	Not applicable

This setting controls runtime server support. If enabled with the value **Yes**, the current user name and password for the drive on which you are presently running are used. To enable RTSS with a different user name, enter *user_name,password.*

SUPERVISOR and ADMIN are not valid user names, even if supplied with the correct password. If the requester cannot find a user name other than SUPERVISOR or ADMIN, it does not attempt to log in.

Linux and Raspbian Client Configuration Properties

You can configure Zen clients using Zen Control Center or the command line tool **bcfg**. For ZenCC, open the properties window by right-clicking a MicroKernel Router node in Zen Control Center. For **bcfg**, see Configuration Using Bcfg.

Note: For Raspbian, where the ZenCC GUI is not supported, you can open ZenCC in another Zen installation and add the client remotely, then access its configuration properties. You can also use **bcfg** in a remote session, such as PowerShell.

Case of Configuration Values

When checking or editing the values of settings, the Linux and Raspbian clients perform a case-insensitive comparison. For example, entering `Yes` or `yes` for a setting value is interpreted identically by the client.

Client Performance Affected by Local Setting

When the Linux and Raspbian client interface is first invoked, it populates its default settings in the Zen registry. The Zen Client does not have knowledge on whether its installation includes a server engine or not. Therefore, it sets the Local setting to yes. This can have an impact on the performance of your client.

If the machine on which you are using the client does not have a server engine, you should set the local setting to no. See Use Local MicroKernel Engine.

File Names with Embedded Spaces

By default, the Linux and Raspbian client interface supports file names that contain embedded spaces.

For example:

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

If you want to use file names without embedded spaces, you need to change the Embedded Spaces setting. See Embedded Spaces.

Configuration Reference

The following table lists the configuration options for Linux and Raspbian clients in alphabetical order. The settings are linked to topics with more information.

Configuration Option	Parameter Name
Access	Use Local MicroKernel Engine
	Use Remote MicroKernel Engine
	Use IDS
	Wire Encryption
	Wire Encryption Level
Communication Protocols	Enable Auto Reconnect
Application Characteristics	Embedded Spaces
	Verify Key Length

Access

Access contains the following configuration settings:

- Use Local MicroKernel Engine
- Use Remote MicroKernel Engine
- Use IDS
- Wire Encryption
- Wire Encryption Level

Use Local MicroKernel Engine

See Use Local MicroKernel Engine.

Use Remote MicroKernel Engine

See Use Remote MicroKernel Engine.

Remote Engine and UNC Paths

For UNC paths to work properly from a client, the following steps must be performed:

- You must be running an engine on the same computer as the file that you are trying to access

- You must set **Use Remote MicroKernel Engine** to on.

Note: You cannot use a UNC path that points to the local Linux or Raspbian system. However, you can use a path that is in the UNC style, such as `//localhost/usr/local/actianzen/data/samples/sample.btr`

If you do not want an engine on your file server (that is, you want to use the client's local engine), then you can mount the remote file system on the client and modify the path so that it is a "native format" path rather than UNC format. For example, the following path is a native format on Linux and Raspbian:

`/mnt/myremotedata/sample.btr`

Use IDS

See Use IDS.

Wire Encryption

See Wire Encryption.

Note: Client-side wire encryption settings are not used by the Zen JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in *JDBC Driver Guide* and Defining Basic Connection Strings in *Data Provider for .NET Guide*.

Wire Encryption Level

See Wire Encryption Level.

Note: Client-side wire encryption settings are not used by the Zen JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in *JDBC Driver Guide* and Defining Basic Connection Strings in *Data Provider for .NET Guide*.

Communication Protocols

Communication protocols contains the following configuration setting:

- Enable Auto Reconnect

Enable Auto Reconnect

Type	Range	Default	Units	Requires Engine Restart
Boolean	On Off	Off	None	Not applicable

This setting specifies whether you want the client to attempt to automatically reconnect during a network outage. A setting of **on** means Auto Reconnect is enabled.

Auto Reconnect is not in effect unless this setting is also enabled in the server configuration.

Note: The Zen Linux and Raspbian client supports this feature, but currently the server on those operating systems does not. You can use Auto Reconnect only from those clients connecting to a Windows server.

Application Characteristics

Application characteristics include the following configuration settings:

- Embedded Spaces
- Verify Key Length

Embedded Spaces

See Embedded Spaces.

Verify Key Length

See Verify Key Length.

Reporting Engine Configuration Properties

You can configure a Client Reporting Engine using Zen Control Center or the command line tool **bcfg**.

- For ZenCC, open the properties window by right-clicking the Client Reporting Engine node in Zen Explorer.

- For **bcfg**, see Configuration Using Bcfg.

Performance

The following topics cover database performance:

- SQL Query Logging

- Analyzing Performance

- Tuning Performance

- Performance on Hypervisor Products

You will find more information about improving database performance under Performance Optimization Reference in *SQL Engine Reference*.

SQL Query Logging

Database performance depends on a range of factors, including the efficiency of SQL statements. Logging of SQL queries may help to identify poorly performing queries so that they can be optimized. Client applications and even other query tools may make unknown queries that logging can discover.

To log queries, Zen v16 introduces two settings in the engine properties:

- Query Logging: Enables SQL engine to log query execution to a file for review.

- Query Log Fetches: Specifies whether to include FETCH operations in the query log. Turning off fetch logging makes the log less verbose and easier to review.

To display these settings in ZenCC, right-click a database and select Properties, then in the properties for that database, select Debugging. Use the check box for each setting to enable or disable it. To enable Query Log Fetches, Query Logging must also be enabled. You can also turn the Query Logging and Query Log Fetches settings on and off using **bcfg** with their numeric ID, which is 184 and 185, respectively.

After you turn on Query Logging for a database, each new database connection logs entries to a new text file in C:\ProgramData\Actian\Zen\logs. The name of the file is of the form <user name>-<database name>-yyyy-mm-dd_hh-mm-ss.mmm-query.log. When the connection session ends, the file is closed. The next new database connection creates a new log file. In the current release, the log file location cannot be changed.

Log entries reference ODBC calls. For security reasons, certain SQL statements are not logged. The following table lists both logged and unlogged statements.

Query Logging					
The following SQL statements are logged:					
CALL	COMMIT	DELETE	EXEC	EXECUTE	INSERT
RELEASE	ROLLBACK	SAVEPOINT	SELECT	UPDATE	
The following SQL statements are not logged:					
ALTER	CREATE	DROP	GRANT	ALTER	CREATE
INSERT	REVOKE	SET	USE		

Notes

- Query logging captures INSERT or UPSERT statements only in combination with SELECT queries with WHERE clauses. The adding or updating of records on its own does not involve a query to optimize. Adding a SELECT operation and WHERE clause introduces opportunities to optimize execution.

- The query log captures queries executed by stored procedure calls.

- Passwords are not visible in query logs.

Example

The following example demonstrates the use of query logging and the Query Plan Viewer to detect and confirm SQL execution performance improvement.

Running the following SQL statement in the Demodata database:

```
SELECT Major, COUNT(*) AS NumMajors from Student GROUP BY Major;
```

Results in this query log entry:

```
===============================================================================================
2024-05-01T13:42:58.353: Start:   SQLPrepare                                Handle: 0x29b7fa21410
SQL Query text:          "select Major, count(*) as NumMajors from Student group by Major"
2024-05-01T13:42:58.384: End:    SQLPrepare            Retcode:0          Handle: 0x29b7fa21410
===============================================================================================
```

The log shows an execution time of 31 milliseconds.

Creating an index before the SELECT statement shortens the execution time:

```
CREATE INDEX Major_index ON Student (Major);
SELECT Major, COUNT(*) AS NumMajors FROM Student GROUP BY Major;
```

Resulting in the query log entry:

```
===============================================================================================
2024-05-01T13:42:58.634: Start:   SQLPrepare                                Handle: 0x29b7f9a4200
SQL Query text:          "select Major, count(*) as NumMajors from Student group by Major"
2024-05-01T13:42:58.634: End:    SQLPrepare            Retcode:0          Handle: 0x29b7f9a4200
===============================================================================================
```

Which reduces execution time to less than 1 millisecond.

Zen Query Plan Viewer can help to visualize the improvement. The following script creates a query plan file.

```
DROP INDEX IF EXISTS student.major_index;
SET qryplanoutput = 'C:\ProgramData\Actian\Zen\logs\Majors.qpf';
SELECT Major, COUNT(*) AS NumMajors from Student GROUP BY Major;
CREATE INDEX Major_index ON Student (Major);
SELECT Major, COUNT(*) AS NumMajors FROM Student GROUP BY Major;
```

```
SET qryplanoutput = NULL;
```
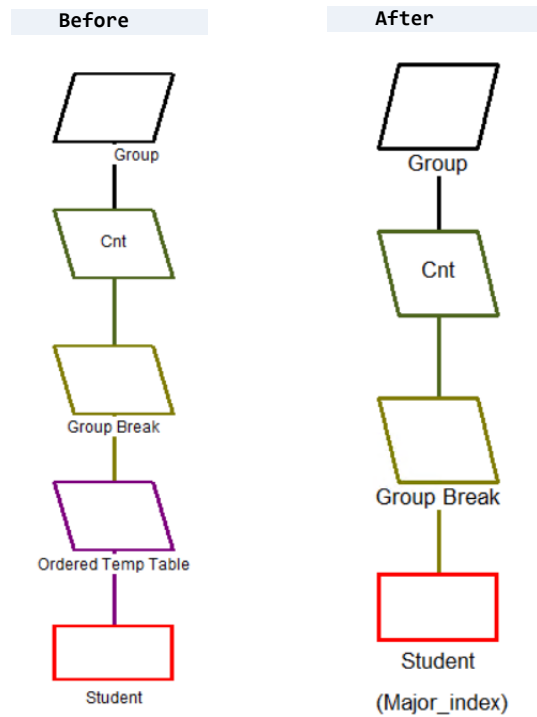

Before adding the index, the query plan shows the creation of a temporary file to do the grouping. After the index is added, the plan shows that the temporary file is no longer needed, improving performance. For more information, see Query Plan Viewer.

Analyzing Performance

Zen servers on Windows provide performance counters for use with the Windows Performance Monitor. The performance counters measure state or activity of the database engine, which allows you to analyze performance of your application.

Detailed documentation for this feature is grouped with a number of other Zen database monitoring capabilities under the topic Monitoring Performance Counters.

Tuning Performance

This section provides some general tips on how to maximize performance on the initial database connection and on runtime operations. While we can offer some general guidelines, the performance of any specific application depends on a great number of factors, including but not limited to the following:

- Network bandwidth and use

- Coding techniques in the application

- Physical storage space available

- Memory available

- Processor speed

- Application usage patterns (such as write-heavy, read-heavy, transactions used/not used, small record size, large record size, complex queries, simple queries, ODBC, Btrieve only, and so forth)

- Unrelated applications competing for CPU cycles

- Database engine configuration

As you can see, the engine configuration plays a relatively limited role in the overall performance of any given application. Further, the database engine dynamically manages a variety of resources based on usage patterns. It tunes itself to your environment as needed. The following sections provided are offered only as helpful guidelines and are not a guarantee of any specific level of performance.

Spotting Performance Bottlenecks

You can use Monitor tool to discover performance bottlenecks related to certain database engine configuration options. You can start Monitor from the operating system **Start** menu or **Apps** screen or from the **Tools** menu in Zen Control Center.

Monitor Displays and Configuration Properties

Two Monitor tabs display performance readings related to configuration options:

- Resource Usage

- MicroKernel Communications Statistics

The database engine dynamically manages several server configuration options, as shown in the following table.

Dynamically Managed Settings	Value Displayed in Resource Usage Tab	Value Displayed in Communications Tab
Files	X	
Handles	X	
Clients	X	
Worker Threads	X	
Total Remote Sessions		X

Interpreting the Displays and Taking Action

You can make use of the information displayed in Monitor. Monitor displays three pieces of information about each type of resource. For example, the Total Remote Sessions display shows:

- *Current*. The current number of actual remote sessions operating.

- *Peak*. The highest number of actual remote sessions that have been used since the engine was started.

- *Maximum*. The maximum number of remote sessions allowed.

If the Peak value for a resource is the same as the Maximum value, then you may want to set the configuration property to increase the Maximum value for the resource, thus allowing the database engine to allocate additional instances of that particular resource when it needs to.

Before You Modify Configuration Properties

The following sections assume the following:

1. Zen Control Center (ZenCC) is already open.

   If you need assistance with this task, see Starting ZenCC on Windows of *Zen User's Guide*.

2. You have already registered (if applicable) the engine you wish to configure.

   If you need assistance with this task, see To register a remote server engine of *Zen User's Guide*.

3. You have appropriate operating system privileges to configure the given engine.

If you need assistance with this task, see Granting Administrative Rights for the Database Engine of *Zen User's Guide*.

4. For some engine settings, you must restart the database engines after making configuration changes.

Minimizing Initial Connection Time

The theory underlying minimal connection time revolves around three requirements. These requirements are summarized next, and detailed procedures follow:

- *Known communications protocol.* You do not want the client or server to spend additional time trying to connect via protocols that are not supported in your environment. By removing client-server support for unavailable protocols, you prevent the networking components from trying to use them.

- *Known location of database engine.* You do not want the client to attempt to connect to an engine that does not exist. By specifying Workgroup-only or server-only support, as appropriate, you prevent the client from timing out while attempting non-existent connections. In environments where you have both unshared and shared databases, you can use Gateway Durability to force the database engine to keep track of machines where no database engine is running, so that it never attempts to connect to an engine on these machines, but uses another method to access the data.

- *Database engine ready to execute.* When a sleeping engine receives a new connection request, it takes time to reallocate resources and return to a runtime state. If connection speed is more important than resource usage on the server, you can prevent the server engine from sleeping when it is not in use.

These requirements can be met by managing client and server properties:

- Client Properties
- Server Properties

Client Properties

You must be at the client machine to change the client properties. You must change the properties at each individual client.

To minimize client-side connection delays

1. In Zen Explorer, expand the **Local Client** node in the tree (click the expand icon to the left of the node).

2. Right-click **MicroKernel Router**.

3. Click **Properties**.

4. Click **Communication Protocols** in the tree.

5. For **Supported Protocols**, ensure that the desired protocols are selected (check marked) and the protocols **not** being used are **not** check-marked.

6. Click **Apply**.

    The client is now prevented from attempting to communicate on protocols that are not being used.

7. Click **Access in the Properties tree.**

8. If you are using only a remote server or remote Workgroup engine, ensure that **Use Local MicroKernel Engine** check box is **not** selected so that it is set it to **Off**.

    If you are using only a local Workgroup engine, ensure that **Use Remote MicroKernel Engine** check box is **not** selected so that it is set it to **Off**.

    If you sometimes use a Workgroup engine and you sometimes connect to a server engine or a remote Workgroup engine, you must leave both settings selected so that they are **On**.

    In such a mixed environment with shared and unshared data, you can set **Gateway Durability** to **On** at each client. This setting forces the client software to keep a list of the names of any machines on which it is unable to connect to a database engine. In order for the client software to determine no engine exists on a given computer, it waits for all of its network protocol requests to time out.

    If your data is stored on a server that does not have a Zen database engine on it, and you have set **Use Remote MicroKernel Engine** to Yes, the client must time out at least once to discover that there is no engine on that machine. Gateway Durability ensures that this time-out only happens the first time your application tries to access that data.

Note: Using Gateway Durability fixes the network topology. If you later install a Zen server or Workgroup engine on the remote computer, you must turn off Gateway Durability on each client so that the list of computers without database engines is deleted (thus allowing you to connect to the new database engine). You may turn Gateway Durability back on immediately, but it starts with an empty list.

9. Click **OK**.

    The client is now prevented from attempting to connect to any database engine types that are not in use.

Server Properties

To minimize server-side connection delays

1. In Explorer, expand the **Engines** node in the tree (click the expand icon to the left of the node).

2. Right-click the database engine for which you want to specify configuration settings.

3. Click **Properties**.

4. For **Supported Protocols**, ensure that the desired protocols are selected (check marked) and the protocols **not** being used are **not** check marked.

5. Click **Apply**.

   The server is now prevented from attempting to communicate on protocols that are not being used.

Note: Ensure that at least one protocol you have selected for the server configuration is the same one as selected for the client configuration. Your client and server cannot communicate if they are not using the same protocol.

6. Click **Memory Usage** in the Properties tree.

7. Select **Allocate Resources at Startup** to set the value to **On (check box selected)**.

   This option specifies that the database engine should allocate all necessary memory when it starts up, rather than when the first connection request comes in. Choosing this value requires more memory, but from the client perspective allows the engine to become operational faster.

8. Ensure that **Back to Minimal State if Inactive** is set to **Off** (check box cleared).

   This option specifies that the database engine should not release resources back to the operating system if the engine is inactive. All resources remain allocated and ready for use at the next client connection.

9. Click **OK**.

10. Click **Yes** to restart the engine for these changes to take effect.

Maximizing Runtime Throughput

The theory behind maximum throughput relies on too many variables to list here. Here are several of the most significant factors:

- *Adequate physical memory.* If your host system has too little RAM, the system spends most of its time and CPU cycles swapping memory to disk and back, as different users and processes compete for limited resources.

- *Adequate CPU capacity.* If your CPU cannot keep up with the inflow of data processing requests, application performance suffers.

- *Adequate network bandwidth.* If your network is slow or suffers from a high collision rate, the database engine may sit idle between data access requests, while each client seems to display poor performance.

- *Minimal disk I/O.* Disk I/O is significantly slower than memory I/O. You want to avoid accessing the disk as much as possible, by having sufficient cache.

- *Adequate resource allocations.* Even with massive physical memory available, database performance may suffer if database engine configuration properties are set artificially low.

In the end, optimal performance is a balancing act among network bottlenecks, disk I/O bottlenecks, memory bottlenecks, and CPU bottlenecks. The following topics provide some guidelines on how to reduce memory and disk I/O bottlenecks:

- Fast Disk versus Fast CPU

- Ensuring Adequate Physical Memory and Database Cache

- Minimizing Disk I/O

- Ensuring Adequate Resource Allocation

Fast Disk versus Fast CPU

If you want to maximize the effect of your hardware investment for performance gains, you must understand the existing constraints on your performance. If you have a database that is so large that you cannot reasonably buy and install enough memory to cache a significant part of the database, then performance is likely to be constrained by the disk I/O. Under these conditions, you may be better off to invest in a fast RAID disk array to maximize performance of the disk I/O.

In addition, if your application uses the Relational Engine and forces temporary files to be created frequently, you may want to ensure that the directory where these files are created is located on a fast disk drive. For more information about the location of this directory and the types of queries that generate temporary files, see Temporary Files in *SQL Engine Reference*.

If your database is small enough to be fully or near-fully cached in memory, then adding a fast disk array is unlikely to provide a significant performance boost. Under these conditions, upgrading the CPU or adding an additional CPU may provide the best performance improvement value.

Ensuring Adequate Physical Memory and Database Cache

Starting with Pervasive.SQL V8, the database engine offers Level 2 dynamic cache in addition to the Level 1 cache specified by the configuration setting, Cache Allocation Size. Assuming you do not turn off the Level 2 dynamic cache by setting Max MicroKernel Memory Usage to zero, the need to manually adjust the Level 1 cache size is much less critical than in previous releases. With that in mind, this section explains how to ensure that you have enough memory available for optimal performance of the database engine.

Ideally, your database engine should be able to allocate enough memory to cache full copies of every database it hosts, thus avoiding as much disk I/O as possible. Obviously, caching one or more entire databases is not practical in some situations, particularly when database size is very large. In addition, such measures as adding RAM to the machine only improve performance if the existing system resources are heavily loaded under normal usage.

The database engine dynamically selects a Level 1 cache size value when it starts the first time. However, this value is based on available memory and may not be the ideal amount of cache for your environment.

To calculate the ideal size of the database memory cache

1. Start by adding up the file sizes of all the data files serviced by the database engine.

Note: If you have more than one database serviced by the engine, but they are never used at the same time, add up the file sizes of just the largest database.

For example, assume that your server has two databases with the following file sizes and that users access both databases at the same time:

Database A		Database 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		

The sum of all these files is 1430 MB.

The number you have now is the maximum amount of memory that the database engine would use if it cached all its hosted data. This number can be referred to as *MaxCache*.

You would never want to specify a value greater than this for Cache Allocation Size, because you would be allocating memory to the database engine that it would likely never use. A

reasonable rule of thumb is to set Cache Allocation Size to about 20% to 70% of *MaxCache*. Lower values in this range are best for read-intensive applications, and higher values are best for write-intensive applications since all write/update operations take place in the Level 1 cache.

Note: File pages are only written to the database cache when they are accessed. Thus, for a database engine to use *MaxCache* amount of memory requires every page in the database to be accessed. This system of estimating assumes a long-term steady state for database usage. If you bring the database engine down nightly or weekly, it may be unlikely that the database engine would access every page in the database within the given period of uptime.

If this situation applies to you, you may wish to estimate the average number of distinct pages that your application accesses within the given uptime period, multiply that by the page size, and obtain a more realistic value of *MaxCache* for your particular uptime scenario.

See also Zen MicroKernel Cache.

On Windows 32-bit operating systems, all user processes are limited to 2 GB of memory. If you have calculated a value of *MaxCache* larger than 2 GB, and your database engine runs on a 32-bit Windows operating system, then you should make sure that the *MaxCache* value you select never allows the engine to exceed the 2GB boundary, or your engine may fail to allocate memory and subsequently fail.

To determine how much total physical memory you need

Use the following equation to determine the approximate amount of total physical memory required by the database engine.

Maximum Database Memory Usage = *L1_cache_size* + *internal_allocations* + *L2_cache_size*

where:

L1_cache_size is the Cache Allocation Size

internal_allocations is approximately 25% of the size of the L1 cache

L2_cache_size is the amount of memory that expands and contracts based on memory load of the system

Note the following:

• The L1 cache is a fixed size based on Cache Allocation Size. It does not expand or contract based on database operations. If your application performs numerous WRITE operations, increase the L1 cache as much as possible.

- The greatest performance is obtained if all of the data can fit into the L1 cache. If all of the data will not fit into the L1 cache, adjust Cache Allocation Size and Max MicroKernel Memory Usage to use a reasonable amount of system memory.

- The L2 cache expands and contracts based on memory load of the system. For example, if other applications require more memory, the L2 cache contracts. If ample memory is available, the L2 cache reaches a maximum based on the equation above. The expansion and contraction affects performance. Contraction causes data pages to be removed from the L2 cache, for example. Reading the pages back from disk storage takes longer than if the pages could have been retained in the cache.

- The L2 cache contains compressed data pages (more pages in less space). Accessing pages from the L2 cache takes longer than from the L1 cache, but is faster than accessing the pages from disk storage. The L2 cache is helpful for applications that perform numerous READ operations but cannot fit all of the read data pages into the L1 cache.

Minimizing Disk I/O

Reading and writing data to/from disk is much slower than reading and writing to/from memory. Thus, one way to optimize performance is to minimize disk activity.

An important consideration in your attempts to minimize disk I/O is recoverability of data. Disk I/O is a direct trade off against transaction durability and recoverability. The more data you keep in memory without pausing to log changes to disk, the faster the database performs. On the other hand, the more data you keep in memory without pausing to log changes to disk, the more data you lose if the system experiences a failure.

To reduce disk I/O

1. As discussed in the topic Ensuring Adequate Physical Memory and Database Cache, one of the most important considerations is to ensure you have enough database memory cache to avoid frequently swapping data pages between disk and cache. See that section for details.

   One of the best ways to reduce disk I/O is to make sure that the dynamic Level 2 cache is turned on. The Level 2 cache adjusts its size dynamically as application usage changes, storing as much data as possible in memory and thus avoiding disk I/O when cache demands exceed the capacity of the Level 1 fixed cache. By default, the Level 2 cache is turned on. To verify that your database engine is using Level 2 cache, check the properties configuration setting Max MicroKernel Memory Usage.

2. Next, consider how much logging you require and what quantity of database operations you are willing to lose in a system failure. The greater the quantity of changes you are willing to lose, the more you can risk in the pursuit of performance.

Using Archival Logging, **Transaction Durability**, and **Transaction Logging** all require log files. By default, archival logging is turned off. Turn it on only if you perform regular backups and you need the capability to restore data up to the moment of a system failure. When you specify the files to be logged, be sure to specify only the files for which you absolutely must have logging. See Chapter , Logging, Backup, and Restore, for more information.

By default, transaction logging is turned on. Turning off transaction logging should improve performance slightly, but does not guarantee multifile consistency and transaction atomicity. Before turning off transaction logging, check with your application vendor to be sure they allow the application to run without this feature.

Caution! The consistency of any multifile database cannot be guaranteed if transaction logging is disabled.

By default, transaction durability is turned off. Turn on this feature only if your application requires completed transaction operations to be durable through a system crash. Transaction durability entails the highest performance penalty, and the trade off is the highest safety of your completed transactions.

3. If you have any logging features turned on, you can specify how much data the engine stores in memory before writing to disk. This feature is important because the changed data builds up over time. The more log data you allow to build up in memory, the less frequent the disk writes are.

   The setting **Transaction Log Buffer Size** specifies the number of bytes of database operations that the engine stores in memory before writing them out to the log files. (Click **Performance Tuning** on the server Properties tree.)

   If a system failure occurs, the data in the log buffer is lost.

4. If you have transaction durability turned on, you can specify the maximum size of the log segments on disk. Specifying a larger log segment size can improve performance slightly, because fewer log segments have to be created and closed over time.

   The setting **Transaction Log Size** specifies the maximum number of bytes that can be stored in a log segment before closing it and opening a new segment. (Click **Performance Tuning** on the server Properties tree.)

5. If your database is highly used, consider configuring your system to maintain the logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the writes to the log files and to the data file are split across different drives instead of competing for I/O bandwidth on a single drive. The overall disk I/O is not reduced, but the load is better distributed among the disk controllers.

6. If your application usage is weighted heavily in favor of database read operations, you can increase performance by turning on **Index Balancing**. (Click **Performance Tuning** on the server Properties tree.) Over time, index balancing increases the number of nodes on the average index page, allowing read operations to occur faster. However, for insert, update, and delete operations, additional time and disk I/O may be required because the engine balances the index nodes across adjacent pages.

7. Be sure that tracing is turned off, both in the MicroKernel and/or at the ODBC level. Tracing may cause a significant reduction in performance because it can introduce a large amount of disk I/O.

   To ensure ODBC tracing is turned off, start **ODBC Administrator** from Zen Control Center. In ODBC Administrator, click the **Tracing** tab. If tracing is off, you should see a button labeled Start Tracing Now, and thus you should click **Cancel**. If tracing is on, click **Stop Tracing Now**, then click **OK**.

   To ensure MicroKernel tracing is turned off, set the properties configuration **Trace Operation** to **Off (not check marked)**. (Click **Debugging** on the server Properties tree.)

Ensuring Adequate Resource Allocation

If your database server platform has adequate memory and CPU power, you should ensure that your database engine can take full advantage of the available hardware resources to service multiple clients and multiple data files most efficiently.

To configure multiple client and file handling

1. The setting **Number of Input/Output Threads** allows you to specify how many threads are available to handle file operations. (Click **Performance Tuning** on the server Properties tree.)

   As a guideline, the value of this setting should be about 1/8 the number of files the application has open, on average. For example, if the application has 40 files open most of the time, I/O Threads should be set to 5.

   Using Monitor, click **MicroKernel** > **Resource Usage** from the menu. In the window that appears, the **Files:** display shows you current and peak number of files open. You can generate an average reading by recording several Current values over time. Then you can specify an appropriate setting for I/O Threads based on the average value.

Large System Cache

Some Windows operating systems provide a Large System Cache setting that allows the system to take advantage of free memory to cache recently accessed data. By default on certain server

editions, this setting is on, which favors simple file sharing and can result in very aggressive system cache growth.

The aggressive use of memory for file caching can swap out Zen, which can cause a substantial performance issue with the database. You may notice a performance decrease if Large System Cache is on, and the Zen setting System Cache is also set to on (either explicitly or you have set Max MicroKernel Memory Usage to zero). One possible solution to increase database performance is to turn off Large System Cache.

To turn off the cache, access the system properties in the Control Panel or in the properties for My Computer. Click the **Advanced** tab, then the **Settings** button for Performance. Under Performance Options, click the **Advanced** tab.

Under Memory Usage, if the **System Cache** option is selected, Large System Cache is turned on. To turn it off, click the *other* option for Memory Usage: **Programs**. Click **OK** as required to close open dialogs, then restart the operating system for the setting to take effect.

Performance on Hypervisor Products

To achieve the best performance for Zen, ensure the following:

- Adherence to the performance best practices recommendations from the hypervisor vendor.

- The VM hosting Zen has ample memory (RAM).

- The hypervisor host has enough virtual CPUs to minimize virtual CPU contention among all of the VMs on the same machine. This prevents contention with the VM running Zen. If you use affinity rules, ensure that all cores are running on the same socket.

- The Zen data files reside on very fast physical storage and minimize spindle contention and controller contention for the physical storage device.

Database Globalization

This section contains the following topics:

- Overview
- Concepts and Definitions
- Choosing a Character Set and Encoding
- Multilingual Database Support with Unicode UTF-8
- Multilingual Database Support with Unicode UCS-2
- Multilingual Database Support with Legacy and OEM Encodings
- Database Code Page and Client Encoding
- Unicode Support in Zen Utilities
- Support for Collation and Sorting
- Locale Support

Overview

Globalization means, in this context, adapting computer software to different languages. It is now commonplace that data be accessed by users around the globe and that applications present the data in the user's own language. Support for globalization in Zen allows your application to store text in multiple languages in the same database. This means that your application can store, process, and retrieve data in whatever language is required.

This chapter explains the Zen features with which you can support globalization for your applications. It discusses overall approaches to globalization and the particular Zen features that support globalized applications. By default, Zen remains backward compatible with legacy text encodings and this chapter also discusses the settings that ease globalization for applications using legacy encodings.

Concepts and Definitions

This section presents important concepts and definitions of terms used in this chapter.

Character Sets

A **character set** defines the list of text and other symbolic characters that are recognized by a given hardware and software system. For a system that only needs to recognize the characters used in English, the set can be as small as the letters A-Z and a-z, the numerals 0-9 and a few punctuation symbols. Support for additional languages increases the size of the character set. For example, European languages add characters with accents and other diacriticals. Other languages have completely different characters.

Legacy Character Sets

A character is represented in a computer system as a numerical value (getting from a character to a number and back is called encoding and is discussed below). To represent all of the characters in a large character set requires a large range of numerical values. In order to efficiently use what was once limited and expensive storage resources, computer systems have commonly used a byte (8-bits) to store character representations. This limited the size of the character set to 256 characters. Consequently, we now have a legacy of specialized character sets for representing specific languages, such as Japanese, or limited groups of languages, such as Western European. Zen supports a number of these legacy character sets.

The Unicode Character Set

The Unicode standard defines a character set that contains every character used in spoken languages in the world (see www.unicode.org). Unicode also expands the concept of a character set by defining additional annotation information to specify letter spacing, right-to-left behavior, word and line breaks, and so forth. This allows applications to properly display and manipulate Unicode text. Applications, and the database, also need this additional information for such actions as case conversion and sorting.

Zen recognizes the Unicode character set, providing support for character data storage and retrieval in whatever languages are required by the application.

Encoding

Encoding is the association of each character in a character set with a numerical value. Initially, computer systems and system programming languages did not distinguish between characters and

bytes and encoding was simply the byte value corresponding to a particular character. In order to respond to the need to display more characters than would be possible to encode in a single byte, different encodings have been defined. Encodings for larger character sets may use multiple bytes per character.

Legacy Encodings

For legacy character sets, the encoding is defined in a **code page**. You can think of the code page as the lookup table for converting from a character to a value (or a value to a character). It is important that applications that use text always use the same code page. A character that is stored in the database using one code page may be displayed as a different character when read using a different encoding.

The binary unit of all legacy code pages is an 8-bit byte. Most legacy code pages currently in use are supersets of ASCII (American Standard Code for Information Interchange), a 7-bit code defining a character set consisting of 128 control codes and printable characters. By using the eighth bit an additional 128 codes are possible for a total of 256 character encodings in a single byte value.

Microsoft Windows has two groups of code pages, ANSI and OEM code pages. Although code pages in both of these groups are extended ASCII code pages, neither group distinguishes between characters and bytes.

In ANSI code pages, non-ASCII values (those greater than 127) represent international characters that are usually customized for a language or group of languages. ANSI code pages are typically used for byte string applications that use a graphical user interface on Windows systems. An ANSI code page is also sometimes referred to as the active code page (ACP, referred to in Zen as the client code page). A Windows application always has one currently active code page. For example, the default active code page for English on Windows is code page 1252.

OEM (original equipment manufacturer) code pages are, as the name implies, code pages developed by a given manufacturer for specific systems. These code pages were originally used for MS-DOS and are still used for console applications. The usual OEM code page for English is code page 437. A typical OEM code page has a character set similar to an ANSI code page but uses a different character order for values greater than 127.

Unicode Encodings

In character encoding systems, each character is assigned a unique value called a code point which can be used for encoding data. The code points are organized into planes. Each plane can contain 65536 (2^{16}) code points. Unicode has provision for up to 17 planes. The first plane, plane 0, is named the Basic Multilingual Plane (BMP) and contains the majority of the code points

currently defined. At the time of this writing, only planes 0, 1, 2, 15, and 16 contain code points. The Unicode standard has several methods of encoding the code points. Two that are commonly used are UTF-8 and UCS-2. UTF-8 encodes character code point values to a byte string using 1–4 bytes per character. UCS-2 encodes character code point values using 16-bit values, often referred to as wide characters.

Zen recognizes the Basic Multilingual Plane code points and is compatible with applications that use the Unicode encodings UTF-8 for byte strings and UCS-2 for wide character strings. The binary unit for UTF-8 is 8-bit. The binary unit for UCS-2 is 16-bit, wide character.

Declaring Encodings

The database code page is a Zen database property that declares the encoding of character data stored in the database. The purpose is to help insure that character data can be interpreted correctly. However, the database code page property is simply a declaration. Zen does not validate the encoding of the data and metadata that an application inserts into a database. The application is responsible for ensuring that character data is stored and retrieved in a particular encoding. Note that the database code page applies only to text encoded with legacy code pages or UTF-8. Wide character text is encoded using UCS-2. A proper setting is required for the engine to convert between wide character text and byte-string text. The default value of the database code page is the system code page of the operating system where the engine is running.

In Zen the SQL access methods infer a **client code page** for byte strings exchanged between the application and the access method. On Windows, the access method assumes that the application is respecting the active code page (ACP) for byte strings. On Linux and Raspbian, the access method assumes that the application is respecting the encoding of the locale, which is usually set in the LANG environment variable.

Zen provides methods to ensure compatible encoding between the database engine and clients. For example, an application can specify that it wants the Zen SQL client to translate data automatically between the database code page and the client application. This is referred to as *automatic translation*. Note, however, that automatic translation can translate characters only if they are present in the character sets of both code pages (the code page on the server machine and the code page on the client machine).

For backward compatibility, automatic translation in access methods is disabled by default. The application must configure the access method to enable automatic translation. When possible, the recommended method is to set the database code page and configure the access method to read and use that value.

Collation and Sorting

Collation is the general term for the process and function of determining the sorting order of strings of characters. Collation varies by language, so it is not possible to arrange the encoding in multilanguage code pages so that simple binary string comparison produces the desired collation order for every language. If multilevel sorting is a requirement, separate data tables are required to define the correct sorting order.

Choosing a Character Set and Encoding

To implement a globalization strategy you typically begin by identifying the character set required based on the languages or other text and character requirements you need to satisfy. The next step is to choose the encodings that support the character set. The encoding used may even be different for the database and for client applications. Let's look at some examples.

The most global character set is the Unicode character set. Even if legacy character sets are used in the clients, they can all be translated to Unicode for storage in Zen. For a new application or a new module, storing text in UCS-2 or UTF-8 encoding is a simple approach. However, not all applications are new.

Another consideration for applications is the technology of the client programs. If the application uses the .NET framework, the Java VM, or the UNICODE option with C/C++, the application is already processing text using wide character strings. In these situations the main consideration is configuring Zen to preserve that text and choosing how to store it.

If the application is using byte strings in C/C++ and the legacy Zen ODBC driver, there are two possible paths to globalization. One is to port the application to use wide character strings; the other is to let the application continue to support the legacy code page of the client where it is installed and to arrange for translation to Unicode storage.

A very conservative approach for existing applications is to continue using your current legacy code page and take advantage of the other languages that it supports. For example, an application developed for the English-speaking market on Windows using ANSI code page 1252 or OEM code page 850 can also support Western European languages without any change in application storage. The main changes would be to localize program text.

Note: User Data and Metadata. Zen has two types of text that it must handle. The first is user data, which is mostly manipulated by the application, and also by index ordering and by SQL string functions. The second type is metadata, which is the names of SQL objects, such as tables, columns, and indexes. Metadata does not handle UCS-2 encoding, and so follows the legacy code page of the database code page declaration. SQL queries can contain both user data in string literals, and metadata in object names. Thus when discussing SQL queries, we must distinguish the character sets of user data and metadata, even when we are using one of the Unicode encodings for the SQL text as a whole.

Zen is not prepared to handle mixed encodings in text storage. The application should consider such text to be BINARY storage and handle all encoding translations in the application. Zen assumes that all CHAR type data and SQL metadata respect the database code pag, and that all NCHAR data is UCS-2.

The following topics cover specific storage cases:

- Multilingual Database Support with Unicode UTF-8

- Multilingual Database Support with Unicode UCS-2

In addition, the following topic covers handling of legacy OEM code pages:

- Multilingual Database Support with Legacy and OEM Encodings

Multilingual Database Support with Unicode UTF-8

If you choose to store text as UTF-8 you will continue to use the CHAR, VARCHAR, and LONGVARCHAR relational types. You also need to consider such aspects as the Unicode support for the operating system on which your application runs, the string manipulation libraries available to your application, the Zen access methods your application uses, any columns that may need a different data type, and so forth.

When to Use Unicode UTF-8

Unicode UTF-8 encoding is a good choice for the following:

- You want to add new language support to an existing application but keep application changes fairly minimal. For example, you have a Zen database with ANSI-only characters (English, for instance). You want to extend your application to include data in English, German, Polish, and Czech. UTF-8 provides compact storage requirements for European scripts because it requires, at most, two bytes per character.

- Web applications, since many web platforms use UTF-8. Because Unicode UTF-8 is ASCII-compatible and compact for Latin-based language character sets, it is often used as a standard encoding for interchange of Unicode text.

- A Linux application that supports UTF-8 string handling

Unicode UTF-8 Support in Zen

One of the code pages supported by Zen is UTF-8. For UTF-8 text storage, you would set the DB code page for your Zen database to UTF-8.

Note that with UTF-8, string storage is byte strings. For byte strings, Zen provides the relational data types CHAR, VARCHAR, and LONGVARCHAR, and the Btrieve data types STRING and ZSTRING. See also Data Types in *SQL Engine Reference*. Columns will likely be wider when storing UTF-8 because European languages often require two bytes per character instead of a single byte for legacy code pages.

All string data inserted by your application for existing CHAR, VARCHAR and LONGVARCHAR data types are interpreted as UTF-8 strings. You can configure the Zen SQL access methods to automatically translate to UTF-8 (see Access Methods for Unicode UTF-8 Support).

When the database code page is UTF-8 and the client environment supports Unicode (wide character or UTF-8), SQL text supports Unicode characters in CHAR literals. With any other database code page, general Unicode characters must be in NCHAR literals.

Collation and Sorting

By default, Zen supports code point order for collation and sorting with UTF-8 storage.

Access Methods for Unicode UTF-8 Support

The Zen access methods ODBC, JDBC, and ADO.NET support translation to UTF-8 storage. These access methods exchange text values with the application as UCS-2 wide character strings or as legacy byte strings for the ANSI ODBC drivers. When properly configured, the access methods translate the application text values to UTF-8 for transmission to the storage engine.

- If your application uses the ANSI ODBC driver on Windows, all data will be converted by the Windows Driver Manager to the client legacy code page for byte strings. This results in the loss of any characters that are not in the legacy character set. You may also need to convert your application to use the Unicode ODBC driver.

- If your application uses the ANSI ODBC driver on Linux, you should set the app locale to use UTF-8 as the string encoding. For completeness, also declare `pvtranslate=auto` in the connection string and declare the database code page to be UTF-8.

- For JDBC, your application needs to specify pvtranslate=auto in the connection string to the JDBC driver. See Connection String Overview in *JDBC Driver Guide*.

- For ADO.NET, your application needs to specify `pvtranslate=auto` in the connection string to the database engine. See Adding Connections in *Data Providers for ADO.NET Guide*.

Migrating an Existing Database to Unicode UTF-8

All text data must be converted from any legacy code page to UTF-8. Columns will likely need to be widened to accommodate the longer UTF-8 byte strings. Any non-ASCII metadata, such as table names, must be converted from the legacy code page to UTF-8. Given these combined changes, it is reasonable to migrate the database by copying from the old schema, using the legacy code page, to the new schema with UTF-8 as the database code page.

Note: In the special case where all existing data and metadata is pure ASCII, it is possible to just change the database code page to UTF-8. All existing (7-bit) ASCII byte strings are also valid UTF-8 byte strings.

Multilingual Database Support with Unicode UCS-2

If you choose to store text as UCS-2 you will use the NCHAR, NVARCHAR, and NLONGVARCHAR relational types. This has no effect on your ability to also store other text in the CHAR family of relational types. You also need to consider such aspects as the Unicode support for the operating system on which your application runs, the string manipulation libraries available to your application, the Zen access methods your application uses, any columns that may need a different data type, and so forth.

When to Use Unicode UCS-2

Unicode UCS-2 is a good option for the following situations:

- Your application supports Asian character data. With UCS-2, all characters are stored as two bytes which provides more compact storage than UTF-8 for Asian character data.

- Your application will not be globalizing all of the storage and so will have wide character columns together with legacy byte string columns.

- Your application needs better compatibility for wide character data with Java, ADO.NET, or wide character clients. Such applications use UCS-2 for application strings.

Unicode UCS-2 Support in Zen

Wide character storage is a separate type from byte-string storage and may be used alongside byte-string storage.

For wide character strings, Zen provides the relational data types NCHAR, NVARCHAR, and NLONGVARCHAR, and the Btrieve data types WSTRING and WZSTRING. All data inserted by your application for NCHAR, NVARCHAR, NLONGVARCHAR, WSTRING and WZSTRING data types is interpreted as wide character strings.

Zen supports Unicode characters in NCHAR literals in SQL query text. Text data in CHAR literals is translated to the database code page. Keep in mind that database code pages other than UTF-8 cannot handle most Unicode characters.

Collation and Sorting with Unicode UCS-2

By default, Zen supports code point order for collation and sorting with UCS-2 storage.

Access Methods for Unicode UCS-2 Support

Using NCHAR storage with the SQL access methods requires that applications specify the appropriate NCHAR SQL types to the access method. Use of CHAR types can cause conversion from NCHAR to CHAR and loss of data if the database code page is not UTF-8.

*   ODBC applications should use the SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR SQL types for parameters.

    *   Windows ODBC applications should be using wide character strings, identified to ODBC as SQL_C_WCHAR. Also, Windows ODBC applications should use the Zen Unicode ODBC driver. Use of the legacy ANSI ODBC driver will result in loss of data because the Windows Driver Manager will convert all wide character strings to byte string.

    *   Linux and Raspbian ODBC applications should use a locale with a UTF-8 encoding and use UTF-8 in application byte strings (SQL_C_CHAR) to provide data for NCHAR values (SQL_WCHAR).

*   JDBC applications should set string parameters to type NVARCHAR or use the 'N' methods such as setNString() to send string data as an NCHAR SQL type. Also, set pvtranslate=auto in the connection string and set the database code page. Otherwise, JDBC will remain backward compatible and use the declared or default byte-string encoding for string data. See Connection String Overview in *JDBC Driver Guide*.

*   ADO.NET applications need to specify pvtranslate=auto in the connection string to the database engine. See Adding Connections in *Data Providers for ADO.NET Guide*.

In all cases, use NCHAR literals in SQL text.

Migrating an Existing Database to Unicode UCS-2

The tasks for migrating an existing database to support a UCS-2 application involve converting byte strings to wide character strings. The extent of the conversion depends on your application. ALTER TABLE can be used to convert columns provided that the database code page is set to match the encoding for existing CHAR columns. It is only necessary to convert columns that will store globalized data. Columns that will never store characters outside of the character set of the database code page can remain as CHAR types.

If your application uses the ANSI ODBC driver on Windows, you need to convert your application to wide character data and use the Zen Unicode ODBC driver.

Multilingual Database Support with Legacy and OEM Encodings

If you choose to store text using a legacy or OEM code page, you will use the CHAR, VARCHAR, and LONGVARCHAR relational types. You may use wide character applications if you configure the access method to do conversion to your legacy code page.

When to Use a Legacy Code Page

Using a legacy code page is a compact and efficient way to store text data provided that the legacy code page defines a character set that meets the needs of your application.

Legacy Code Page Support in Zen

The Zen database engine assumes that CHAR data is encoded using the database code page. The access methods have configuration options that help ensure that this is the case.

Older applications that use OEM code pages for storage did not declare the database code page. You can continue to work with this situation so long as the application is careful to use the appropriate relational string functions.

Collation and Sorting with Legacy Code Pages

The default collation in Zen is binary ordering of the encoded bytes. The declared database code page does not affect this default collation.

Zen provides both alternate collating sequence (ACS) and international sorting rules (ISR) mechanisms for controlling collation. These may be declared on Btrieve indexes and on relational columns. Zen will always use a particular ACS for the ASCII character set when using the CASE declaration on a relational column or the case-insensitive bit on a Btrieve index.

Access Methods for Legacy Code Pages

All Zen access methods support byte-string text. Some assume that the database code page is the same as the application's code page, and some allow configuration.

- The ANSI ODBC drivers provide a DSN setting, OEM/ANSI translation, which declares that the application OEM code page (as distinct from ANSI code page) is the database code page. The ANSI ODBC drivers also support the encoding property in connection strings and the pvtranslate=auto option.

- The Unicode ODBC driver always operates as if the pvtranslate_true property is set and translates application CHAR data to the database code page as needed.

- The JDBC driver has an explicit encoding property in the connection string. The driver will set encoding to the database code page if the pvtranslage=auto property is given.

- The ADO.NET provider has an explicit encoding connection property. The driver will set Encoding to the database code page if the PvTranslate=Auto connection property is given.

- The PDAC access method for Delphi has the OEMConversion property that controls whether CHAR data is sent to the engine using the OEM code page or the ANSI (ACP) code page.

Migrating an Existing Database to a Different Legacy Code Page

Changing the code page of a database requires copying the data to a new database that uses the new code page. The copy must be done by your application. Zen does not translate code pages within a transaction.

If your database metadata use characters that require translation, those changes must be made when creating the schema of the new database. Table names are a common example of this type of metadata.

Database Code Page and Client Encoding

As discussed in Concepts and Definitions, above, encoding specifies how character data is translated between the Zen database engine and a Zen client application. Zen handles much of the complexity of the encoding between client and server and the various combinations of operating system, languages, and access method. The encoding enhancements are divided into database code page and client encoding. The two types of encoding are separate but interrelated.

Database code page and client encoding apply only to the Relational Engine. The MicroKernel Engine is not affected.

Database Code Page

The database code page is a database property that specifies the encoding used for character data stored in the database. The default database code page is "server default," meaning the operating system code page on the server where the database engine is running. (The operating system code page is generally referred to as the "OS encoding," which is the phrase used throughout this chapter.)

Database code page is particularly handy if you need to manually copy Zen DDFs to another platform with a different OS encoding and still have the metadata correctly interpreted by the database engine.

When you create a database with Zen, the default is to use the active code page for the machine on which the database engine is running. For example, on a machine running Windows for English, the code page is assigned to 1252. Zen encoding translation is set to None. Your application must either use the same code page as the Zen database, or ensure that encoding translation is set to Automatic.

Supported Code Pages

Zen supports the following code pages. All of the pages listed use byte string storage.

- ASCII
- EUCJP
- ISO8859_1
- UTF-8
- Windows Code Pages
  - CP437, CP737, CP775, CP850, CP852, CP855, CP857, CP858, CP862, CP866, CP932

- CP1250, CP1251, CP1252, CP1253, CP1254, CP1255, CP1256, CP1257, CP1258

Client Encoding

Client encoding is the data encoding used by an application on a Zen client. An application can manage text in any encoding it chooses. A compatible encoding must be established between the database engine and the client application.

Zen can automatically translate between different encodings used by the database engine and clients provided the characters are present in both the code page on the server machine and the code page on the client machine.

Data translation, if required, occurs at the client. Translation is not always required – for example, when client and server OS encoding match.

Encoding Support in ZenCC

Note: You can use ZenCC to set the database code page when you create a database or to modify the code page setting for an existing database.

Note: Changing the database code page property does not change any data in the database. However, changing the database code page for an existing database will affect how existing data entries are interpreted.

Zen Control Center (ZenCC) is, itself, a client application to the database engine. As a client, ZenCC lets you specify the encoding to use for each database session when ZenCC reads and inserts metadata and data. The default for an existing database is to use the encoding of the machine where ZenCC is running. This is the legacy behavior of ZenCC. The default for a new database is to use automatic translation. See ZenCC Connection Encoding in *Zen User's Guide*.

The following table explains the interaction between the settings for ZenCC connection encoding and database code page. ZenCC connection encoding applies only to ZenCC. It has no effect on other client applications.

ZenCC Connection Encoding Set to a Specific Encoding	ZenCC Connection Encoding Set to Automatic Translation
ZenCC ignores the database code page and uses the encoding specified to read and insert CHAR data, string literals, and metadata. NCHAR data is not affected by this setting. This is the legacy behavior of ZenCC.	ZenCC and the database automatically establish the encoding for CHAR data and metadata. String literals in queries are sent to the engine as Unicode. NCHAR data is not affected by this setting.

Encoding Support in Btrieve API

When using the Btrieve API, you must provide file names and paths in the local encoding used in your application. The Btrieve API handles the differences between OS encoding on the server and client.

Encoding Support in DTI

When using the Distributed Tuning Interface (DTI), you must provide file names and paths in the local encoding used in your application. DTI handles the differences between OS encoding on the server and client.

If you use the DTI API to create a database, you may specify the database code page property at the time of creation. This property may be used by SQL access methods to configure automatic translation of character data.

Encoding Support in ADO.NET

The .NET Framework and .NET applications use UTF-16 strings. These must be translated to a code page when storing text in CHAR columns.

The connection property PVTranslate=Auto sets the connection encoding to the database code page. It is also possible to set the encoding property directly.

For more information, see Adding Connections, PsqlConnectionStringBuilder Object and Character Set Conversions in *Data Providers for ADO.NET Guide*.

Encoding Support in JDBC

The Java Virtual Machine and Java applications use UTF-16 strings. These must be translated to a code page when storing text in CHAR columns.

The connection property PVTranslate=Auto will set the connection encoding to the database code page. It is also possible to set the encoding property directly.

When the PvTranslate=Auto property is set, the JDBC driver will send string literals to the engine as Unicode. Without this setting, the legacy behavior is to translate string literals to the database code page. If your application uses NCHAR string literals (e.g., "N'ABC'"), it should set the PvTranslate=Auto connection property.

See Connection String Elements in *JDBC Driver Guide*.

Encoding Support in ODBC

The Zen ODBC drivers support a number of mechanisms to control client encoding.

When configuring a DSN, it is possible to select the encoding options Automatic, OEM/ANSI, and None. The Automatic setting causes the driver to translate from the client encoding to the database code page. The OEM/ANSI setting causes the driver to translate from the client encoding to the corresponding OEM code page. The None setting prevents the driver from doing any text translation. See Encoding Translation in *ODBC Guide* for more details.

Legacy Conversion Methods for OEM-to-ANSI Data

If a database has OEM character data in it, a legacy solution is to specify OEM/ANSI conversion in the access method. This topic discusses some legacy methods for Linux clients using OEM character data.

Note: While the legacy methods are still supported, the recommendation is to specify the OEM code page for the database and have the access methods use automatic translation as discussed above.

See also OEM/ANSI Conversion in *ODBC Guide*.

When using ODBC, Win32 encoding is expected to be SHIFT-JIS.

Japanese versions of Linux by default have their encodings typically set to EUC-JP or UTF-8.

When using Japanese versions of Linux, a client can connect to another Linux server (for example, locally) or to a Win32 SHIFT-JIS server. It is also possible to connect to a database encoded in SHIFT-JIS but located on a Linux server.

Use the following instructions for your listed configuration. In each case, it is assumed that the application itself does not do any conversion and uses the encoding that is native for the machine.

- Connecting a Linux EUC-JP Client to a Win32 SHIFT-JIS Server
- Connecting a Linux UTF-8 Client to a Win32 SHIFT-JIS Server
- Connecting a Linux EUC-JP Client to a Linux EUC-JP Server
- Connecting a Linux UTF-8 Client to a Linux UTF-8 Server
- Connecting a Linux UTF-8 Client to a Linux EUC-JP Server
- Connecting a Linux EUC-JP Client to a Linux EUC-JP Server, with SHIFT-JIS Encoding Used to Store Data on the Server

Connecting a Linux EUC–JP Client to a Win32 SHIFT–JIS Server

The server requires that everything is received as SHIFT-JIS. The client requires that the server send everything as EUC-JP.

To accomplish this, the client DSN settings in ODBC.INI (located by default in `/usr/local/actianzen/etc`) used to connect to the given database should be set up as follows:

```
[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
```

The TranslationDLL line sets the translation library for the ODBC client interface to use.

The TranslationOption line specifies that translation is needed from 9000 (EUC-JP) to 0932 (SHIFT-JIS).

Using this example, all data coming from the client will be translated to SHIFT-JIS before it is sent to the server, and to EUC-JP before the data is sent back to the client.

Connecting a Linux UTF–8 Client to a Win32 SHIFT–JIS Server

The server requires that everything is received as SHIFT-JIS. The client requires that the server send everything as UTF-8.

To accomplish this, the client DSN settings in ODBC.INI (by default in /usr/local/actianzen/etc) used to connect to the given database should be set up as follows:

```
[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
```

The TranslationDLL line sets the translation library for the ODBC client interface to use.

The TranslationOption line specifies that translation is needed from 9001 (UTF-8) to 0932 (SHIFT-JIS).

Using this example, all data coming from the client will be translated to SHIFT-JIS before it is sent to the server, and to UTF-8 before the data is sent back to the client.

Connecting a Linux EUC-JP Client to a Linux EUC-JP Server

Using this configuration, no changes to the DSN description are needed. Use the DSN as it was created by the **dsnadd** tool.

Connecting a Linux UTF-8 Client to a Linux UTF-8 Server

Using this configuration, no changes to the DSN description are needed. Use the DSN as it was created by the **dsnadd** tool. See dsnadd in *Zen User's Guide*.

Connecting a Linux UTF-8 Client to a Linux EUC-JP Server

The server requires that everything is received as EUC-JP. The client requires that server send everything as UTF-8.

To accomplish this, the client DSN settings in ODBC.INI (by default in /usr/local/actianzen/etc) used to connect to the given database should be set up as follows:

```
[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
```

The TranslationDLL line sets the translation library for the ODBC client interface to use.

The TranslationOption line specifies that translation is needed from 9001 (EUC-JP) to 9000 (UTF-8).

Using this example, all data coming from the client will be translated to EUC-JP before it is sent to the server, and to UTF-8 before the data is sent back to the client.

Connecting a Linux EUC-JP Client to a Linux EUC-JP Server, with SHIFT-JIS Encoding Used to Store Data on the Server

This situation is possible if you have a SHIFT-JIS database on a Win32 engine, and you want to move all the files to the Linux EUC-JP server. In this case, the database resides on a EUC-JP Linux machine, but all the data inside the DDF files and data files are in SHIFT-JIS.

In this case, your DSN should be set up as follows:

```
[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
```

The last line specifies that even though the server uses EUC-JP encoding, it should treat the data on the server as SHIFT-JIS.

Encoding Support for Wide ODBC Driver

Zen supports UCS-2 with ODBC with a driver for wide character data and defaults for DSN encoding translation. See Encoding Translation and ODBC Connection Strings in *ODBC Guide*.

ODBC Driver for Applications with Wide Character Data

Zen provides an ODBC driver for 32-bit and 64-bit applications that use wide character data. The driver is for Windows operating systems only and is an addition to the previous set of drivers.

Driver Name	Discussion
Zen ODBC Unicode Interface	• Connects to a local or remote named database. • With the 32-bit ODBC Administrator, creates 32-bit DSNs for use by 32-bit applications that use wide character data. The 32-bit driver is installed with all Zen editions. • With the 64-bit ODBC Administrator, creates 64-bit DSNs for use by 64-bit applications that use wide character data. The 64-bit driver is installed with all Zen editions when installing on a 64-bit platform.

On Linux, the system encoding is usually UTF-8, which allows SQL text to contain any Unicode character code point. The Zen ODBC Unicode Interface driver is not available on Linux because an application can use the Zen ODBC Client Interface driver with UTF-8. A Linux application can handle wide character data either as UTF-16 strings (SQL_C_WCHAR) or request conversion to the system encoding (usually UTF-8) as SQL_C_CHAR. SQL text using UTF-8 is compatible with the existing Pervasive ODBC Client Interface driver so an additional ODBC driver on Linux is not required.

Default for DSN Encoding Translation

The encoding translation options for a DSN specify how character data is translated between the Zen database engine and a Zen client application that uses ODBC. The default for encoding translation depends on the Zen ODBC driver that you use.

Driver Name	Encoding Translation Default	Remarks
Zen ODBC Unicode Interface	Automatic	The connection string parameter Pvtranslate also defaults to "auto."
Zen ODBC Interface	None	Same default as the previous version of Zen.
Zen ODBC Client Interface	None	Same default as the previous version of Zen.
Zen ODBC Engine Interface	None	Same default as the previous version of Zen.

The ODBC drivers process SQL text differently depending on the driver and the setting for the DSN encoding translation.

Setting	Processing of Incoming SQL Text	Zen Driver		
		ODBC Unicode Interface	ODBC Interface and ODBC Client Interface	ODBC Engine Interface
Automatic	SQL text gets converted to UTF-8, then sent to the database engine. The code pages for client, server, and database are ignored.	Yes[1]	No	No
	SQL text gets converted to the database code page then sent to the database engine.	No	Yes	Yes
None	SQL text is not translated between the client and database engine.[2]	Yes	Yes	Yes
OEM/ ANSI	SQL text in the client code page is converted to the OEM/ANSI encoding and then sent to the database engine.	Yes[3]	Yes[3]	Yes[3]

Setting	Processing of Incoming SQL Text	Zen Driver		
		ODBC Unicode Interface	ODBC Interface and ODBC Client Interface	ODBC Engine Interface

[1] With the encoding translation set to Automatic, you can use NCHAR columns and NCHAR literals with wide character data.

[2] The assumption is that the client and the database engine use the same operating system encoding.

[3] If the SQL text is wide character, it is first converted to the client encoding. If the SQL text is not wide character, it is is already in the client encoding. The SQL text is then converted to the OEM encoding and sent to the database engine.

Unicode Support in Zen Utilities

Unicode Support in Zen Control Center

See also the Encoding Support in ZenCC.

Dialogs for Opening and Saving Files

The ZenCC dialogs for opening and saving SQL documents, saving exported schemas, and importing and exporting table data have all been enhanced to accommodate a variety of file encodings. Previously, these files were presumed to be in the default system code page. It is not possible to select a number of Unicode encodings when saving files. When opening a file, the new dialogs detect whether the file uses a byte order mark (BOM) to identify the Unicode encoding. The opening dialogs also allow you to set the expected encoding of the file. For your convenience, a new ZenCC setting controls the default encoding used in these dialogs.

For more information on these new features, see *Zen User's Guide* under the topics Dialogs for File Open and File Save, Wide Character Data Support for Import Data, Export Data, and Export Schema, and File Encoding Preferences.

Bulk Data Utility (BDU)

The Bulk Data Utility (BDU) is a command line tool that allows you to load data from a delimited text file into a Zen table. A command line parameter, -c *encoding*, is provided to specify the data encoding to use when loading the data file. The encoding options are UTF-8, UTF-16LE, and UTF-16BE. If a data file contains a byte order mark (BOM), BDU uses the encoding specified by the BOM. That is, if a data file uses a BOM to indicate an encoding of UTF-8, UTF-16LE, or UTF-16BE, BDU uses that encoding regardless of what value you specify for the encoding parameter on the command line. Without a BOM or the -c parameter, BDU defaults to using the system code page.

See bdu in *Zen User's Guide*.

Support for Collation and Sorting

What Is Collation and Sorting?

Collation refers to the ordering of a set of binary values matched with characters in a code page. Collations can differ in important ways. For example, one code page may put digits before letters and another may put them after. Sorting is the rearrangement of data so that text is in collation order.

Zen supports the specification of a named collation on byte-string text segments. Indexes then sort the record keys according to the specified collation.

Sort Order with No Collation Sequence Specified

When no collation is specified, Zen by default sorts characters in code point order. The default is ascending order from lowest value to highest. You can change this to descending order. See Sort Order in *Zen Programmer's Guide* for more information.

Collation Support in Wide Character Columns

Zen supports the default collation of Unicode data according to code point order. In addition to working with UTF-16, it also can sort multibyte UTF-8 text in code point order.

Collation Support Using an Alternate Collating Sequence (ACS)

You can specify an alternative to the default code page collation order. This user-defined alternate collating sequence or ACS is a mapping between the code page collation order and the desired collation order. You can define one or more alternate sequences for determining the collation of string keys of type STRING, LSTRING, and ZSTRING. For example, you can use a user-defined ACS to specify a collation that places numbers after letters or changes the ordering of upper- and lowercase letters. Zen comes with an ACS named upper.alt that maps the lowercase letters to uppercase letters to sort as equivalent. This result could also be achieved by setting case insensitivity but the example shows what can be done with an ACS.

Essentially, the user-defined ACS is a table that associates the code page sequence position for a character with the alternate desired sequence position. Creating an ACS is described in Alternate Collating Sequences in *Zen Programmer's Guide*, including examples. You specify the ACS for key value fields in the definition of the layout of the data file. See Specifying a Alternate Collating Sequence for a Key in this guide and Data Layout in *Zen Programmer's Guide*.

For additional information about setting an ACS, see Create (14), Create Index (31) and Get Next Extended (36) in *Btrieve API Guide*, Alternate Collating Sequence (ACS) Files in *DDF Builder User's Guide* and SET DEFAULTCOLLATE in *SQL Engine Reference*.

Collation Support Using an International Sort Rule (ISR)

Another type of ACS is an international sort rule or ISR. An ISR is a predefined alternate collating sequence for language-specific sort orders. You can use an ISR to correctly sort languages such as German with the letters ä, ö, ü (sorted as ae, oe, ue) and ß (sorted as ss). Zen provides a number of ISR tables in the collate.cfg file in your Zen installation. Examples of their use can be found in Sample Collations Using International Sorting Rules in *Zen Programmer's Guide*. See the references for alternate collating sequences, above, for more information.

Collation Support Using an ICU Unicode Collation

Zen supports two Unicode collations for use with UTF-8 or UTF-16 data if you need sorting other than the default binary collation. These alternate Unicode collations are based on the International Components for Unicode (ICU) libraries, release version 54. The following table summarizes the collations.

ICU Collation Name	Installed File	Description
u54-msft_enus_0	u54-msft_enus_0.txt	Emulates the ISR collation MSFT_ENUS01252_0. The emulation applies only to the 1252 subset of Unicode. Characters outside this range are sorted according to the ICU root collation.
root	icudt54l.dat	Defines default ICU collation and other configuration data.

ICU collations are used like ISR table names, except that instead of having names starting with `PVSW_` or `MSFT_`, their names must have the prefix `u54-` or be simply `root`. In addition, these collations can be applied only to the following Unicode data types:

- STRING (assumed to be UTF-8)

- ZSTRING (assumed to be UTF-8)

- WSTRING

- WZSTRING

A default Zen installation provides the two ICU collations in the same location as the collate.cfg file. For generic sorting, the default ICU collation is referred to by the name root, and its configuration data resides in the file icudt54l.dat. For locale-specific sorting, supplemental data resides in files whose names start with u54 and end with a .txt extension. Zen currently supports one supplemental ICU collation.

For more information about ICU collations, see the [ICU Project website](.).

Locale Support

An important aspect of globalization is locale, which is a model and definition of a native-language environment. A locale consists of a number of categories for which country-dependent formatting or other specifications exist. For example, a locale defines date and time formatting conventions, monetary conventions, decimal formatting conventions, and collation order. Depending on the operating system, a locale may be called a region.

More than one locale can be associated with a particular language, which allows for regional differences. For example, English can have a United States locale and a Great Britain locale.

When executing string functions, Zen uses the locale of the operating system where the database engine is running. Zen uses the locale of the client when converting data types as requested by the application through one of the Zen access methods.

For more information, see SET DECIMALSEPARATORCOMMA, Comma as Decimal Separator and SET TIME ZONE in *SQL Engine Reference*.

Setting Up Referential Integrity

Referential integrity is a system of checks and balances that you can create in your database to ensure that tables with related data remain synchronized.

- Concepts of Referential Integrity
- Setting up Primary Keys
- Setting up Foreign Keys
- Interactions Between Btrieve and Relational Constraints

Concepts of Referential Integrity

Referential integrity (RI) allows you to modify or prohibit updates, inserts, or deletes based on whether identical field values exist in the same or other tables.

Definitions

An understanding of RI depends the concepts of rule, primary key, foreign key, cascade rule, and restrict rule. This topic gives their definitions.

Rule

A *rule* is a simple statement of cause and effect, carried out by the RI system defined in the database.

Example A

For example, a *delete rule* defines what happens to records containing a foreign key when a record containing a primary key is deleted: "When the record containing 'Bhargava Building' is deleted, all rows in Table A that reference that record are deleted."

A delete rule can also prohibit the row containing the primary key value from being deleted if there are any foreign key values that reference the given primary key value.

Example B

An *update rule* defines what happens to a record containing a foreign key when a user attempts to update the record or add a new record: "When a user attempts to insert a new record to Table B, reject the attempt if the building name does not exist in Table C."

Primary key

A *primary key* is the column or columns upon which a rule depends. Only one primary key is permitted in any table, and the primary key must not allow duplicate values. For an update rule, the primary key is the column or columns against which updated or inserted columns are compared to determine if the updated or inserted record should be allowed.

In Example A, the column containing "Bhargava Building" is the primary key.

In Example B, the column in Table C that contains the building name is the primary key.

Foreign key

A *foreign key* is the column or columns that are compared against a primary key to determine how to proceed.

In Example A above, the column in Table A that may contain the value "Bhargava Building" is the foreign key.

In Example B above, the column in Table B that contains the building name is the foreign key.

Cascade Rule

A *cascade* rule is a rule in which the database permits the desired operation to occur, then enforces RI by changing other tables or rows to synchronize with the first operation. For example, if a *delete cascade rule* is defined, deleting a record in the primary key table causes the database to find and delete all rows throughout the database that have foreign key values the same as the primary key value of the deleted row.

Restrict Rule

A restrict rule is a rule in which the database decides whether or not to permit the desired operation based on existing values in the database. For example, if an *update restrict rule* is defined, an attempt to add a row to a table containing a foreign key causes the database engine to compare the value in the foreign key field to the values in the primary key. If there is no primary key row with the same value, the new row is not permitted to be added to the foreign key table.

Understanding Keys and Rules

This topic explores the concepts behind primary keys and foreign keys in further detail.

Table A		Table B	
student_ID	Name	stud_ID	Class
20543	John	20543	ENG-101
20577	Mary	20543	AST-202

In the example shown above, the column named student_ID in Table A (A.student_ID) is an IDENTITY data type that does not allow two rows to the have the same value. Every student has a unique ID number. We will define student_ID as the primary key of Table A.

We can then define the column named stud_ID in Table B (B.stud_ID) as a foreign key that references A.student_ID. Note that the data type of stud_ID must be a type that can be compared with IDENTITY, such as INTEGER. The data types of primary and foreign keys must be compatible. You can have as many foreign keys as you need in order to enforce your desired referential integrity scheme. Multiple foreign keys can reference the same primary key.

The table with the primary key can be referred to as the parent table, while the table with the foreign key is called the child table. Once the keys are defined, we have a range of behaviors to choose from, as shown in the following table. You can define as many rules as fit your needs, but you can only have one of each type. For example, if you define a delete restrict rule, you cannot define a delete cascade rule on the same keys, because the two behaviors are mutually exclusive.

If you want this behavior...	... define this rule:
Do not allow a row to be inserted or updated in Table B unless the proposed value of B.stud_ID matches any value in A.student_ID.	Update Restrict
Do not allow a row to be deleted from Table A if any value of B.stud_ID matches that row.	Delete Restrict
If a row is deleted from Table A, delete all rows from Table B in which B.stud_ID matches the value of A.student_ID in the deleted row.	Delete Cascade

Update Restrict

Continuing with the example, setting an update restrict rule ensures that the value of B.stud_ID in any new or updated row must first exist in A.student_ID. It follows, then, that you must have rows in Table A before you can add any rows in Table B. Stated another way, you must create at least one parent row before you can create a child row.

Delete Restrict

In the example, setting a delete restrict rule ensures that a row from Table A cannot be deleted if any rows in Table B reference that row. You cannot delete the row with Name value "John" because John's student ID is referenced in Table B.

Once all rows from Table B that reference John's student ID are deleted, then John's row can be deleted from Table A.

Delete Cascade

In the example, setting a delete cascade rule ensures that both records in Table B are deleted if the row with Name value "John" is deleted.

Zen allows a circular delete cascade on a table that references itself. Because of this, use delete cascade with caution. Ensure that you do not inadvertently delete all records in the parent table, the child table, or both.

An example helps clarify how such cascading deletion could occur. Suppose that you create the following table, d3, with two columns:

```
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)
```

You then alter the table to add a foreign key with a delete cascade rule:

```
ALTER TABLE d3 ADD FOREIGN KEY (c2) REFERENCES d3 ON DELETE CASCASE
```

The following statement deletes *all* rows in the table:

```
DELETE FROM d3 WHERE c1 = 2
```

Why are all rows deleted instead of just the row where c1 =2?

Delete cascade deletes any row with a foreign key equal to the primary key that is deleted. The second row has a foreign key relationship to the first row. Similarly, the third row has a foreign key relationship with the third, and the fourth row with the third. Because of the foreign key relationships, the delete cascade rule traversed all of the rows, causing the second row to be deleted because of the first, the third because of the second, and the fourth because of the third.

Zen does **not** allow circular delete cascade on tables that reference each other. For example, consider the following scenario in which you have tables d1 and d2:

```
CREATE TABLE d1 (c1 INT PRIMARY KEY, c2 INT)
CREATE TABLE d2 (e1 INT PRIMARY KEY, e2 INT)
```

The following alter statement is allowed:

```
ALTER TABLE d1 ADD FOREIGN KEY (c2) REFERENCES d2 ON DELETE CASCADE
```

The following alter statement is **not** allowed because tables d1 and d2 already have a delete cascade relationship:

```
ALTER TABLE d2 ADD FOREIGN KEY (e2) REFERENCES d1 ON DELETE CASCADE
```

Setting up Primary Keys

You can create primary keys using SQL statements or Zen Control Center. See Columns Tasks in *Zen User's Guide*.

Creating a Primary Key During Table Creation

You can create a primary key when you create a table, by using the PRIMARY KEY keywords in your CREATE TABLE statement. A primary key can consist of one or more columns. The following example shows the column named id being created then being designated the primary key:

```
CREATE TABLE mytable (id INTEGER,
    myname CHAR(20),
    PRIMARY KEY(id))
```

The next example shows how to create a primary key using more than one column as the unique key value:

```
CREATE TABLE mytable (id INTEGER,
    myname CHAR(20),
    PRIMARY KEY(id, myname))
```

Regardless of whether you specify the UNIQUE attribute on the column or columns that you designate as a primary key, the database engine automatically creates an index on the designated columns that does not allow duplicate values or null values in the columns. Null values are never allowed in a key column. Every primary key value must be unique.

For more examples, see CREATE TABLE in *SQL Engine Reference*.

Adding a Primary Key to an Existing Table

You can add a primary key to an existing table through ZenCC or by using the ALTER TABLE statement with ADD PRIMARY KEY. In *Zen User's Guide*, see To set or remove a column as a primary key and SQL Editor.

You must create the primary key on a column or columns that do not allow duplicate values or null values.

If necessary, you can modify the column attributes and make the column the primary key at the same time. Here is an example using SQL:

```
ALTER TABLE mytable MODIFY id INTEGER UNIQUE NOT NULL PRIMARY KEY
```

If you want to add a primary key consisting of more than one column, you must add the key separately:

```
ALTER TABLE mytable ADD PRIMARY KEY(id, myname)
```

For more examples, see ALTER TABLE in *SQL Engine Reference*.

Setting up Foreign Keys

You can create foreign keys using SQL statements or Zen Control Center. When you create a foreign key, you may define an associated rule at the same time. You can define multiple rules on the same key. If you create a foreign key without specifying associated rules, the default referential integrity is restrict for both update and delete.

Creating a Foreign Key During Table Creation

You can create a foreign key when you create a table, by using the REFERENCES keyword in your column definition. A foreign key can consist of one or more columns. The data types of the column(s) must be the same as the primary key that this foreign key references. The example next shows the column named your_id being created then being designated the foreign key, referencing mytable.id:

```
CREATE TABLE yourtable (your_id INTEGER REFERENCES mytable(id) ON DELETE CASCADE, yourname CHAR(20))
```

You can also add the foreign key designation at the end of the statement. You must use this technique if you wish to use multiple columns in the key:

```
CREATE TABLE yourtable (your_id INTEGER,
    yourname CHAR(20),
    FOREIGN KEY(your_id, yourname) REFERENCES
    mytable(id, myname) ON DELETE CASCADE)
```

When you create a foreign key, the database engine adds an index on the designated columns.

For more examples, see CREATE TABLE in *SQL Engine Reference*.

Adding a Foreign Key to an Existing Table

You can add a foreign key to an existing table with ZenCC or by using the ALTER TABLE statement with ADD FOREIGN KEY. In *Zen User's Guide*, see Foreign Keys Tasks and SQL Editor.

In the following example, two rules are defined for this foreign key, both a delete rule and an update rule:

```
ALTER TABLE yourtable ADD FOREIGN KEY (your_id,yourname) REFERENCES mytable(id,myname) ON DELETE
CASCADE ON UPDATE RESTRICT
```

Use DELETE CASCADE with caution. See examples in Delete Cascade.

For more examples, see ALTER TABLE in *SQL Engine Reference*.

Interactions Between Btrieve and Relational Constraints

While Zen is designed to support concurrent access to the same data through both the Relational Engine and the MicroKernel Engine, some features of the relational (SQL) database architecture may interfere with Btrieve access to the data. For example, features that are designed to limit relational access, such as referential integrity (RI), may also limit Btrieve access in the interest of preserving data integrity.

You should fully understand integrity enforcement, bound databases, ODBC/SQL security, triggers, referential integrity and owner names before implementing these features on a database that is used by a transactional (Btrieve) application. In most cases you can get "best of both worlds" access to your database, but since security, referential integrity, and triggers can put constraints on access or operations, some Btrieve operations may be restricted or prevented depending on the implementation.

In some cases using integrity enforcement, bound databases, security, triggers, or referential integrity restrict Btrieve access to the data or file or prevent it completely when Btrieve access would violate restrictions placed on that data. Triggers and RI mainly limit the ability to manipulate data through the Btrieve API.

Security and owner names can limit access and/or the ability to manipulate that data without the proper account, rights, and password. There are many possible combinations of these features, so only the most common ones are listed here.

CONDITIONS						RESULTS	
DDFs Exist?	Integrity Enforced?	Bound Database?	Relational Security Used?	Triggers Used?	RI Used?	Btrieve Access allowed?	SQL/ODBC Access Allowed?
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)

CONDITIONS						RESULTS	
DDFs Exist?	Integrity Enforced?	Bound Database?	Relational Security Used?	Triggers Used?	RI Used?	Btrieve Access allowed?	SQL/ ODBC Access Allowed?
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) Regardless of the Bound Database setting for a database, the database engine automatically stamps a data file as bound if it has a trigger, a foreign key, or a primary key that is referenced by a foreign key. For more information on the meaning of a bound database or file, see Bound Databases.

(2) Adding triggers on a table will cause the trigger to execute when the corresponding Btrieve operation is performed. A trigger can validate data, do additional work, or cause the Btrieve operation to abort. See Bound Database versus Integrity Enforced for more information.

(3) When a database or file is secured, access is allowed as long as the user has permissions (that is, a relational user name and password or a valid Btrieve owner name) to that file. Files that are in a secure database but do not have Btrieve owner names set are accessible to Btrieve users. When relational security is first set up for a file that already has a Btrieve owner name, the Master user must grant relational permissions to users using the Btrieve file owner name. See Zen Security for more information.

(4) If a table contains referential integrity constraints, and Integrity Enforced is turned on for the given database, both Btrieve and SQL operations that would violate the constraints are disallowed. This mechanism preserves the integrity of the data regardless of the method of access.

Bound Database versus Integrity Enforced

If you do not specify the Integrity Enforced attribute for a named database, the database engine does not enforce any referential integrity, triggers, or security rules. If you specify the Integrity Enforced property for a named database, the MicroKernel enforces the defined security, RI, and triggers regardless of the method you use to access the data. The MicroKernel enforces these rules as follows:

- Btrieve users are not subject to relational security. If you have owner names on the files, they remain in effect. If you do not have owner names on the files, any Btrieve user can access the data regardless of relational security constraints. Btrieve operations are subject to all the RI and other constraints defined in the database as well as the Trigger restrictions listed below.

- If constraints exist on a given file, Btrieve access is permitted as follows:

Constraint on File	Level of Access Permitted Using Btrieve
RI constraints defined	User can access the data and perform any operations within RI constraints.
INSERT, UPDATE, or DELETE triggers defined	Corresponding Btrieve operations will cause triggers to execute.

The Integrity Enforced setting is not directly related to the Bound Database setting. A database can be bound without enforced integrity, or a database can have integrity enforced without being bound.

Bound Databases

If you specify the Bound attribute for a named database, the DDFs and data files in that database cannot be associated with any other database. Also, a bound data file cannot be associated with more than one table definition in the database. When you add new tables or DDFs to a bound database, the database engine automatically binds the new objects. This behavior prevents conflicts that could cause unpredictable behavior or data integrity corruption. For example, if you used the same data file for two different table definitions in the same database, you could define RI rules on one table but not on the other. In this case, inserting a row into the table without the RI rules would violate the RI rules on the other table. Binding the data files and DDFs prevents such conflicts.

DDFs and data files can be individually bound. The database engine automatically marks a data file as bound if it has a trigger, has a foreign key, or has a primary key that is referenced by a foreign key. These files cannot be shared with another database or associated with more than one table definition.

Whether a data file is bound has no direct affect on Btrieve access to that data file. However, files that are bound often have other constraints that may limit Btrieve access.

See Also

For information on how to manipulate the Integrity Enforced and Bound Database settings for a given database, see New Database GUI Reference.

Zen Security

Zen provides security both at the relational database level for SQL-based applications and at the file level for Btrieve applications that call the MicroKernel engine. Data access can be secured by authenticating users through the operating system, through an Active Directory account, or as a Zen database user. Once user identity is verified, user rights are authorized by operating system or Active Directory rights groups or by Zen database rights defined at the user or group level.

The following topics explain these security models and how to work with them, followed by a topic covering Zen data file encryption capabilities.

- Security Models for the Relational Engine
- Security Models for the MicroKernel Engine
- Planning Security for the MicroKernel Engine
- MicroKernel Engine Security Quick Start
- Data Encryption Over Networks

Security Models for the Relational Engine

Zen offers SQL applications the authentication and authorization configurations shown in the following table.

Database Security	Authentication	Authorization	Level of Security
Disabled	• Operating system • User name and password for server connection	• Unrestricted • Owner names (optional)	None. All tables are accessible through SQL connections.
Local database	• Zen users • User name and password, unrelated to operating system or domain	• Zen user rights (if no group) • Zen group rights (if no user rights) • PUBLIC rights • Owner names (optional)	• Databases • Tables • Table columns • Stored procedures (if V2 format) • Views (if V2 format)
Windows domain	• Active Directory users • User name and password • Users assigned to domain groups named after Zen database groups	• Zen group rights (required) • PUBLIC rights • Owner names (optional) • No individual user rights	Same as local database

By default, database security is disabled when you create a new database. You enable it by entering a password for its Master user, which is created when security is turned on. For information about allowed passwords, see Identifier Restrictions.

After enabling security for a database, you must create any users or groups needed to configure the permissions for various database activities. For local database security, you can assign permissions directly to users, or you can assign permissions at the group level and then assign users to a group. For Windows domain security, you create only groups, using the same names as rights groups in which network users are members. You can manage users and groups for each database using ZenCC or SQL scripting.

You can also configure permissions in the built-in PUBLIC group, which applies to all users connected to the database. Permissions for any user are a combination of either user or group permissions and those in the PUBLIC group. The PUBLIC group in a new database has no permissions set and does not need any settings if all permissions are managed at the group or individual user level.

The following topics explain these concepts in more detail:

- Master User
- The PUBLIC Special Group
- Users and Groups
- SQL and Zen Security

Master User

Enabling Zen database security creates a user named Master, who has full access to the database. The Master user requires a password, which you set when you enable security. Master passwords can differ among databases and are not connected. To make changes to the security configuration for a database, you must provide the Master password, so be sure to remember it.

After you authenticate yourself as Master for a database, you can use ZenCC to create users for that database, create groups to assign them to, and manage data access permissions at both the group and user level. You can also execute SQL statements to create groups and then users within those groups. A user can be a member of only one of the groups you create. All users automatically become members of the PUBLIC special group in each database.

The PUBLIC Special Group

If you want to grant the same permissions to all users, you can grant them to a special group named PUBLIC. The database engine automatically creates the special group PUBLIC when you turn on security. Initially, no permissions are assigned to PUBLIC. Until you assign PUBLIC permissions or create users and groups with their own permissions, access to Zen data is blocked.

PUBLIC is a special group that provides default permissions for *all* users. Even if you assign a user to another Zen group, that user remains a member of PUBLIC. A couple of examples help to clarify how PUBLIC permissions apply. Suppose in ZenCC that you assign the CREATE TABLE permission to PUBLIC. You then create a user named myuser whose permissions in ZenCC do *not* include individual rights to create a table. Myuser *can* create a table because the database engine first checks default permissions in PUBLIC, and PUBLIC, in this case, grants rights to create a table.

Conversely, if a permission is *not* granted to PUBLIC, then the permission granted to the individual user or group applies. For example, suppose in ZenCC that you do *not* assign the CREATE TABLE permission to PUBLIC. No user can create a table unless the permissions for the user, or the group to which the user belongs, allow creating a table.

Users and Groups

After you enable local database authentication, you can manage permissions for users and groups for the database. For Windows domain authentication, you can manage only group permissions, since user membership is assigned in Active Directory. Zen Explorer in ZenCC displays the users and groups you can work with in either case. The following table lists the general rules that apply to users and groups.

Rules for Users and Groups	Local Database	Windows Domain
A user is not required to be a member of a group and can have individual permission settings.	X	N/A
All users in a group have the permissions defined for that group.	X	X
When a user is a member of a group, that user has no individual permissions.	X	X
The permissions of a user unite with permissions of the PUBLIC group.	X	N/A
The permissions of a group unite with permissions of the PUBLIC group.	X	X
A user can be a member of only one group at a time, not counting the PUBLIC group.	X	X
A group cannot be a member of another group.	X	X

For more information, see User and Group Tasks in *Zen User's Guide*.

SQL and Zen Security

If you manage Relational Engine security through SQL scripting, the following topics in *SQL Engine Reference* provide useful information:

- Permissions on Views and Stored Procedures
- 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

Accessing Data in More Than One Database

A SQL application may access data from more than one database in a single connection if the databases are on the same machine. However, since you can be logged in to only one database at a time, your access to another database where you are not logged in depends on the security settings of both databases.

Security for Logged-in Database	Security for Second Database	Access to Second Database
Security enabled	None	Access granted with all rights.
Security enabled	Security enabled Database uses **same** user name and password as logged-in database.	Access granted for rights defined in second database.
Security enabled	Security enabled Database uses **different** user name and password as logged-in database.	Access denied.
None	Security enabled	Access denied.

Security Models for the MicroKernel Engine

Zen offers Btrieve applications the authentication and authorization configurations shown in the following table.

Btrieve Security	Authentication	Authorization
Classic	• Operating system • User name and password	• File system rights • Owner names (optional)
Database, secured by local database	• Zen users • User name and password, unrelated to operating system or domain	• Zen user rights • Zen group rights (optional) • PUBLIC rights (optional) • Owner names (optional)
Database, secured by Windows domain	• Active Directory users • User name and password • Users assigned to domain rights groups with same names as Zen database groups	• Zen group rights (required) • PUBLIC rights (optional) • Owner names (optional) • No individual user rights
Mixed	• Operating system or domain • User name and password	• Individual user rights, if using local database security • Group rights (required if using Windows domain security) • PUBLIC rights (optional) • Owner names (optional)

The following topics explain these configurations in more detail:

* Classic Btrieve Security

* Mixed Btrieve Security

* Database Security for Btrieve Files

* Notes on Classic and Mixed Btrieve Security

* Notes on Mixed and Database Btrieve Security

* Setting Up Mixed or Database Btrieve Security

* Owner Names

These topics apply only to applications that directly call the MicroKernel Engine. The terms *credentials*, *login credentials*, or *user credentials* refer to a valid user name and password.

Zen database authentication and authorization for the MicroKernel Engine can be configured to be dependent on or independent of the operating system. You can allow Btrieve users access to the database without allowing them operating system access to the data files.

Classic Btrieve Security

Classic security is the Btrieve security model provided in all releases of the database. For Btrieve users, authentication is performed by the operating system, and data access privileges are determined by file system rights for the given user. Users have the same kinds of rights to access a file through Btrieve that they have for other types of files controlled by the operating system.

Btrieve owner names can add another layer of security by restricting access to individual files. For more information, see Owner Names.

Setting Up Classic Btrieve Security

Under Classic security, you set up application users and access permissions simply by creating operating system users and assigning permissions to files and directories. You do not need to take other actions.

Mixed Btrieve Security

Mixed Btrieve Security provides access to Btrieve data files without giving the user operating system file permissions. You can use this option in environments where you are strengthening security and are unable to modify your Btrieve application to use API login or a URI to specify a file.

In the Mixed security model, when a Btrieve open request occurs, the database engine authenticates the operating system user against the users in a special database called DefaultDB. If the operating system finds the user, then the database engine uses the rights table for the database to determine the user's access to the file to be opened. Thus, permissions for users must be defined within the database they need to use. The database engine enforces its permissions regardless of operating system permissions.

Database authorization for Btrieve applications is provided by extending the Relational Engine security model so that it also can be used for Btrieve applications. The ability to create and define users and groups and set their permissions is provided in ZenCC and through SQL statements such as GRANT, REVOKE, ALTER GROUP, CREATE USER, ALTER USER, and DROP USER.

Under the Mixed security model, local database authentication and Windows domain authentication differ slightly. For local database authentication, any user names defined in the DefaultDB database must be the same as those defined in the operating system. During a Btrieve file create or open operation, the database engine passes the entered user name and password to the operating system for authentication. If the operating system authenticates the credentials, then the database uses the user's individual permissions or the permissions of his or her assigned group. Instead of defining permissions for each user or group, you can define them once using the PUBLIC group.

Under Mixed security with Windows domain authentication, group names in Active Directory must be the same as groups defined in the database, but user membership is handled only in Active Directory. During a Btrieve file create or open operation, the database engine passes the user name and password entered by the user to the network for authentication. If the network authenticates the credentials, then the database uses the user's group assignment to determine permissions. You do not need to add users to the database, and in fact, the Users node in ZenCC is no longer displayed. As an alternative to defining permissions for various groups, you can define them once using the PUBLIC group.

For Mixed security setup procedures, see Setting Up Mixed or Database Btrieve Security.

Notes on Classic and Mixed Btrieve Security

Since the Workgroup engine performs no operating system authentication, the behavior of the Classic and Mixed security policies using the Workgroup engine are the same. If you wish to secure a Btrieve database using the Workgroup engine, you must choose the Database security policy and set up any needed users and groups.

Database Security for Btrieve Files

Database security for Btrieve files requires applications to issue a Btrieve login operation or use a Btrieve URI to specify the file in an Open operation. In either case, a database is referenced in the login or URI. This database provides the permissions for the specified user. Under Btrieve database security, the Zen database authenticates and authorizes users of Btrieve data files either based on users defined locally in the Zen database or based on network logins defined in Windows Active Directory where users are members of rights groups with the same names as groups in the Zen database. With the one exception noted below, the ability of users to connect to and access data is unrelated to file system permissions as long as they can successfully log in to the system where the needed application runs. The ability to define database users and groups and set permissions is provided in ZenCC and through SQL statements such as GRANT and REVOKE.

Note: To create new databases, a user must have administrator privileges in the operating system.

Notes on Mixed and Database Btrieve Security

You can use the Mixed or Database security models only with Btrieve data files that reside in the directories defined to belong to a given database, including the default database DefaultDB described in The Default Database and the Current Database. Data files residing in directories that have not been associated with a database cannot be accessed.

One of the primary advantages of these security models is the ability to restrict direct user access to the data files while still allowing full access to the data through the database engine. In contrast, under the Classic model, any user permitted to add records to the data file must necessarily also have the ability to copy or delete the data file from the operating system.

Setting Up Mixed or Database Btrieve Security

Migrating to mixed or database security requires that you make a number of choices and plan carefully. In a well-established environment, you may have to plan how your Btrieve files will be grouped together into databases, and schedule the migration so that you do not disrupt your production environment.

For the procedures for moving from the default Classic to Mixed or Database security, see Security Tasks in *Zen User's Guide*.

Owner Names

An owner name is a string of bytes used to restrict access to a Btrieve data file. You can think of an owner name as a password that gives access to an individual file. In addition, if Zen encrypts a file, the owner name serves as the private encryption key. Owner names reside in the file header and are always encrypted, whether or not the file is encrypted. As with most passwords, owner names are case-sensitive.

All Zen database engines offer the ability to set a "short" owner name of 1–8 ASCII characters. In addition to restricting access, the short owner name is used to apply proprietary encryption when you select the Encrypt File setting in Function Executor or the Encrypt Data in File setting in the Maintenance tool.

Zen 10.10 introduced a "long" owner name of 1–24 ASCII characters, which increased file security by enabling the use of stronger, 128-bit encryption.

Zen 13.30 added AES-192 encryption and support for long owner names as hexadecimal strings.

Zen 14.00 lengthened long owner names to a maximum of 32 ASCII characters, extended the hexadecimal limit accordingly, and added AES-256 encryption.

The possibilities for long owner names and file encryption are summarized in the following table.

Zen Version	Long Owner Name	File Encryption
14.00	**13.0 format** 1–32 ASCII characters 32–64 hexadecimal digits, plus 0x or 0X prefix **9.5 format** 1–24 ASCII characters 32–48 hexadecimal digits, plus 0x or 0X prefix	Files in 13.0 format with long owner names of 25–32 ASCII characters or 50–64 hexadecimal digits use AES-256 encryption. Files in 13.0 format with long owner names of 1–24 ASCII characters or 32–48 hexadecimal digits use AES-192 encryption. Files in 9.5 format with long owner names use 128-bit encryption. **Note:** Zen v16 introduced file version 16.0. This format provides the same owner name and file encryption capabilities as 13.0.
13.30	**13.0 format** 1–24 ASCII characters 32–48 hexadecimal digits, plus 0x or 0X prefix **9.5 format** 1–24 ASCII characters 32–48 hexadecimal digits, plus 0x or 0X prefix	Files in 13.0 format with long owner names use AES-192 encryption. Files in 9.5 format with long owner names use 128-bit encryption.
10.10 to 13.20	**9.5 format** 1–24 ASCII characters	Files in 9.5 format with long owner names use 128-bit encryption.

Long owner names in hexadecimal must meet the following requirements:

- The owner name string is prefixed with the characters 0x or 0X.

- The owner name string uses only hexadecimal digits 0123456789abcdefABCDEF, with no spaces.

- No null characters, encoded as 00, are used.

- The number of digits for 13.0 and 16.0 format files must be an even total between 32 and 64, plus the 0x or 0X prefix. The number of digits for a 9.5 format file must be an even total

between 32 and 48, plus the 0x or 0X prefix. Note that the prefix serves only to indicate hexadecimal values and is not used by the encryption algorithm.

The following restrictions apply to files with long owner names:

- Due to stronger encrypting of owner name strings in Zen v14 or later, a 13.0 format file assigned a long owner name in Zen v14 can be read only in Zen v14 or later. This applies whether or not the file itself is encrypted. Likewise, only Zen v16 or later can read 16.0 format files. Files in 9.5 format do not have the stronger encrypting and are not affected.

- Zen before version 10.10 does not support long owner names and cannot read files that use them.

- A file can be rebuilt to a file format before 9.5 only after the owner name is removed.

Note: Owner names have no connection with user names authenticated at the network, system, or database level. For example, the file owner name `Master` is not the same as the default Master user.

The following topics provide more information:

- Choosing and Setting an Owner Name
- Owner Names and SQL Access
- Owner Names and Encryption
- Owner Name Examples

Choosing and Setting an Owner Name

You can set or clear an owner name on a file in the Maintenance and Function Executor tools. You cannot use SQL scripts to set or manage an owner name for a table, but SET OWNER and GRANT statements can be used to provide an owner name when it is required.

Depending on the options selected when an owner name is assigned, a user can access the file in the ways shown in the following table.

Option	Description
Read-only	Without specifying the owner name, users can perform data access operations that do not modify the data file.
Read-only encrypted	Without specifying the owner name, users can perform data access operations that do not modify the data file. When you set this option, the database engine encrypts every record in the file using the owner name as a key. Records added later are also encrypted.

Option	Description
Normal	Without specifying the owner name, users cannot perform any file access operations.
Normal encrypted	Without specifying the owner name, users cannot perform any file access operations. When you set this option, the database engine encrypts every record in the file using the owner name as a key. Records added later are also encrypted.

Owner Names and SQL Access

If you have a Btrieve owner name set on a file serving as a table in a SQL database, then you can access the table in two ways, depending on whether security is enabled for the database. With no security, the SET OWNER statement provides the owner name before any other attempt to access the table. With security, the Master user of the database must provide the owner name in a GRANT statement to issue permissions for the table to any user, including the Master user itself.

If a file with an owner name is read-only in the operating system, the Master user automatically has SELECT rights on this table without specifically granting itself SELECT rights with the owner name. In this case, other users do not have this automatic access, but the Master user can grant SELECT permission without providing the owner name.

Owner Names and Encryption

When you first set an owner name with encryption on a file, the database engine immediately begins to encrypt the entire file. The larger the file, the longer encryption takes.

Data access operations on an encrypted file are slower than for an unencrypted file. The database engine must decrypt each page as it reads it from disk, and then encrypt it again before writing it back to disk.

Caution! Do not forget or lose a file owner name. If the file itself is not encrypted, it is still readable through the Btrieve API, but you cannot write to it without the owner name. It is not possible to discover the owner name of a file by examining its contents because the owner name string is always encrypted in the header of the file to which it is assigned, whether or not the file data is encrypted.

Owner Name Examples

For examples of granting access to files with Btrieve owner names, see GRANT in *SQL Engine Reference*.

Planning Security for the MicroKernel Engine

In every new database, by default the Btrieve security setting is Classic, where the database engine authenticates through the operating system and authorization is based on directory and file permissions. Any user who can access a data file through the operating system has the same level of permission to data records in the file, unless Btrieve owner names restrict access.

The following topics describe the steps to set up the default database, authorized users, and other aspects of the Btrieve security policies:

- Available Options
- Choosing Your Policy
- Preparing to Set Up Security
- Process Overview

Available Options

There are three security options available to you. The following table describes the features of these options to help you choose the best one. Encryption is optional in every configuration.

Feature	Classic	Mixed	Database
Administrator must set up separate operating system and database user accounts for each user when local database security is used.		X	X
Database user accounts are derived directly from OS user accounts. Always the case when Windows domain database security is used.	X	X	
Data access rights are unrelated to file system. Administrator must assign data access privileges to each user or group in the database.		X	X
User data access rights derive from OS user file system rights.	X		
Supports automatic login dialog for entering database user name and password from any Windows application based on Zen.	X	X^1	X
Database accepts successful OS login as valid database user.	X	X	X^2
User must log in to database separately from logging in to computer.			X^3

[1] The login dialog may appear if the requester cannot establish an identity through the operating system.

[2] For database secured using Windows domain authentication.

[3] For database secured using local database authentication.

Under Btrieve **Database** security plus Relational local database security, database user accounts are completely unrelated to operating system user accounts.

In contrast, under Btrieve **Classic** security, a user who successfully logs into the computer has access to the database contents, at whatever level of file system rights that the user has been assigned to the file that contains the data.

Lastly, the Btrieve **Mixed** security policy has aspects of both of the other policies. Under this scheme, users log in using their operating system user names and passwords, but then the users access rights to the data are governed by user or group permissions set up in the secured database.

Choosing Your Policy

The following topics capture some of the major reasons you might choose one security policy over another.

Reasons to Choose Classic

- You are comfortable with users having file system access to data files. For example, any user with rights to delete records from the data file can also delete the entire file from your operating system.

- You want the minimum administrative hassle; you don't want to set up both OS user accounts for each user and at least one database account.

- You do not need to have a variety of data access rights that vary from each user's file system rights.

- You don't want your users to have a separate login for the database.

Reasons to Choose Mixed

- You have existing Btrieve applications that cannot be changed to use a Btrieve login or a Btrieve URI path for opening files.

- You don't want your users to have a separate login for the database.

- You want to prevent valid database users from having any rights to the data files on the operating system. For example, you can prevent users who have all rights in the database from having rights to delete data files from the operating system.

- You are using local database security and are willing to set up database user accounts that have the same user names as OS user accounts, and you are willing to assign permissions to each database user or to his or her group. If you choose, all of your users can have the same level of permissions by inheriting them from the special group PUBLIC.

- You are using Windows domain security and are willing to set up rights groups in Active Directory with the same names as Zen groups defined in the database, with permissions assigned to each Zen group. No permissions are assigned in Active Directory, only group membership for users.

Reasons to Choose Database

- You have Btrieve applications that use the Btrieve login operation or Btrieve URI path formats.

- You want to have a separate login for the database. That is, after logging into the operating system, users must log in again to the database. This behavior is useful when some authorized computer users are permitted access to the database and some are not.

- You want to prevent valid database users from having rights to the data files on the operating system. For example, you can prevent users who have all rights in the database from having rights to delete data files from the operating system. You can also achieve this goal using the **Mixed** security policy.

- You want database user accounts that use different names than the operating system accounts. For example, operating system user jsmith might be required to log in to the database as john.

- The users and their permissions stay with the database, not with the server or machine. This allows you to move a database from one machine to another without having to recreate the users and their permissions for the database.

Preparing to Set Up Security

Setting up security for the MicroKernel Engine is a simple process, but it affords enough flexibility that some preparation is necessary. This section describes the information you should know before you begin to set up Btrieve security.

How Do Your Btrieve Applications Access Data?

If your applications have been or can be changed to use Btrieve login operations or open files with URIs, then you can choose either Database or Mixed Btrieve security and in both cases the database to be secured is registered in ZenCC. If your applications do not use Btrieve logins or URIs, then you can manage access by choosing Mixed and configuring the DefaultDB database.

How Many Databases?

For Mixed or Database security, you must either assign all users the same level of permissions, or create a set of defined users for each database.

In some cases where your Btrieve data files encompass two or more completely unrelated bodies of data, you may want to set up two or more separate databases, each with its own set of authorized users. Generally speaking, however, you want to minimize the number of separate databases so that you do not have to create and maintain multiple sets of defined users. Often, a single database is sufficient. User permissions within the database will allow you to regulate each user's access to the database, so you do not need to create separate databases just to limit certain users' access.

If you determine that you need only one database, you may use the preexisting database, **DefaultDB**, as the database associated with your Btrieve files. You may also set up your own named database instead.

Where Are the Data Files?

You associate a Btrieve data file with a database by specifying the directory containing the data file as a **Data Directory** for the given named database. Thus, you need to know the directories containing all the data files that you want to associate with the database. If all the data files reside in a subdirectory of a specific directory, all you need to know is the top-level directory path name. You can even use "C:\" if you wish to include *all* data files on the hard drive.

What Are the User Names?

If you plan to use Mixed security with local database authentication, you must either assign all users the same permissions, or set up user accounts for the users whose rights differ. If you are going to set up individual users, you must have a list of the operating system user names that you want to make into database user names. The database user names must match their operating system counterparts exactly.

If you plan to use Mixed security with Windows domain authentication, you must either assign all users the same permissions in the PUBLIC group or set up different Zen groups for users whose rights differ. In both cases, you must have a list of the network group names that you want to duplicate as database group names. The database group names must match their network counterparts.

What Security Policy?

Before you set up security, you must know what policy you plan to use. The setup process varies somewhat for each policy. Considerations in choosing a policy are presented in Choosing Your Policy.

Process Overview

The following high-level steps outline how to set up security for a database. More detailed instructions are provided in MicroKernel Engine Security Quick Start.

1. Preparation. As described in Preparing to Set Up Security, gather the information you need and make the decisions necessary to get started. How many databases? Where are the Btrieve files located? What are the user names? What security policy will you use?

2. Select a database to use with your Btrieve files, and populate the database with the data directory specifying the location of your data files. This step is only necessary for Mixed or Database security.

   For details on this step, see To use an existing database, including the predefined DefaultDB, with your Zen files in *Zen User's Guide*.

3. Turn on security.

   For details on this step, see To turn on security using Zen Explorer in *Zen User's Guide*.

4. Create users or groups as needed and set their permissions using SQL statements or ZenCC. This step is necessary only for Mixed or Database security.

   For the fastest, easiest way to grant users access, see To assign permissions to all users using Zen Explorer in *Zen User's Guide*.

5. Set the Btrieve Security for your database to Mixed or Database.

   For details on this step, see To set or change the security policy for a database in *Zen User's Guide*.

6. Secure the data files in the operating system. For Mixed or Database security, users now can access the data without having any rights to access the data files in the operating system. Refer to your operating system documentation for information on securing access to files.

Summary of Tasks for MicroKernel Engine Security

The following table illustrates the basic level of effort required using the different security models. The tasks required to implement the security models, see Security Tasks in *Zen User's Guide*.

Security Model	Authentication/ Authorization	Summary of Behavior and High-Level Setup Tasks
Classic	Operating system/ Operating system	• Give users file permission access to all database files. • Add an owner name to Btrieve files to further limit access (optional)
Mixed	Operating system/ Database	• Set up users in the operating system. Users will be authenticated against these user names and passwords. • Set up like-named users in the database using the ZenCC. Although the OS authenticates, the database stores permissions, so OS users or domain groups must match users or groups in the database. • Define user or group permissions using ZenCC or SQL statements. Alternatively, define a set of rights for the group PUBLIC. Each authenticated user will have the same rights as PUBLIC. No user can have rights defined that are lower than that of PUBLIC. • For the Workgroup engine, this security model behaves like Classic.
Database	Database/ Database	• Operating system user names and passwords are unrelated to Zen database security. • Define users or groups using ZenCC or SQL statements. • Define database permissions using ZenCC or SQL statements.

MicroKernel Engine Security Quick Start

This section provides step-by-step instructions on the fastest, easiest way to secure your Btrieve data files in the operating system while still allowing database users to access the data.

When this procedure is complete, you can revoke operating system user rights to the data files without affecting database user rights to access the data through an application.

Note: You must be logged into the computer where the database engine is installed, as an operating system user with administrative rights or as a user who is a member of the Zen_Admin security group.

1. Start Zen Control Center (ZenCC). For how to start ZenCC, see Starting ZenCC on Windows in *Zen User's Guide*.

2. If you are using Mixed security, then the database to use in the following steps is DefaultDB. If you are using Database security, then make sure your database is registered in ZenCC. For how to register a database engine, see To register a remote server engine in *Zen User's Guide*.

3. Click the expand icon to the left of the node for the database you are using, whether DefaultDB or the database for your application.

4. In ZenCC, right-click the database DefaultDB, then click **Properties**.

5. Click **Directories** then click **New**.

6. Type a path for the Btrieve files then click **Apply**.

    If your files are spread over many directories, specify a high-level directory that they all have in common. You can specify a root level if necessary, but doing so includes in DefaultDB all Btrieve files at the root level and its subordinate directories. For example, a root level could be C:\ for Windows. See To use an existing database, including the predefined DefaultDB, with your Zen files in *Zen User's Guide*.

    You do not need to enter every directory, just the lowest level directory that is common to all Btrieve files you want to include in the database.

7. Enable security on the database by clicking the Security node in the Properties tree.

8. Select the Btrieve Security tab and choose Mixed or Database security.

9. Select the Database Security tab and choose local database or Windows domain authentication.

10. Enter a password to use for the Master user, twice as prompted.

Security is now turned on, but access is based on OS user rights by default, so your users currently have the same access that they had before. The next step addresses this situation.

Note that passwords are limited to a maximum of 8 bytes. You may use any displayable character in a password except for the semicolon (;) and the question mark (?).

11. Click **OK** to close the Properties dialog.

12. Expand the Groups for your database (click the expand icon to the left of the node), then right-click the group **PUBLIC**.

13. Click **Properties** then **Permissions** in the tree.

14. Click the Database tab.

15. Click the desired permissions.

    For example, if you want to grant read-only rights to all authenticated users, click **Select**. This option will give all users read-only rights to the data. To give all users update permission, click **Update**, and so forth.

    If you need to grant individual users varying rights, then you must create group accounts (if using domain authentication) or individual user accounts (if using local database authentication) using the GRANT statement in SQL or using ZenCC. For more information, see Security Tasks in *Zen User's Guide*.

16. Click **OK**.

17. Secure the data files in the operating system according to your operating system instructions. You can now deny operating system users from having any rights to the data files, without affecting their ability to access the data through the database engine.

Caution! Be sure to secure the data files in the operating system. If you do not perform this step, the users still can access the files through the operating system with the same level of permissions that they had prior to this procedure. You must revoke the users' operating system privileges to the data files if you want to prevent users from being able to delete or modify the files directly.

Data Encryption Over Networks

Zen supports the encrypting of network traffic between Zen and the applications that call it. This type of encryption is often called *wire encryption* because it protects the data when it is traveling on the network wire, or on any network infrastructure, including wireless. While the use of wire encryption is not required, it provides additional deterrence against unauthorized access to the data transmitted by your application.

Zen wire encryption is not tied to any particular security model. All Zen security configurations can be used with or without turning on wire encryption. The rest of this topic covers the following:

- Configuration Properties for Wire Encryption
- Wire Encryption Notes
- Setting Up Encryption
- Effects of Encryption
- Encryption of Files on Disk

Configuration Properties for Wire Encryption

Two configuration settings are associated with wire encryption. The settings must be configured for each client machine as well as for the server. For more information on these settings, see the following:

- Wire Encryption
- Wire Encryption Level

To access wire encryption settings

1. In ZenCC, do one of the following:

    - For a server, right-click the server name under the **Engines** node. You can click the plus signs to expand the nodes.

    - For a client, right-click **MicroKernel Router** under the **Local Client** node.

2. Click **Properties**.

3. Click **Access** in the tree.

Wire Encryption Notes

To perform data encryption before data passes over the network, Zen uses Blowfish, a well-known and time-tested public domain algorithm, implementing its 40-, 56-, and 128-bit keys. Encryption using a 40-bit key provides the least amount of protection for the data. Encryption using a 56- or a 128-bit key is progressively more difficult to compromise.

As with all security using encryption, the greater the deterrence, the slower the performance, since some amount of processor time is needed to perform encryption and decryption.

Backward Compatibility

Earlier versions of Zen that did not support wire encryption are unable to communicate with a client or server from a later release that provides encryption. Any client or server that does not support encryption will return an error if it attempts to connect to a client or server that is using encryption.

Setting Up Encryption

Before turning on the encryption settings in your environment, first think about your encryption needs. You can select from four possible schemes for your encryption environment, depending on your situation:

- No encryption
- All communications encrypted
- Encryption to/from specific clients
- Encryption to/from specific servers

No Encryption

First of all, consider whether your data has characteristics that would favor encryption. Is your data confidential or proprietary? Is it valuable in the hands of unauthorized users? Can it be used to harm your organization? If you answer no to these question and others like them, then your data may not need to be encrypted at all. Under these circumstances, there may be no reason to incur the performance trade-off that encryption requires. If you aren't sure, talk to a data security expert.

Assuming your data does need to be protected, you still may not need encryption. If your applications run solely on a LAN, and you are comfortable with the existing security of your network, encryption may not provide any additional benefit.

Encryption to/from Specific Clients

Now suppose that you have one major customer at a remote site that has a connection to your database. You may wish to use encryption only for the communications that go to/from that remote client. You can achieve this affect by setting **Wire Encryption** at the remote client to **Always** and setting the server values accessed by that remote client to **If Needed**. All your internal clients would be set to **Never**. Thus, the servers will only use encryption when communicating with the remote client that requires encryption.

Encryption to/from Specific Servers

Now, suppose the situation is reversed and your environment includes one or more remote servers that are accessed by network infrastructure that you do not trust 100%. In this case, you can set those server values to **Always**, and set the local client values to **If Needed**. The result is encrypted communications only to those remote servers that require it.

All Communications Encrypted

Finally, if your Zen applications often run over WAN, VPN, or other external networks that you do not trust 100%, then you may wish to encrypt 100% of your database communications. In this scenario, you would set **Wire Encryption** to **Always** at *all* clients and servers.

Choosing an Encryption Level

Once you have decided which clients and servers require encrypted communications, you must decide what level of deterrence is appropriate for your needs.

While Actian Corporation cannot offer advice regarding the encryption level that meets your specific needs, we can provide some guidelines to help inform your discussions with an appropriate data security expert. These guidelines do not represent a guarantee or warranty from Actian Corporation that no third party will be able to intercept and/or decode your encrypted data. As with any encryption scheme, there is no such thing as an "unbreakable" code, only varying levels of difficulty to compromise different types of encryption. The 128-bit encryption used by Zen would be considered very difficult to decode using techniques and equipment available to a highly sophisticated individual hacker.

Low (40-bit) Encryption

Consider using this level of encryption in cases where your data has limited ability to harm your organization or your customers if it falls into the wrong hands. Another reason to consider a Low level of encryption is if you wish simply to prevent a casual observer on your network from being able to read your data as it travels over the wires.

Medium (56-bit) Encryption

Consider using this level of encryption in situations where you believe you need somewhat more protection than against just a casual observer, but you do not believe you require the strongest level of security.

High (128-bit) Encryption

Consider using this level of encryption in situations where your data contains very sensitive information such as credit card numbers, social security numbers, financial account numbers, or other information protected by law. Especially consider this level of encryption if your database is associated with an entity on the network that is well-known to contain sensitive data, such as an Internet shopping website or an online securities brokerage website. Consider this level of encryption if your organization has previously suffered attempts to compromise its data security.

Effects of Encryption

Using encryption reduces client-server performance. With encryption turned on, each piece of data must be encoded at the source and decoded at the destination. This process requires additional CPU cycles when compared to the same operations performed without encryption. The level of encryption should not affect the performance. The performance drop in using encryption is roughly the same no matter which of the three encryption levels you choose.

Encryption of Files on Disk

Zen also offers encryption of a data file when it is written to disk. To use this feature, you must set an owner name for the file and choose the encryption option. For more information, see Owner Names.

Logging, Backup, and Restore

Zen provides several powerful features to ensure data integrity and to support online backups and disaster recovery.

- Transaction Logging and Durability
- Understanding Archival Logging and Continuous Operations
- Using Archival Logging
- Using Continuous Operations
- Data Backup with Backup Agent and VSS Writer

Transaction Logging and Durability

Zen offers two levels of data integrity assurance for database operations that involve transactions: Transaction Logging and Transaction Durability.

This section contains the following topics:

- Using These Features
- Feature Comparison
- Which Feature Should I Use?
- How Logging Works
- See Also

Using These Features

Both of these features can be turned on or off in the database engine using configuration within Zen Control Center, or programmatically using the Distributed Tuning Interface. See Transaction Durability and Transaction Logging.

The default value for Transaction Durability is **Off**, and the default value for Transaction Logging is **On**.

Feature Comparison

Both features offer multifile transaction atomicity, to ensure that the data files remain consistent as a set and that incomplete transactions are never written to any data files.

Atomicity means that, if any given data operation within a transaction cannot successfully complete, then none of the operations within the transaction are allowed to complete. An atomic change does not leave partial or ambiguous effects in the database. Changes to individual files are always atomic whether Transaction Logging and Transaction Durability are on or off. But transactions make it possible to group changes to multiple files into one atomic group. The atomicity of these multifile transactions are assured by the MicroKernel only when using transactions in your application, and Transaction Logging or Transaction Durability is turned on.

In addition to these benefits, Transaction Durability guarantees that, in the event of a system crash, the data files will contain the full results of any transaction that returned a successful completion status code to the application prior to the crash.

In the interest of higher performance, Transaction Logging does not offer this guarantee. Whereas Transaction Durability ensures that a completed transaction is fully written to the transaction log before the engine returns a successful status code, Transaction Logging returns a successful status code as soon as the logger thread has been signaled to flush the log buffer to disk.

Transaction Logging is a subset of Transaction Durability; that is, if Transaction Durability is turned on, then logging takes place and the Transaction Logging setting is ignored by the database engine.

The main differences between Transaction Logging and Transaction Durability are shown in the following tables:

Feature	Guaranteed data consistency and transaction atomicity across multiple files	Guaranteed commit for all completed transactions that have returned a successful status code
Transaction Logging	Yes	No
Transaction Durability	Yes	Yes

Feature	Timing of log buffer writes to disk
Transaction Logging	The log buffer is written to the log file when the log buffer is full or Initiation Time Limit is reached. A successful status code for each End Transaction operation is returned to the application as soon as the logger thread has been signaled to flush the buffer to disk.
Transaction Durability	The log buffer is written to the transaction log file with each End Transaction operation. A successful status code for each End Transaction operation is not returned to application until the log disk write is successful. For insert or update operations that are not part of a transaction, the log buffer is written to the log file when the log buffer is full or Initiation Time Limit is reached.

Which Feature Should I Use?

For the fastest performance, you want to use the lowest level of logging that meets your transaction safety needs. The best way to determine your appropriate level of logging is to ask your application vendor. If you have multiple applications that use Zen on the same computer, you must use the highest level of logging required by any of the applications.

If you only have one data file, or if none of your applications perform transactions involving multiple data files, you generally do not need to use Transaction Durability or Transaction

Logging. Under these circumstances, Zen guarantees the internal consistency of each data file, with or without logging.

Transaction Logging

Turn on Transaction Logging if at least one of your Zen applications performs transactions across multiple data files. Without Transaction Logging, Zen cannot guarantee multifile atomicity of transactions or multifile data integrity.

In the event of a system crash, this level of logging does not guarantee that every completed transaction has been written to the data files.

Transaction Durability

Turn on Transaction Durability if at least one of your Zen applications requires that completed transactions across multiple data files be absolutely guaranteed to have been written to the data files under almost any circumstances.

In the event of a system crash, this level of logging guarantees that every transaction that has been successfully completed has been written to the data files.

How Logging Works

Note that these features ensure atomicity of transactions, not of operations. If you are using SQL, a transaction is defined as a set of operations that take place between a BEGIN statement or START TRANSACTION statement, and an END or COMMIT statement. If you are using Btrieve, a transaction is defined as a set of operations that take place between a Start Transaction operation and an End Transaction operation.

All data file inserts and updates are stored in the log buffer. When a transaction is completed (Transaction Durability) or when the buffer gets full or the Initiation Time Limit is reached (Transaction Durability or Transaction Logging), the buffer is flushed to the transaction log file.

In the case of Transaction Logging, when the engine receives the operation ending the transaction and successfully signals the logger thread to flush the log buffer to disk, the engine returns a successful status code to the application that initiated the transaction. In the case of Transaction Durability, the engine does not return the successful status code until the logger thread signals that is has successfully written the buffer to disk.

Transaction log file segments are stored in the location specified in the setting **Transaction Log Directory**. The log segments are named *.LOG, where the prefix can be 00000001 through FFFFFFFF.

Note: All operations, regardless of whether they take place within a transaction, are written to the log file when Transaction Logging or Transaction Durability is in effect. However, only operations executed within a transaction are guaranteed to be atomic. In the case where a system crash has occurred and the transaction log is being rolled forward, only completed transactions are committed to the data files. All operations without an associated End Transaction operation are rejected, and are not committed to the data files.

Tip... If your database is highly used, consider configuring your system to maintain the transaction logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the writes to the log files and to the data file are split across different drives instead of competing for I/O bandwidth on a single drive. The overall disk I/O is not reduced, but the load is better distributed among the disk controllers.

You can specify the location of the transaction logs using the configuration setting **Transaction Log Directory**.

If a system failure occurs after the log file has been written but before the committed operations are flushed to the data files in a system transaction, the committed operations are not lost. In order to flush the committed operations the affected files need to be opened and operations performed after the system failure. When the files are opened and operations attempted, it is then that the data is rolled forward to the files affected at the time of system failure. Simply restarting the database engine will not invoke the roll forward operation nor will it make the data consistent.

Note: Log files associated with the rolled forward files will not be automatically deleted, as they may be associated with more than one data file.

This feature allows individual client transactions to receive a successful status code as soon as possible while at the same time taking advantage of performance gains offered by grouping multiple client transactions together and writing them to the data files sequentially.

If your database server suffers a disk crash of the volume where the data files are stored, and you have to restore the data from an archival log, the engine does not roll forward the transaction log file. The archival log contains all the operations in the transaction log, so there is no need to roll forward the transaction log.

Tip... After a system failure, open all data files and perform a stat or read operation on those files. Once you are certain that all data has been restored, old log files may then be stored in a safe location.

See Also

For further information, see:

- Transaction Durability
- Transaction Logging
- Transaction Log Directory

Understanding Archival Logging and Continuous Operations

The product offers two mutually exclusive features to support online backups and disaster recovery.

If your situation is like this...	... use this feature:
You must keep your database applications running while performing backups.	Continuous operations
You are able to shut down the database engine to perform backups	Archival logging

Archival Logging allows you to keep a log of database operations since your last backup. In case of a system failure, you can restore the data files from backup then roll forward the changes from the log file to return the system to the state it was in prior to the system failure.

Caution! Archival logging does not guarantee that all your data files will be in a consistent state after restoring from an archival log. In the interest of speed, the database engine does not wait for a successful status code from the logging function before emptying the log buffer. Thus, in rare circumstances such as a full disk or a write error in the operating system, updates that were successful in the data files may not be recorded in the archival log. In addition, archival logging does not require you to log all of your files, so a transaction that updates more than one file may not be completely recorded in the archival log if you are only archival logging some of those files. As a result, one file may not be consistent with another. If you use transactions and require multifile transaction atomicity, see Transaction Logging and Durability.

Continuous Operations allows you to backup database files while the database engine is running and users are connected. After starting Continuous Operations, the database engine closes the active data files and stores all changes in temporary data files (called *delta* files). While Continuous Operations are in effect, you perform a backup of the data files. The delta files record any changes made to the data files while the backup is taking place.

When the backup is complete, you turn off Continuous Operations. The database engine then reads the delta file and applies all the changes to the original data files. The temporary delta file may surpass the size of the original data file if users make extensive changes to the file during continuous operation.

A file put into continuous operations locks the data file from deletion through the Relational Engine and the MicroKernel Engine. In addition, the file is locked from any attempts to change the file structure, such as modifying keys and so forth.

Difference Between Archival Logging and Transaction Logging

Transaction Logging is another feature designed to protect the integrity of your data in the event of a system failure, but it is not directly related to Archival Logging. You can have Transaction Logging in effect at the same time as either Archival Logging or Continuous Operations. Transaction Logging uses a short-term log file to ensure that transactions are safely written to disk. The transaction log is reset frequently as completed client transactions are rolled into the physical data files by way of system transactions. In the event of a system failure, when the database engine starts up again, it reads the transaction log and flushes to the data files the transactions that were completed prior to the system failure.

The archival log is written to at the conclusion of each system transaction, so the archival log and the transaction log should remain properly synchronized unless a system failure occurs exactly during the system transaction.

For more information on Transaction Logging, see Transaction Logging and Durability.

What if a File Restore is Needed

In the event of a system crash that requires restoring data files from backup, Archival Logging allows you to restore from backup and then recover database activity up to the moment of the crash.

If you experience a similar crash without Archival Logging (for example if you use Continuous Operations to perform backups), then you will not be able to recover database activity that took place between the last backup and the system crash.

If Archival Logging is...	... this much data will be unrecoverable after a crash:
On	Unfinished transactions at the moment of failure.
Off	All database operations that have occurred after the last backup of the data files.

The remainder of this chapter describes the options and procedures associated with Archival Logging and Continuous Operations.

Using Archival Logging

This section explains the procedures you must follow to set up Archival Logging, make backups, and restore data files. It is divided into the following topics:

- General Procedures
- Setting up Archival Logging
- Roll Forward Command

General Procedures

For Archival Logging to work properly, you must follow a clearly defined setup procedure, and another procedure in the event that restoring from backup is needed.

Caution! If any steps of the procedures are omitted or compromised, you may not be able to restore your data from backup.

To use Archival Logging properly

1. Turn on Archival Logging, if it is not already in effect. See Setting up Archival Logging for the detailed setup procedure.

2. Shut down the database engine.

3. Backup the data files.

4. After a successful backup, delete all existing archival logs.

Caution! Delete the corresponding log files *before* you resume working with the data files. Synchronizing the backup data files and the corresponding log files is a critical factor of successful recovery.

5. Restart the database engine.

To restore data files from backup and apply changes from the archival logs

Note: You cannot use this procedure to roll forward the archival logs if you experienced a hard disk crash and your archival logs and data files were both located on the lost hard disk.

1. When the computer restarts after the system failure, ensure that the database engine is not running, and ensure no other database engine is accessing the data files you wish to restore.

2. Restore the data files from backup.

3. Start the database engine, ensuring that no applications of any kind are connected to the engine.

Caution! It is crucial that no database access occurs before the archival logs have been applied to the data files. Make sure no other database engine accesses the files. You must roll forward the archival logs using the same engine that encountered the system failure.

4. Issue the Roll Forward command as described in Roll Forward Command.

5. After the Roll Forward completes successfully, stop the database engine and make a new backup of the data files.

6. After you have successfully backed up the data files, delete the archival log files. You may now restart the database engine and allow applications to access the data files.

Setting up Archival Logging

Setting up Archival Logging requires two steps:

• Turning on the Archival Logging feature

• Specifying the files to archive and their respective log files

Note: To perform these procedures, you must have full administrative permissions on the machine where the database engine is running or be a member of the Zen_Admin group on the machine where the database engine is running.

To turn on Archival Logging

1. Access **Control Center** from the operating system **Start** menu or **Apps** screen.

2. In Zen Explorer, expand the **Engines** node in the tree (click the expand icon to the left of the node).

3. Right-click the database engine for which you want to specify archival logging.

4. Click Properties.

5. Click **Data Integrity** in the tree to display the settings for that category of options.

6. Click **Archival Logging Selected Files**.

7. Click **OK**.

A message informs you that the engines must be restarted for the setting to take effect.

8. Click **Yes** to restart the engine.

To specify files to archive

You specify the files for which you want the MicroKernel to perform Archival Logging by adding entries to an archival log configuration file you create on the volume that contains the files. To set up the configuration file, follow these steps:

1. Create the directory \blog in a real root directory of the physical drive that contains data files you want to log. (That is, do not use a mapped root directory.) If your files are on multiple volumes, create a \blog directory on each volume.

   For example, if you have data files located on C:\ and D:\, and both drives are physical drives located on the same computer as the database engine, then you would create two blog directories, as next:

   ```
   C:\blog\
   D:\blog\
   ```

Note: On Linux and Raspbian, the log directory must be named blog and be created in the directory specified by the ACTIANZEN_ROOT environment variable (by default /usr/local/actianzen).

2. In each \blog directory, create an empty blog.cfg file. You can use any text editor, such as Notepad, to create the blog.cfg file. On Linux and Raspbian, the file must be named blog.cfg (lowercase).

3. In each blog.cfg file, create entries for the data files on that drive for which you want to perform Archival Logging. Use the following format to create each entry:

   ```
   \path1\dataFile1[=\path2\logFile1]
   ```

path1	The path to the data file to be logged. The path cannot include a drive letter.
dataFile1	The name of the data file to be logged.
path2	The path to the log file. Because the log file and the data file can be on different drives, the path can include a drive letter.
logFile1	The name of the log file. If you do not specify a name, the default value is the same directory and file name prefix as the data file, but replace the file name suffix with ".log." You may specify a different physical drive, so that the log and the data files are not on the same drive. Each data file being logged requires a different log file.

A single entry cannot contain spaces and must fit completely on one line. Each line can contain up to 256 characters. If you have room, you can place multiple entries on the same line. Entries must be separated by white space.

Caution! You must use a different log file for every data file that you wish to log. If you use the same log file for more than one data file, the MicroKernel cannot use that log file in the event that a roll-forward is needed.

If you do not provide a name for a log file, the MicroKernel assigns the original file name plus a .log extension to the log file when you first open it. For example, for the file b.btr, the MicroKernel assigns the name b.log to the log file.

Caution! You are not required to log every file in your database. However, if your database has referential integrity (RI) rules defined, you must log all or none of the files involved in each RI relationship. If you log only a subset of the files involved in a given RI relationship, rolling the archival logs forward after a system crash may result in violations of your RI rules.

Examples

The following examples show three sample entries in the blog.cfg file on drive C. All three entries produce the same result: Activity in the file C:\data\b.bti is logged to the file C:\data\b.log.

```
\data\b.bti
\data\b.bti=\data\b.log
\data\b.bti=c:\data\b.log
```

The next example directs the engine to log activity in the file C:\data\b.bti to the log file D:\data\b.lgf. This example shows that archival log files do not have to reside on the same drive as the data file and do not require the .log extension. However, the .log extension is provided by default if no other extension is used.

```
\data\b.bti=d:\data\b.lgf
```

Tip... Writing the log to a different physical drive on the same computer is recommended. If you experience a hard disk crash, having the log files on a different physical disk protects you from losing your log files and your data files at the same time.

The next example shows a blog.cfg file that makes the MicroKernel log multiple data files to a different drive (drive D), assuming this blog.cfg file is on drive C.

```
\data\file1.mkd=d:\backup\
\data\file2.mkd=d:\backup\file2.log
\data\file3.mkd=d:\backup\file3.log
```

Roll Forward Command

The Btrieve Maintenance tool (GUI or **butil** command line) provides a command allowing you to roll forward archival log files into the data files. See Performing Archival Logging.

Using Continuous Operations

Continuous operations provides the ability to back up data files while database applications are running and users are connected. However, in the event of a hard drive failure, if you use continuous operations to make backups, you will lose all changes to your data since the last backup. You cannot use Archival Logging and the Maintenance tool Roll Forward command to restore changes to your data files that occurred after the last backup.

Zen provides backup capabilities in the **butil** command for continuous operations.

Note: A file put into continuous operations locks its data against deletion through the relational and transactional engines. In addition, the file is locked against any attempt to change its structure, such as modifying keys. Actian Corporation also provides the companion product Backup Agent to set and manage continuous operations.

This section is divided into the following topics:

- Starting and Ending Continuous Operations
- Backing Up a Database with Butil
- Restoring Data Files when Using Continuous Operations

Starting and Ending Continuous Operations

This topic covers details on the commands **Startbu** and **Endbu**.

Command	Description
Startbu	Starts continuous operation on files defined for backup (BUTIL).
Endbu	Ends continuous operation on data files defined for backup. (BUTIL).

Caution! The temporary delta files created by continuous operation mode have the same name as the corresponding data files but use the extension ".^^^" instead. No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not use a naming scheme such as invoice.hdr and invoice.det for your data files. If you do, the MicroKernel returns a status code and no files are put into continuous operations.

Continuous operation mode does not significantly affect MicroKernel performance. However, using a server to back up files can affect performance.

To protect against data loss using continuous operation

1. Use the **startbu** command to put your files in continuous operation. See for an explanation of the command syntax with **butil**.

2. Back up your data files.

3. Use the **endbu** command to take your files out of continuous operation. See for an explanation of the command syntax with **butil**.

Backing Up a Database with Butil

This topic provides detailed information on backing up a database using the **butil** commands **Startbu** and **Endbu**.

Startbu

The **startbu** command places a file or set of files into continuous operation for backup purposes.

Format

```
butil -startbu <sourceFile | @listFile> [/UID<name> </PWD<word>> [/DB<name>]]
```

`sourceFile`	The fully qualified name of the data file, including the drive specification for Windows platforms, on which to begin continuous operation for backup. This fully qualified name must reside on the same machine as the one from which you are running **butil**. You cannot used mapped drives with **startbu**.
`listFile`	The name of a text file containing the fully qualified names of files on which to begin continuous operation. Separate these file names with a carriage return/line feed. The file names may contain blank characters. If the Maintenance tool cannot put all of the specified files into continuous operation, then it does not put any of the files into continuous operation.
`/UID<name>` `/UIDuname`	The name of the user authorized to access a database with security enabled.
`/UID<word>` `/UIDpword`	The password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is used.
`/UID<name>` `/UIDdbname`	The name of the database on which security is enabled. If omitted, the default database is assumed.

> **Note:** The startbu command begins continuous operation only on the files you specify. You cannot use wildcard characters with the **startbu** command.
>
> On Linux and Raspbian, all slash (/) parameters use the hyphen (-) instead of the slash. For example, the **/DB** parameter is **-DB**.

File Considerations

When selecting files for backup, we recommend that the temporary delta files created by Continuous Operations mode be excluded since they are open and in use during backup. If the delta files are included in the backup, they should be deleted before the database engine is started after the restore.

Examples for Windows Server

Example A The first example starts continuous operation on the course.mkd file.

For Windows Server:

```
butil -startbu file_path\Zen\Demodata\course.mkd
```

For default locations of Zen files, see Where are the files installed? in *Getting Started with Zen*.

Example B The following example starts continuous operation on all files listed in the startlst.fil file.

```
butil -startbu @startlst.fil
```

The startlst.fil file might consist of the following entries:

```
file_path\Zen\Demodata\course.mkd
file_path\Zen\Demodata\tuition.mkd
file_path\Zen\Demodata\dept.mkd
```

Endbu

The **endbu** command ends continuous operation on a data file or set of data files previously defined for backup. Issue this command after using the **startbu** command to begin continuous operation and after performing your backup.

Format

```
butil -endbu </A | sourceFile | @listFile> [/UID<name> </PWD<word>> [/DB<name>]]
```

/A	If you specify **/A**, the tool stops continuous operation on all data files initialized by **startbu** and currently running in continuous operation mode.

sourceFile	The fully qualified name of the data file (including the drive specification for Windows platforms) for which to end continuous operation.
	This fully qualified name must reside on the same machine as the one from which you are running butil. You cannot used mapped drives with the endbu command.
@listFile	The name of a text file containing a list of data files for which to end continuous operation. The text file must contain the fully qualified file name for each data file. Separate these file names with a carriage return/line feed. The file names may contain blank characters.
	Typically, this list of data files is the same as the list used with the **Startbu** command.
/UID*\<name\>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*\<word\>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*\<name\>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Note: On Linux and Raspbian, all slash (/) parameters use the hyphen (-) instead of the slash. For example, the /**A** parameter for is **-A**, as in `butil -endbu -A`.

Example for Windows Server

The following example ends continuous operation on the course.mkd file.

```
butil -endbu file_path\Zen\Demodata\course.mkd
```

However, you can also simply enter `butil -endbu course.mkd` instead of the full path if your current directory is F:\demodata.

Restoring Data Files when Using Continuous Operations

If you are using Continuous Operations for your backup strategy, then you have no recovery log that can be used to recover changes since your last backup. All database changes since your last backup are lost, with the possible exception of any transactions stored in the transaction log. Any such transactions are automatically rolled forward by the database engine when it starts up.

To restore data and normal database operations

1. Resolve the failure.

   Perform the maintenance required to make the failed computer operational again.

2. Restore the data files from backup, or restore the hard drive image from backup, as appropriate.

3. Reinstall Zen if it was not restored as part of a disk image.

Caution! If the delta files were included in the backup, they should be deleted before the database engine is started in the next step.

4. Restart the database engine.

   Any database operations performed since the last backup must be performed over again.

Data Backup with Backup Agent and VSS Writer

In addition to the topics previously discussed, both Zen Enterprise Server and Cloud Server also provide the following solutions for data backup:

- Backup Agent
- Zen VSS Writer

If your backup software is *not* aware of the Microsoft Volume Shadow Copy Service (VSS), you can use Backup Agent with your backup software. If your backup software *is* VSS aware, Zen VSS Writer is automatically invoked during VSS backups. You do not need to use Backup Agent if your backup software is already VSS aware.

Backup Agent and Zen VSS Writer can be used together, but there is no advantage in doing so. Your backup process will be more streamlined if you select one method or the other.

Backup Agent

Backup Agent is an optional product. By default, it is not installed. You must install it after you install Zen Enterprise Server or Cloud Server.

Backup Agent provides a quick and simple method for you to set and manage Continuous Operations on your Zen database files. Setting and managing Continuous Operations is a critical piece when backing up your Zen databases without using Microsoft Volume Shadow Copy Service. Backup Agent handles setting and managing Continuous Operations on your open files so that your data is still available from your application during your backup. Once the backup procedure is complete, stopping Backup Agent automatically takes the files out of Continuous Operations and rolls in all the changes captured during the backup.

Backup Agent is compatible with many popular backup applications on the market. Note that the backup application must be able to issue commands to start and stop other applications (so that the commands can start and stop Backup Agent).

Zen VSS Writer

The Microsoft Volume Shadow Copy Service (VSS) consists of Writer, Provider, and Requestor components. Zen supports VSS with only a Writer component, Zen VSS Writer.

Zen VSS Writer is a feature of the database engine and is enabled for Zen Enterprise Server and Cloud Server. Zen VSS Writer is available for use after that product is installed. Zen VSS Writer is currently not available for use with Zen Workgroup.

Zen VSS Writer is available only on Windows operating systems. For more information on Volume Shadow Copy Service, see the Microsoft document, [A Guide for SQL Server Backup Application Vendors](#).

Overview

During VSS snapshots, Zen VSS Writer blocks all disk I/O *write* activity to all Zen data and transaction log files, regardless of the volume on which they reside. After the snapshot is taken, Zen VSS Writer allows all disk I/O to resume, including any writes deferred during the snapshot.

Zen VSS Writer never blocks disk I/O *read* activity, allowing normal database processing to continue as long as writes are not required. Zen VSS Writer operates normally during the backup phase, although performance may likely be reduced due to the backup activity of the VSS service and VSS Requestor.

The Microsoft Volume Shadow Copy facility allows Backup and Restore products to create a shadow copy for backup in which the files are in either one of the following states:

• A well-defined and consistent state

• A crash-consistent state (possibly not suitable for a clean restore).

Files in the VSS snapshot will be in the well-defined and consistent state if all of the following are true:

• The file's writer is VSS-aware.

• The Backup and Restore product recognizes and notifies the VSS-aware writer to prepare for a snapshot.

• The VSS-aware writer successfully prepares for the snapshot.

Otherwise the writer's files are backed up in the crash-consistent state.

VSS Writer Details

The following items discuss specifics about Zen VSS Writer.

• Supported Operating Systems

    The same Windows operating systems that support the Zen server products also support Zen VSS Writer. Zen VSS Writer is functional on the same bitness as the machine's operating system and the installed Zen server product. Zen VSS Writer 32-bit is supported only on 32-bit machines, and 64-bit is supported only on 64-bit machines. If the bitness does not match, Zen functions properly, but VSS Writer is unavailable.

• Supported Backup Types

Zen VSS Writer supports manual or automatic backups of data volumes. Zen VSS Writer is supported on Full and Copy Volume backups. Incremental, Differential, and Log backups are not supported. VSS recognizes Zen VSS Writer as a component. However, Zen VSS Writer does not support component backups. If the VSS Requestor does call Zen VSS Writer in a component backup, the VSS Writer performs the same actions as in a Full or Copy Volume backup.

• Virtualized Environment Support

Zen VSS Writer supports VSS Requesters that trigger VSS backups in virtualized environments. Performing a VM snapshot does not invoke a VSS backup.

• Multiple Volume Zen Data Files

Zen files and transaction logs can reside on multiple volumes. When backing up Zen files, remember to backup the transaction logs and related files on other volumes simultaneously. Files that are independent of one another may not need to be backed up at the same time as related Zen files.

• Backup Solution Compatibility

To determine if a particular backup product recognizes Zen VSS Writer and will notify the Writer to prepare for a snapshot, start a backup with the product. After the backup is in progress, consult the zen.log file to determine whether the Zen VSS Writer logged the Frozen or Thawed states. If the backup and restore product did not notify Zen VSS Writer to prepare for the backup, another solution must be used. For example, you could use Backup Agent to back up Zen data files in the well-defined and consistent state.

• Zen VSS Writer and Restore Operations

Stop the Zen engine service before performing a Restore operation with the Backup software. Failure to do so causes the VSS Writer to inform the VSS Requester that it cannot participate in the Restore. Transaction logs will need to be restored along with the data files to guarantee the integrity of the data. If Zen data and transaction log files are restored while Zen is running, the results are unpredictable and could lead to data corruption.

• Zen VSS Writer and Zen Continuous Operations

You may have an existing backup process that already uses Zen Continuous Operations or Backup Agent. If you choose, you can continue to use that process with Zen and Zen VSS Writer. Zen VSS Writer does not interfere with Continuous Operations or Backup Agent. However, there is no advantage to using *both* Zen VSS Writer and Continuous Operations (or Backup Agent) together. Your backup process will be more streamlined if you select one method or the other.

When Zen VSS Writer is called and files are in Continuous Operations, be aware that VSS Writer operates independently from any Continuous Operations. If files *are* in Continuous

Operations when a VSS backup is in progress, view zen.log after the backup completes. Ensure that the Frozen and Thawed states completed successfully and that the data is in a well-defined and consistent state.

Also note that Zen VSS Writer requires the Microsoft VSS framework. Backup Agent does not use the Microsoft VSS framework. Consequently, Backup Agent does not participate in the backup when the VSS framework calls Zen VSS Writer and I/O operations are quiesced. Backup Agent must be added separately to the backup process. The backup process must also start and stop Backup Agent.

High Availability Support

This section includes the following topics:

- Overview of Technologies
- Failover Clustering
- Migration
- Fault Tolerance
- Disaster Recovery

Overview of Technologies

Zen is compatible with numerous solutions that maximize uptime in physical and virtual environments. Such solutions continually evolve but can be classified generally as high availability, fault tolerance, and disaster recovery.

High Availability

The definition of *high availability* can differ depending on the software vendor that provides high availability solutions. In general, it refers to a systems design approach for a predictable baseline level of uptime, despite hardware failure, software failure, or required maintenance.

A common approach to ensure high availability in a physical environment is failover clustering. A common approach in a virtual machine (VM) environment is migration.

Failover Clustering

Zen is designed to function as a resource in a failover cluster environment in which only one server node at a time accesses the shared storage subsystem. If the primary node fails, a failover (or switch) to a secondary node occurs. Failover clustering allows a system to remain available while you perform software upgrades or hardware maintenance.

Zen is compatible with Microsoft Failover Cluster Services and with Linux Heartbeat. Refer to the documentation from those vendors for the specific manner in which they define and implement failover clustering. Zen Enterprise Server and Cloud Server is the recommended editions for failover clustering.

See Failover Clustering.

Migration

In general terms, migration allows a running VM or application to be moved between different physical machines without disconnecting the client or application. The memory, storage, and network connectivity of the VM are typically migrated to the destination.

Zen is compatible with the migration capability offered by Microsoft Hyper-V, VMware vSphere, and Citrix XenServer. As long as host names remain the same after the VMs are moved, Zen continues to operate normally. See the documentation from those vendors for the specific manner in which they define and implement migration.

See Migration.

Fault Tolerance

While high availability aims for a predictable baseline level of uptime, fault tolerance is the uninterrupted operation of a system even after the failure of a component. Fault tolerance requires synchronized shared storage. In virtualized environments, the VM that fails must be on a different physical host from the VM that replaces it.

Fault tolerance can be achieved using just physical machines. However, virtual environments lend themselves so readily to maintaining virtual servers in lockstep with each other that exclusively physical environments are increasingly less common. Zen servers are compatible with fault tolerance capabilities in an exclusively physical environment.

For virtual environments, Zen is compatible with the fault tolerance capability offered by VMware vSphere and Citrix XenServer. Refer to the documentation from those vendors for the specific manner in which they define and implement fault tolerance.

See Fault Tolerance.

Disaster Recovery

Disaster recovery involves duplicating computer operations after a catastrophe occurs and typically includes routine off-site data backup as well as a procedure for activating vital information systems in a new location.

Zen is compatible with major hypervisors that support disaster recovery technology that initializes backup physical or virtual machines. As long as all host names remain the same after the VMs are moved, Zen continues to operate normally. This allows rapid server replacement and recovery time.

Refer to the documentation from the hypervisor vendors for the specific manner in which they define and implement disaster recovery.

See Disaster Recovery.

Hardware Requirements

For all of the technologies mentioned in this section, we recommend that you select servers, disk subsystems, and network components from the hardware compatibility list provided by the vendor. We follow this same practice when testing for compatibility with vendor products.

Failover Clustering

Failover clustering provides a group of independent nodes, each with a server, all of which have access to a shared system of shared data files or volumes. The failover services ensure that only one server at a time controls data files or storage volumes. If the currently active server fails, then control is passed automatically to the next node in the cluster, and the server on that node takes over.

In Zen, only one database engine can service data. In a cluster, because a Zen engine is installed on every node, you must configure these engines so that they appear as a single engine, and not the multiple engines that they actually are.

Each Zen engine must be licensed separately for each cluster node while the node is Active. This is for both physical and virtual machines. The license type used for each node is Enterprise Server - Active-Passive. A standard Enterprise license cannot be authorized on both nodes.

This section covers the following topics:

- Microsoft Failover Cluster for Windows Server
- Linux Heartbeat
- Managing Zen in a Cluster Environment

Microsoft Failover Cluster for Windows Server

This topic covers adding a Zen engine service to a Microsoft failover cluster and assumes the following:

- You know how to install and configure a failover cluster and need only the information required to add and manage the Zen engine service.
- You are familiar with using Zen and its utilities such as Zen Control Center (ZenCC).
- You can set up DSNs using Zen ODBC Administrator.

General Steps

Here are the steps for creating a Windows failover cluster with Zen engines.

1. Set up and configure the nodes of the failover cluster.

   You must set up the Windows servers in the cluster and confirm that failover succeeds before you add a Zen engine. For example, verify that a failover finishes and that all resources remain available, then fail over back to the original node and verify again. See the Microsoft

documentation for setting up and verifying failover clustering using the Server Manager dashboard and Failover Cluster Manager.

2. Set up a shared file server on a separate, dedicated system.

    After you install the Zen engines, you will set them to use this common data location.

3. Install a Zen engine on each cluster node.

    See the steps in the following table. Install all of the Zen engines before moving to the next step.

4. Return to Failover Cluster Manager to configure them. Configure the Zen installations in the cluster to appear as a single database server.

    Continue with the steps in the following table.

Installing and Configuring Zen in a Windows Failover Cluster

The following table gives the recommended steps to add Zen to a failover cluster in Windows Server.

Action	Notes
Install Zen on cluster nodes.	Install a Zen server on each node and choose identical settings for each installation.
	Do **not** install Zen on the dedicated shared storage system where the Zen data files reside.
	By default the Zen engine service starts automatically with Windows. After installing on each node, in the Actian Zen Enterprise Server service properties, change the startup type to **manual** and stop the service. Failover Cluster Manager will start it when needed.
Add a role and select the Zen server as a generic service. Select the storage to be used for the application data files. Add a resource and select the Zen server as a generic service.	Do these steps in Failover Cluster Manager.
	Registry replication is not supported when running Zen engine as a service, so skip that option. You must manually configure the database properties for each Zen engine on each node, as explained later in this table.
	Set the file share as a dependency for the Zen engine service. Select the option **Use network name** for computer name.

Action	Notes
Confirm that shared storage has the necessary files and directories.	The Zen engine service typically runs under the LocalSystem account. Confirm that the LocalSystem account has permissions to read and write to the shared location.
	On the active node where you installed Zen, copy the dbnames.cfg file from the ProgramData directory to your chosen directory in shared storage.
	On the active node, copy the following directories from ProgramData to the same directory in shared storage. Optionally, you can copy them to the same directory as dbnames.cfg.
	• defaultdb
	• Demodata
	• tempdb
	• Transaction Logs
Configure database engine properties with ZenCC.	In ZenCC, configure the engine on the current active node in your cluster, then do the same settings for each of the other nodes.
	Set these engine properties for **Directories**. When prompted to restart the engine, click **No**.
	• For **Transaction Log Directory**, enter the location on the shared disk where you copied the Transaction Logs directory.
	• For **DBNames Configuration Location**, enter the location on the shared disk where you copied the dbnames.cfg file.
	In Failover Cluster Manager, take Zen resources offline and back online to apply changes.
	In ZenCC, under the Databases node for your server, set these properties:
	• In the **DEFAULTDB** properties for Directories, set **Dictionary Location** and **Data Directories** to the location on the shared disk where you copied the Defaultdb directory.
	• In the **DEMODATA** properties for Directories, set **Dictionary Location** and **Data Directories** to the location on the shared disk where you copied the Demodata directory.
	• In the **TEMPDB** properties for Directories, set **Dictionary Location** and **Data Directories** to the location on the shared disk where you copied the Tempdb directory.

The failover cluster is now configured. Remember that when failover occurs, client connections may also fail and require restarting of client applications.

Note: To apply a patch to Zen servers in a cluster environment, see this knowledge base article.

Linux Heartbeat

The Heartbeat program is one of the core components of the Linux-HA (High-Availability Linux) project. Heartbeat runs on all Linux platforms and performs death-of-node detection, communications and cluster management in one process.

This topic discusses adding the Zen engine service to Linux Heartbeat and assumes the following:

- You know how to install and configure the Heartbeat program and need only the information required to add Zen to a Cluster Service group.

- You are familiar with using Zen and its primary utilities such as ZenCC.

Preliminary Requirements

It is essential that Linux Heartbeat be functioning correctly before you add Zen to the cluster. See the documentation from the High-Availability Linux Project (www.linux-ha.org) for how to install Heartbeat, verify that it is working correctly, and perform tasks with it.

Just as you would for any application, set up the essential clustering components before you add Zen.

Installing and Configuring Zen in a Linux Heartbeat Failover Cluster

The following table gives the recommended steps to add Zen to Linux Heartbeat.

Action	Discussion
Install Zen on the Cluster Nodes	Install a Zen server on each cluster node and choose identical options for each installations. Do **not** install Zen on the cluster shared storage where the Zen data files reside.
	After installation, the database engine is set to start automatically when the operating system starts. With clustering, however, Linux Heartbeat controls starting and stopping the database engine. The controlling node in the cluster starts the engine, the other nodes do not.
	After you install a Zen server, ensure that the Group IDs for zen-data and zen-adm and the UID for zen-svc match on all nodes. If required, change the IDs to ensure they are the same.

Action	Discussion
Configure the Shared Storage	The shared storage is where the Zen data files reside. Shared storage for Heartbeat can be implemented many different ways. The multitude of possible implementations is beyond the scope of this document. This documentation assumes the use of an NFS mount.
	Create (or at least identify) a location on shared storage where you want the database to reside. The location is your choice. Confirm that user **zen-svc** has read, write, and execute authority for the location.
	Create two groups and a user on the shared storage so that each cluster node can access the database files.
	• Groups **zen-data** and **zen-adm** must match **zen-data** Group ID and **zen-adm** Group ID, respectively, on the cluster nodes.
	• User **zen-svc** must match **zen-svc** UID on the cluster nodes.
Create the Directory for the Shared Storage Mount	On each cluster node, log in as user **zen-svc** then create a directory that will be mounted to the shared storage. User **zen-svc** has no password and can only be accessed through the root account with the **su** command. The name of the directory is your choice.
Configure Heartbeat Server	Configure the Heartbeat server *on each of the nodes* that will control the Zen database engine. Configure the following:
	• Nodes. Add all nodes that you want in the cluster.
	• Authentication. Specify the type of authentication to use for the network communication between the nodes.
	• Media. Specify the method Heartbeat uses for internal communication between nodes.
	• Startup. Specify the setting for when the Heartbeat Server starts. Set this to **on**, which means that the "server starts now and when booting."
Assign Password for Heartbeat User	Linux Heartbeat provides a default user named hacluster for logging in to the Heartbeat Management Client. Assign a password to user hacluster *on each of the nodes* from which you want to run Heartbeat Management Client.
Add a Resource Group for Zen	Log in as root and start the Heartbeat Management Client on one of the cluster nodes. Log in as user hacluster and add a new group. For **ID**, specify a name for the Zen group. Set **Ordered** and **Collocated** to true.

Action	Discussion
Add the Resources to the Group	Add three resources to the Zen group: • IPaddr • Filesystem • Zen (OCF resource agent) IPaddr In the Heartbeat Management Client, add a new native item. For **Belong to group**, select the group you added for Zen. For **Type**, select IPaddr. On the resource you just added, specify the IP address of the cluster for the **IP Value**. Use the IP address assigned to the *cluster* (not the node) when Linux Heartbeat was installed and configured. Filesystem Add another new native item. For **Belong to group**, select the group you added for Zen. For **Type**, select Filesystem and delete the parameter fstype, which is not required. Add a new parameter and select "device" for **Name**. For **Value**, specify the device name of the shared storage, a colon, and the share mount location. Add another new parameter and select "directory" for **Name.** For **Value**, specify the directory to use with the NFS mount. Zen (OCF resource agent) Add another new native item. For **Belong to group**, select the group you added for Zen. For **Type**, click zen-svc with a **Description** of Zen OCF Resource Agent. No additional settings are required.
Create the Subdirectories on the Mounted Shared Storage	Now that you have added the Filesystem resource, the mount exists between the cluster server and the shared storage. On one of the cluster nodes, log in as user **zen-svc**. Under the shared storage mount, create a directory named "log" and another named "etc." For example, if the mount directory is /usr/local/actianzen/shared, you would add directories /usr/local/actianzen/shared/log and /usr/local/actianzen/shared/etc.

Action	Discussion
Configure the Cluster Server in ZenCC	On each of the cluster nodes, you need to configure the cluster server with ZenCC.
	Place all cluster nodes into standby mode except for the one from which you will run ZenCC. As user **zen-svc**, start ZenCC on the one active node or from a client that can access the active node.
	In Zen Explorer, add a new server and specify the name (or IP address) of the *cluster*.
	Access the properties for the server you just added. If prompted to log in, log in as user admin. Leave the password blank. Access the **Directories** Properties. For **Transaction Log Directory**, specify the directory that you created for the "log" location. For **DBNames Configuration Location**, specify the directory that you created for the "etc" location. See Create the Subdirectories on the Mounted Shared Storage.
	Use ZenCC to add a new server and set its properties from each of the other cluster nodes. Place all nodes into standby mode except for the one from which you run ZenCC.
Create the Database on the Shared Storage	From the operating system on one of the cluster nodes, log on as user **zen-svc** and create the directory under the file system share where you want the database to reside. (If you create the directory as user **root**, ensure that user **zen-svc** has read, write, and execute authority on the directory.)
	Place all cluster nodes into standby mode except for the one from which you will run ZenCC.
	As user **zen-svc**, start ZenCC on the one active node or from a client that can access the active node. Create a new database for the server you added in Configure the Cluster Server in ZenCC. For **Location**, specify the directory you created where you want the database to reside. Specify the other database options as desired.
	For the new database, create tables as desired.
Verify Access to the Database from each Node	Each cluster node must be able to access the Zen database on the shared storage. Place the cluster node from which you created the database into standby mode. This is the node running the zen-svc resource (the database engine).
	Fail over to the next node in the cluster. Verify that the next node receives control of running the zen-svc resource. Repeat the standby, fail over and verification process for each node in the cluster until you return to the node from which you began.

Managing Zen in a Cluster Environment

After you install Zen in a failover cluster environment, you can manage it as a resource. The following items discuss common management topics:

- Zen Licensing and Node Maintenance
- Zen Failure Behavior
- Stopping or Restarting the Actian Zen Engine Service
- Zen Configuration Changes
- Software Patches

Zen Licensing and Node Maintenance

The normal Zen licensing and machine maintenance procedures also apply to the nodes in a failover cluster environment. Deauthorize the Zen key before you modify the configuration of the physical or virtual machine where the database engine is installed. Reauthorize the key after changes are complete.

See To Deauthorize a Key and To Authorize a Key in *Zen User's Guide*.

Zen Failure Behavior

If a cluster node fails, a Zen client does not automatically reconnect to the Zen engine on the surviving node. Your application must reconnect the client to the Zen database or you must restart the application. This applies even if Enable Auto Reconnect is turned on for the database engine.

If transaction durability is turned off and a failure occurs before a transaction completes, the transaction is automatically rolled back to its state before the transaction began. That is, to the last completed check point. The rollback occurs when the active server requests access to the data file.

If transaction durability was turned on, completed changes can be recovered that occurred between the time of the cluster node failure and the last check point. Transaction durability must be configured the same way on all nodes and the transaction log located on the shared storage. Transactions that had not completed at the time of the cluster failure, however, are lost even if transaction durability was in effect.

Stopping or Restarting the Actian Zen Engine Service

A cluster failover occurs from the active node if you manually stop the Zen database engine service through the operating system. If you are performing service node maintenance and want to avoid such a failover, stop the service through the cluster utilities.

Zen Configuration Changes

Some configuration changes require that you restart the database engine. See Configuration Reference.

To stop and restart the Zen database engine service to apply configuration changes

Use the following steps in the Windows Cluster Administrator in the order listed:

1. Right-click the Actian Zen engine service and select **Bring this resource offline**.

2. Right-click the Actian Zen engine service and select **Bring this resource online**.

Software Patches

At some point, you may need to patch Zen or the failover cluster software. To help you do so, Actian Technical Support provides a knowledge base article on this topic.

Migration

Migration moves a VM running Zen from one physical host to another. The memory, storage, and network connectivity of the VM are typically migrated to the destination. Depending on the hypervisor, migration is sometimes referred to as "live" migration or "hot" migration.

With a "live" or "hot" migration, client connections to Zen remain intact. This allows changes to hardware or resource balancing. With a "cold" migration, network connectivity is interrupted because the VM must boot. Client connections to Zen must be reestablished.

A migration environment has only one instance of Zen running, which makes the environment somewhat vulnerable if the host machines crashes or must be quickly taken offline. Also, if the shared storage fails, the database engine cannot process reads from or writes to physical storage. Some hypervisors offer a migration solution that does not use shared storage.

As long as host names remain the same after the VM migrates, Zen continues to operate normally. The product key remains in the active state.

No special steps are required to install or configure Zen in a migration environment. Refer to the hypervisor documentation.

Fault Tolerance

A fault tolerant environment is similar to a migration environment but includes additional features to ensure uninterrupted operation even after the failure of a component. A fault tolerant environment ensures network connections, continuous service, and data access through synchronized shared storage. If a component switch occurs, client machines and applications continue to function normally with no database engine interruption.

No special steps are required to install or configure Zen in a fault tolerant environment. Refer to the hypervisor documentation.

Disaster Recovery

Disaster recovery includes data recovery and site recovery. Data recovery is how you protect and restore your data. Site recovery is how you protect and restore your entire site, including your data.

Data recovery is facilitated with the hypervisor shared storage and Zen transaction logging and transaction durability. See Transaction Logging and Durability. You can use transaction logging and transaction durability with Zen Enterprise Server and Cloud Server.

Site recovery can be accomplished with both physical machines and virtual machines. Zen operates normally provided that host names remain the same in the recovered site. This is typically the case for virtual machines. If you are recovering physical machines and the host name at the recovery site is different, the Zen product keys will change to the failed validation state when Zen starts. Zen will continue to operate normally in the failed validation state for several days, during which you can either repair the key or move back to the original site.

No special steps are required to install or configure Zen in a disaster recovery environment. Refer to the hypervisor documentation.

Zen and Hypervisor Products

The following topics help you to use Zen most effectively with a hypervisor product:

- Hypervisor Product Installation
- Usage Topics for Zen

Hypervisor Product Installation

As with any complex software, hypervisor products can be installed various ways, including with nonstandard and atypical configurations. For compatibility with Zen, install and configure the hypervisor product using the best practices recommendations of the product vendor.

Usage Topics for Zen

This section discusses the following topics for using Zen:

- Physical Machine To VM Migration
- Configuration
- VM Resource Pools and Templates
- Failover Cluster Support
- Performance
- Data Backup

Physical Machine To VM Migration

You can install Zen Enterprise Server, Cloud Server, or Workgroup on physical machines initially and later transition to VMs as your business needs change. For migrations from physical machines to VMs, you must ensure that the host name remains the same.

The Zen engine does not depend on IP address, but the VM itself might. If your VMs depend on raw IP addresses, or on the hosts file, rather than on the Domain Name System (DNS), ensure that you also take appropriate actions concerning IP addresses.

Configuration

No special steps are needed to configure Zen to use hypervisor product features such as live migration, fault tolerance, high availability, paravirtualization, resource scheduling and disaster recovery. Zen remains authorized and fully functional provided that the host name remains consistent.

Certain scenarios, such as for disaster recovery, may require network and hardware changes. You may change the following without adversely affecting Zen:

- IP or MAC address of the VM
- Hardware in the VM, such as CPU type, CPU speed, amount of memory, and type and size of storage.

Note that Zen is not aware of certain hardware changes, such as increasing memory or physical storage, if the database engine is running. You must stop and restart the database engine if you want it to be aware of such changes.

VM Resource Pools and Templates

Zen may be used with VM resource pools and templates. For both uses, each copy of Zen requires its own product key. See License Enforcement in *Zen User's Guide*.

Resource Pools

Zen must be authorized in each VM within a resource pool that includes the database engine.

Templates

To authorize Zen in a VM launched from a template, you may use a configuration script. The script can invoke the CLI License Administrator tool to authorize the Zen key during the customizing of the guest operating system. See License Administrator Command Line Interface in *Zen User's Guide*.

Remember to customize other properties of the guest operating system, such as host name, that are independent of running Zen.

Failover Cluster Support

As a general guideline, if you use affinity rules, ensure that all cores are running on the same socket. This aids performance of Zen because of its multicore support. Anti-affinity rules may also be used depending on your configuration.

If you use Raw Device Mapping (RDM) as the data drives for MSCS configurations, be aware of the considerations. Refer to the vendor documentation for RDM.

If you use fault tolerance/high availability with Distributed Resource Scheduler (DRS), be aware that load balancing can be done only after failover. Refer to the vendor documentation for DRS.

Performance

To achieve the best performance for Zen, ensure the following:

- Adherence to the performance best practices recommendations from the hypervisor vendor.
- The VM hosting Zen has ample memory (RAM).
- The hypervisor host has enough virtual CPUs to minimize virtual CPU contention among all of the VMs on the same machine. This prevents contention with the VM running Zen. If you use affinity rules, ensure that all cores are running on the same socket.

- The Zen data files reside on very fast physical storage and minimize spindle contention and controller contention for the physical storage device.

Data Backup

See Data Backup with Backup Agent and VSS Writer.

Workgroup Engine in Depth

The following topics explain how to get the most out of Workgroup Engine.

- Networking
- Technical Differences Between Server and Workgroup
- Troubleshooting Workgroup Issues
- Redirecting Locator Files

Networking

Both the server and workgroup products are shipped with the same networking components. So you can upgrade a Workgroup engine to a Zen server engine. The client-side requesters can connect to either type of engine.

The client-side MicroKernel router discovers gateway ownership of files by following a well-established algorithm.

Priority	Procedure
1	Try to connect to a database engine on the same computer as the data files.
2	Try to open the data files using the local engine (on the client machine).
3	Find and connect to the gateway engine that owns the files.

The first thing the client-side router attempts is to connect to an engine on the same computer as the data. Because of this, it is always more efficient to have an engine running where the data resides.

Because the Zen network services layer uses many methods to find and connect to a remote database engine, there may be a delay on the first attempt to open a file on a file server that does not have a database engine running. If Gateway Durability is turned on, that connection will not be attempted thereafter because the router remembers each machine on which it fails to locate an engine.

If the router cannot connect to an engine on the remote file server, the router then allows the local engine to attempt to open the remote files. The local engine first attempts to create a new locator file and take ownership of the remote directory. If the directory is already owned by another MicroKernel, the local engine returns status code 116 to the router.

Finally, the router attempts to discover the gateway computer. It opens the locator file and reads the name of the gateway engine. Then it sends the request to that engine. Notice that the router never tries to read a locator file unless it has first received status code 116 from a MicroKernel. Accordingly, in order to use the gateway feature, you must have a local workgroup engine installed. If the attempt to open the remote files with the local engine fails because there is no local engine, the router does not try to read the locator file and no gateway engine is found.

Technical Differences Between Server and Workgroup

Zen server and Workgroup engines have a few significant differences.

Platforms

The Zen server engine offers 64- and 32-bit editions for various platforms. The Workgroup engine has only a 32-bit Windows edition. Although the Workgroup engine can run on a 64-bit Windows operating system, it is restricted to an address space of only 4 GB and cannot take advantage of the larger amounts of memory typically installed on such systems. When the Workgroup engine is installed on a 32-bit operating system, it has an address space limit of 2 GB.

User Interface

On Windows, a Zen server engine is always installed to run as a Windows service. The Workgroup engine can be installed as an application or as a service. In a fresh installation, it is installed by default as a service. If you choose instead to install it to run as an application, it uses an icon in the notification area of the taskbar as an interface. See Setting Up a Workgroup Engine in *Getting Started with Zen*.

Authentication and Btrieve Security Policies

The Zen database engine enforces file permissions in the operating system based on operating system user authentication. The Workgroup engine does not authenticate users on its own. If the Workgroup engine can connect to a system on a network, it can access the data. The more relaxed Workgroup security is intended for small offices where security may be less important than ease of use.

The lack of operating system authentication for the Workgroup engine means that the Mixed security policy for Btrieve is the same as the Classic security policy. This difference in security policy is a behavior difference between the Zen server and Workgroup engines. For more information, see Zen Security.

Gateway Support

The Workgroup engine creates locator files everywhere it opens files, both locally and remotely, allowing the engine to dynamically adjust gateway ownership daily. By default, the Workgroup engine also runs under a user ID, which can be authenticated on other computers and network

devices. This makes the Workgroup engine ideal for use in a gateway environment. See Setting Up a Gateway Configuration in *Getting Started with Zen*.

The Zen server engine does not always create or honor gateway locator files. Because of this, it is not designed or tested for use in a gateway environment. Therefore, we do not support replacing a Workgroup engine with a server engine as a gateway in a workgroup environment.

Asynchronous I/O

The Zen server for Windows makes use of asynchronous I/O. Furthermore, coalescing of database page writes is done only by a Zen server. These features can provide a significant performance advantage for the server engine over the Workgroup engine during heavy I/O usage.

Default Configurations

The default values for some database settings, such as cache size and system cache, differ between a Zen server engine and a Workgroup engine. The default values for Workgroup engine options are set to consume less system resources. See Configuration Reference.

License Model

Zen Enterprise Server and Zen Workgroup use a user count license model. Zen Cloud Server uses a capacity-based license model. See License Model in *Zen User's Guide*.

Troubleshooting Workgroup Issues

This section provides a few tips on troubleshooting problems in a Workgroup environment.

Time delay on first connection

If you regularly experience a delay when the first file-open request is issued, see if these techniques help.

If possible, make sure an engine running on the system where the data resides

Connecting to an engine on the same machine as the data is first priority for a client deciding where to send a file-open request. To ensure a Workgroup engine is running as an application, put an engine icon in the startup folder with the command:

`zenengnapp.exe`

Another option is to install the Workgroup engine as a service. For more information, see *Getting Started with Zen*. Also, for default locations of Zen files, see Where are the files installed?

If you are running a gateway topology

If you cannot run an engine where the data is, then the time delay during the first connection is a more important issue. Here are a few things you can do.

1. Reduce the supported protocols in the client settings so that protocols that are not used in your network are not attempted.

2. Use Gateway Durability. Gateway Durability is a client configuration setting that allows you to virtually eliminate the delay in making the first connection in a gateway environment. If Gateway Durability is turned on, it forces the client router to write into the registry the names of computers it finds that do not have an engine running. Once a failure to connect happens, instead of remembering this server name only while the router is running in-process, it saves the name in the registry. The next time the application starts up, it does not try to connect to the engine where the data is. It immediately goes to the next step of determining the identity of the current Gateway.

    You can turn this setting on in ZenCC. Within ZenCC Zen Explorer, expand Local Client node, then right-click MicroKernel Router. Click **Properties** then click **Access**. Click the **Gateway Durability** option to set it to on (a check mark indicates that the setting is on) then click **OK**.

Note: This feature is off by default since it fixes the topology. If you add a server engine or a Workgroup engine where the data is, you must turn this setting back to off on each of the clients where you turned it on. Turning the setting off erases the registry of computers without an engine running, so you can turn it back on immediately and a new list will be generated based on the new topology.

Status Code 116

Status 116 is a MicroKernel status code which means that the file is being used by another MicroKernel engine acting as a gateway. If your application receives a status code 116, it means that the MicroKernel router can read the locator file but cannot contact the engine running on the gateway computer.

The first thing you need to do is find out who the gateway is. You can perform this task with the Gateway Locator tool.

Next, use Zen System Analyzer (ZenSA) network tests to try to connect to that computer. ZenSA can provide valuable information to isolate the problem.

One situation when this could occur is when the two computers are separated by a router such that they can both see the file server but they cannot see each other.

Redirecting Locator Files

This feature of gateway engine operation guarantees transaction atomicity for multidirectory databases and also makes it easy to change the name of a gateway engine across multiple data directories.

Recall that the Zen client uses the following approach to access remote data files:

1. First, attempt to connect to a database engine on the same computer as the data files.

2. Second, if no database engine is available on the remote machine, attempt to use a local engine to take ownership of the remote directory and create a locator file. If a gateway locator file already exists, the local engine is not used.

3. Third, try to use the specified gateway engine.

It is important to remember that the gateway configuration only goes into effect when there is no database engine available on the same computer as the data files.

This feature allows a dynamic (floating) gateway engine while at the same time preserving transaction durability for multidirectory databases on the same volume. This benefit is provided by a new type of gateway locator file that points to another gateway locator File. The new type is called a *redirecting locator file*. By having redirecting locator files in directories A, B, and C that point to the locator file in directory D, you can ensure that the gateway engine specified by the locator file in directory D services data files in the other directories as well.

Regardless of whether the locator file in directory D specifies a permanent gateway or is dynamically created by the first engine to open those files, this architecture ensures that all the specified directories use the same gateway engine. Likewise, if you decide to change the permanently assigned gateway engine for several directories, redirecting locator files allow you to do so by changing only one locator file, rather than all of them. Thus, it is possible to specify that all data files on a given hard drive must use the same gateway engine, with or without designating a permanent gateway.

Redirecting Locator File Requirements

The first line of a redirecting locator file must start with "=>" and be followed by a path specifying another locator file, which must be on the same drive. You can use any combination of slash and backslash in the path name. All slashes are converted to the type of separator used by the local operating system.

If your specified path ends with a slash, the database engine assumes the default locator file name (~PVSW~.LOC) and appends it to the path. If the specified path does not end with a slash, the database engine assumes that the path already contains the file name.

The following table lists the ways a redirecting locator file path can be specified:

Path	Meaning
=>\path_name	Specifies the path from the root of the drive where the current Locator File is stored.
=>.\path_name	Specifies the path relative to the current directory.
=>..\path_name	Specifies the path relative to the parent directory of the current directory.

You can assign multiple levels of redirection to these locator files. For example, you can have the first locator file pointing to a second locator file, the second locator file pointing to a third locator file, and so on. Each Workgroup engine opens each locator file sequentially, looking for the actual gateway name. It stops searching once it has found the locator file that does not start with "=>". The engine then assumes this locator file specifies the gateway engine.

Creating Redirecting Locator Files

As with any locator file, a redirecting locator file is a plain text file. You can create redirecting locator files by hand or programmatically. A redirecting locator file must be flagged as read-only, or it will be overwritten by the first engine to attempt to access the data files in that directory.

To Create a Redirecting Locator File

1. Open a text editor and create a new text file.

2. Decide where you are going to save the file when you are finished. You will save the file in the same directory as the data files which you want to redirect to another locator file.

   For example, if you want to ensure that the data files in C:\data are accessed by the same Gateway engine as other data files, then you will want to keep in mind the folder C:\data.

3. Enter `=>` and the path name of the next locator file. Continuing the example from the previous step, if you want the current data files in C:\data to be owned by the gateway engine specified in the locator file located in C:\moredata, then you would enter the following:

   ```
   =>..\moredata\ (recommended) or
   =>\moredata\ (not recommended)
   ```

In the first case, you are specifying a relative path from the current directory. In the second case, you are specifying an absolute path from the root of the current drive. In this particular example, both cases resolve to the same target directory.

Note: It is strongly recommended that you use relative path names (starting with ./ or ../) in your redirecting locator files, and also that you use the same share names on all workstations to access the same data. Following these two recommendations can prevent errors that may occur with network path name resolution over mapped drives.

4. Save the file as ~PVSW~.LOC in the data file directory where you want to configure a Gateway engine.

5. Close the text editor.

6. Flag the text file as read-only.

    To mark the file as read-only, you can use the Properties dialog box (right-click the file icon) in Windows Explorer, or you can use the ATTRIB command at a prompt or in a program:

    ```
    attrib +r ~pvsw~.loc
    ```

To synchronize many data directories on a permanent Gateway

1. Either by hand or by using the Gateway Locator program, create a read-only (permanent) locator file that does not redirect. It must specify a Workgroup engine to use as the gateway.

    For example, your locator file may specify the computer named workgroup1 as the gateway engine, and the file may be located in C:\data\db1.

2. For each of the other data directories that you want to use the gateway engine specified in the previous step, you need to create a Redirecting Locator File in that directory. Each redirecting locator file must point to the file you created in the previous step.

    Continuing the example, each redirecting locator file in C:\data\db2 and C:\data\db3 would then contain the following text:

    ```
    =>..\db1\
    ```

    This causes any engine reading this file to follow the relative path and search the specified directory C:\data\db1 for another locator file. In this case, the specified directory contains a locator file that names workgroup1 as the gateway computer.

To synchronize many data directories on a dynamic gateway

1. Follow the steps above, only in step 1, ensure that the locator file is writable, not permanently-assigned.

In this case, remember that if no engines are accessing any data files in the redirecting hierarchy, then there will be no locator file in the target directory. This is normal. The dynamic locator file is created each session by the first engine to access the data, and the file is deleted when the last user session ends. It is permissible to have redirecting locator files that point to a data directory that has no locator file in it. In this case, the first engine to open those data files creates the locator file.

Example

Using the example locator files shown in the following figure, the redirecting locator file on the left forces the database engine to go up one directory, then look in the subdirectory newdir for another locator file with the default name (~PVSW~.LOC). This locator file, in turn, specifies that the Workgroup engine on the computer named ntserver1 is the correct gateway engine. As a result, the database engine on ntserver1 is used to access the data files in the directory mydir.

mydir/~pvsw~.loc

```
=>../newdir/
```

newdir/~pvsw~.loc

```
ntserver1
```
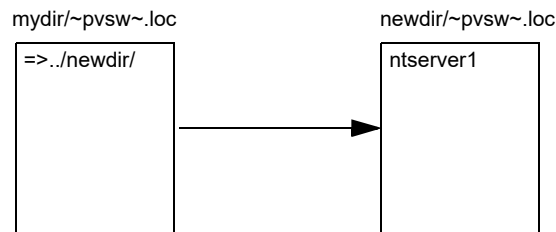
Monitoring

Zen is designed for ease of use and minimal administration. At times, however, you may want to monitor various conditions or resources in the Zen environment. For example, you may need to tune portions of your application and monitor various performance aspects pertaining to the database. Or, you may have adjusted Zen configuration settings for your business needs and want to monitor the changes from the default settings.

Monitoring is covered in the following topics:

- Monitoring Database State
- Monitoring Data File Fragmentation
- Monitoring Performance Counters
- Monitoring License Usage
- Monitoring Database Access
- Reviewing Message Logs
- Receiving Email Notification of Messages

Monitoring Database State

You can monitor the state of your database with the Monitor tool. It allows you to oversee the following:

- Active Files
- MicroKernel Sessions
- Resource Usage
- MicroKernel Communication Statistics
- SQL Active Sessions

If you are new to Monitor, or want an overview of the tool, see Monitor Overview.

Monitor Overview

Monitor is a tool that allows you to systematically observe certain activities and attributes of the database engine. It provides information useful for both database administration and application programming diagnostics. It can monitor aspects of both the MicroKernel Engine and the Relational Engine.

Monitor has two interfaces, both of which provide the same functionality:

- Graphical Interface Monitor presents information in a series of tabs.
- Command Line Interface Monitor uses an executable program that directs the information to a selected location.

Graphical Interface Monitor

ZenCC includes a Monitor tool that presents information organized into a series of tabs. The tabs can be rearranged for your convenience, with columns of data that can rearranged and sorted. It presents a snapshot of a particular moment and can be refreshed either manually or automatically.

Using GUI Monitor

Within ZenCC, you access Monitor from Zen Explorer.

To access Monitor for a database engine

Use one of the following options to open the Monitor window. You can monitor multiple engines at the same time if you choose.

- In Zen Explorer, locate and open the Engines node. Right-click the database engine that you want to monitor, then click **Monitor** on the context menu.

- Click **Tools** > **Monitor**. A dialog asks you to select a server to monitor.

- To launch Monitor in a stand-alone window, such as for a third-party application, at a prompt you can use the -monitor_any parameter:

```
zencc.exe -monitor_any
```

A dialog asks you to select a server to monitor.

GUI Features

The table below the following image explains the user interface components. Click any area of the image for which you want more information.

GUI Object	Description	Related Information
File menu	Allows you to set the refresh rate.	Setting Refresh Options
Window menu	Allows you to display the database window of the ZenCC and individual Monitor tabs, and to set preferences.	Setting Monitor Preferences
Help menu	Allows you to access the documentation and logs.	Zen Event Log (zen.log)
Tab Set area	Includes five tabs with grids that display information about activity occurring at that moment in the database. When you select a tab, a selection of icons that apply to that tab appear at the right of the row of tab labels.	Tab Functionality

Setting Monitor Preferences

You can set preferences for the Monitor from either Monitor itself or from ZenCC. In Monitor, select **Window** > **Preferences** > **Monitor** to open the **Monitor** tab of the **Preferences** dialog. In ZenCC, select **Window** > **Preferences** > **Zen** > **Server Monitor**.

You can set two types of preferences. First, you can set the layout of the Monitor user interface, so that when you reopen Monitor, its tabs are arranged just as you left them. You can set this individually for each server that you access. Second, you can set the features for any particular grid, for example, column width, sort order, and column order. If you open that particular grid on another server, it follows the same settings. That way, you can easily compare the same grid on different servers.

Setting Refresh Options

Information in the Monitor can be refreshed automatically at a configured interval, as desired, or not at all. Be aware that refreshing too many windows at a short interval may slow performance.

Automatic refresh	1. Use the **Set Automatic Refresh** icon to turn on automatic refresh.
	2. Either select **File** > **Set Automatic Refresh Rate** or click the **Set Refresh Rate** icon to open the **Set Refresh Rate** dialog box.
	3. In the **Set Refresh Rate** dialog box, enter the number of second between each refresh and click **OK**.
Refresh as desired	Either select **File** > **Automatic Refresh** or click the **Refresh** icon.
No refresh	Use the **Set Automatic Refresh** icon to turn off automatic refresh.

Tab Functionality

Tabs can be rearranged, separated, and reaggregated for your convenience. To move a tab, put the cursor on the tab label, hold down the left mouse button, and pull the tab label where you want the tab to be.

Because of the different nature of the data on each tab, different operations can be performed on each tab. Those operations are initiated by the icons that appear at the right end of the row of tab labels. The following table describes the icons.

Icon	Meaning
Set Automatic Refresh	Toggles between automatic refresh on and automatic refresh off. See Setting Refresh Options.
Set Refresh Rate	Displays a dialog box in which you can set the number of seconds between each repopulation of the user interface with current information. See Setting Refresh Options.
Refresh	Repopulates the user interface with the current information.
Select Columns to Display	Displays a dialog box in which you can select which columns to display. Because each tab includes different columns, each dialog box includes a different list of column names. All dialog boxes include the choices **Select All** and **Deselect All**.
Hide/Show Handles	Toggles between displaying or hiding the **Handle Information** grid at the bottom of the tab. The **Handle Information** grid includes different information, depending on the tab. The **Handle Information** grid also has a **Select Columns to Display** icon, so that you can hide or display columns as desired.
Delete Selected Session	Removes a highlighted session from the tab. This icon becomes activated only after a session is selected. **Caution:** This procedure actually terminates a session, so you are interrupting someone's work in progress. Consequently, a message asks you to confirm that you actually want to do this.
Delete All Sessions	Removes all sessions from the tab. **Caution:** This procedure actually terminates all sessions, so you are interrupting work in progress. Consequently, a message asks you to confirm that you actually want to do this.
Reset the Deltas	Deletes a statistic and restarts the count at zero.

Monitoring Active Files

The **Active Files** tab provides information about MicroKernel files that are currently open. To select columns to monitor, see Select Columns to Display.

Path	Provides the directory and all subdirectories to the location of the file.
File	Indicates the name, including suffix, of the file.
Page Size	Indicates the size in bytes of each page in the file.
Read-Only	Indicates whether the file is flagged as read-only by the operating system.
Record Locks	Indicates whether any of the active handles for the selected file have record locks. Any application can read a locked record, but only the application that placed the lock can modify or delete the record. A record lock exists only as long as the application that opened the file is updating a record. "Yes" indicates that one or more record locks are applied to the file. "No" indicates that no records are locked.
Transaction Lock	Indicates whether any of the active handles for the selected file have a transaction lock. A transactional file lock exists only as long as the application that opened the file is processing a transaction.
Physical File Size	Indicates the size of the file in kilobytes (KB). This information is particularly useful for the capacity-based license model if you want to review data in use on a file-by-file basis.
	Monitor uses kilobytes (KB) for the size of an individual file and megabytes (MB) as the units for resource usage (Monitoring Resource Usage). License Administrator uses gigabytes (GB) as the units because that is how data in use is associated with a key. The different contexts require units appropriate for each context.
	If a file is immediately closed after you insert a large number of records, Monitor does not immediately reflect the changes in file size. For example, the statistics for Physical File Size KB are not refreshed for that file until the next time the file is opened for reading or writing.

You can view the handle information for any active file. See Hide/Show Handles. Active file handles include the following information.

Client	Indicates the name (typically the login ID of the user) or an index to the Client list of the database server.
Connection Number	Displays the network connection number of the client. If the client does not have a network connection, this field displays NA for "not applicable."

Task Number	Displays the process-supplied task number for processes originating at the server or a Windows Client.
Site	Specifies the location of the user process (local or remote).
Network Address	Identifies the location of the calling process on the network.
	If the calling process is TCP/IP, the address is preceded by T4 for IPv4 addresses, T6 for IPv6 addresses, and T for the fully qualified domain name of a client machine. Examples: T4: 180.150.1.24 T6: 1234:5678:0000:0000:0000:0000:9abc:def0 T: *<mymachine.mydomain.mycompany>*.com
Open Mode	Indicates the method the application uses to open the specified handle of the file. Valid open modes are the following: • Normal – The application that opened the file has normal shared, read/write access to it. • Accelerated – The application that opened the file has shared read/write access. • Read-only – The application that opened the file has read-only access; it cannot modify the file. • Exclusive – The application that opened the file has exclusive access. Other applications cannot open the file until the calling application closes it. Monitor also specifies all open modes as *nontransactional* or *shared locking* when applicable.
Record Lock Type	Displays the types of record locks currently held by the handle. The possible record lock types are Single, Multiple, and None. Single-record locks enable a user to lock only one record at a time. Multiple-record locks enable a user to lock more than one record at a time.
Wait State	Indicates whether the user is waiting because of some type of lock on this handle: Waits for Record Lock, Waits for File Lock, or None.
Transaction State	Displays the state of the transaction lock currently held by the handle. The possible transaction types are Exclusive, Concurrent, or None.

Monitoring MicroKernel Sessions

The **MicroKernel Sessions** tab provides information about current connections to the MicroKernel Engine. To select columns to monitor, see Select Columns to Display.

Session	Provides a unique identifier for the connection. A session is defined as a client ID used by the MicroKernel Engine or a connection to the Relational Engine. Client ID is defined as a 16-byte structure that combines elements provided by the application, by the client platform, and by the database engine to uniquely identify a database transaction context.
	Session information reflects the sessions established through the MicroKernel Engine **and** through the Relational Engine. If you want to view sessions established only through the Relational Engine, see Monitoring SQL Active Sessions.
	This tab allows you to delete a session or all sessions. See Delete Selected Session and Delete All Sessions.
Connection Number	Displays the network connection number of the session. If the session does not have a network connection, this field displays NA for "not applicable."
Task Number	Displays the process-supplied task number for processes originating at the server, or from a Windows Client.
Site	Specifies the location of the session process (local or remote).
Network Address	Identifies the location of the calling process on the network.
	If the calling process is TCP/IP, the address is preceded by T4 for IPv4 addresses, T6 for IPv6 addresses, and T for the fully qualified domain name of a client machine. Examples: T4: 180.150.1.24 T6: 1234:5678:0000:0000:0000:0000:9abc:def0 T: <*mymachine.mydomain.mycompany*>.com
	If multiple clients from a single machine connect by different TCP/IP addresses, each address is valid for that client. However, internally to the database engine, an address associated with a client may not be the actual address used by that client. This is because of the way the database engine identifies and manages multiple clients from the same machine. Consequently, since Monitor is reporting engine information, the tool may display an associated address instead of the actual address.
Locks Used	Indicates the number of locks the session is currently using.
Transaction State	Displays the type of transaction lock the session currently holds. The possible transaction types are Exclusive, Concurrent, or None.
Read Records	Displays the number of records read since the session first opened a file.
Inserted Records	Displays the number of records the session has inserted.

Deleted Records	Displays the number of records the session has deleted.
Updated Records	Displays the number of records the session has updated.
Disk Accesses	Indicates the number of times the session required a disk access. You will not see any information for disk accesses for files that have just been opened.
Cache Accesses	Indicates the number of times this client finds data in L1 or L2 cache in order to fulfill the request.

You can view the handle information for any MicroKernel session. See Hide/Show Handles. MicroKernel session handles include the following information.

Path	Provides the directory and all subdirectories to the location of the file.
File	Indicates the name, including suffix, of the file.
Open Mode	Indicates the method the application uses to open the specified handle of the file. Valid open modes are:
	Normal – The application that opened the file has normal shared, read/write access to it.
	Accelerated – The application that opened the file has shared read/write access.
	Read-only – The application that opened the file has read-only access; it cannot modify the file.
	Exclusive – The application that opened the file has exclusive access. Other applications cannot open the file until the calling application closes it.
	Monitor also specifies all open modes as *nontransactional* or *shared locking* when applicable.
Record Lock Type	Displays the types of record locks currently held by the handle. The possible record lock types are Single, Multiple, and None.
	Single-record locks enable a user to lock only one record at a time. Multiple-record locks enable a user to lock more than one record at a time.
Wait State	Indicates whether the user is waiting because of some type of lock on this handle: Waits for Record Lock, Waits for File Lock, or None.
Transaction State	Displays the state of the transaction lock currently held by the handle. The possible transaction types are Exclusive, Concurrent, or None.

Monitoring Resource Usage

The **Resource Usage** tab displays the resources in use by the MicroKernel since the engine was last started. To select columns to monitor, see Select Columns to Display.

The database engine dynamically controls the maximum values for some of these resources. The maximum value for User Count, Session Count, and Data In Use depends on the product license. See License Model in *Zen User's Guide*.

For example, User Count does not apply to Cloud Server. Therefore, "n/a" appears as the Current, Peak, and Maximum value for User Count if Cloud Server is being monitored. Similarly, "n/a" appears as the Maximum value for Session Count and Data in Use MB if a Zen server is being monitored.

If you are considering using Cloud Server, you need the ability to estimate Current and Peak values for Session Count and Data in Use MB. Consequently, those statistics are displayed for the server without being enforced. No notifications are sent about them regardless of their values.

Engine Uptime	Lists the amount of time in weeks, days, hours, and minutes that the MicroKernel engine has been running.
	Engine Uptime is not a selectable column for Resource Usage. It is part of the grid title.

Resource	Indicates the type of resource being monitored. See Monitoring Resource Usage.
Current	Indicates the current usage by a resource.
Peak	Indicates the the highest value for a resource since the MicroKernel was started.
Maximum	Indicates the highest allowed value for a resource.

The following table lists the types of resources for usage monitoring.

Files	Indicates the number of files currently open by the MicroKernel. The maximum for this resource is unlimited.
Handles	Indicates the number of active handles. The MicroKernel creates a handle each time a user opens a file. A single session can have several handles for the same file. The maximum for this resource is unlimited.

Clients	Indicates the number of clients accessing the MicroKernel. A machine can have multiple clients accessing the database engine simultaneously. The engine dynamically manages the client list. The maximum for this resource is unlimited (the number of clients is limited only by the memory in the computer).
	"Client" indicates a session established by a client ID (transactional engine interface) or a connection to the relational engine interface. The database engine uses various client sessions for its own internal processes, such as for accessing Zen system files, metadata files, dbnames.cfg, and default system databases. The number of clients indicates both internal client sessions and non-internal client sessions (see Monitoring MicroKernel Sessions).
Worker Threads	Indicates the number of concurrent MicroKernel processes.
User Count	Indicates the number of concurrently connected users. The maximum value shows the maximum permitted users as granted by a license agreement.
Session Count	Indicates the number of sessions in use by the database engine. For brevity, "number of sessions in use" is also referred to "session count." The maximum value (also called the "session count limit") shows the maximum permitted sessions as granted by a license agreement.
	Session count reflects all sessions whether established through the MicroKernel Engine or through the Relational Engine.
	Messages pertaining to session count are logged to the various Zen logging repositories. See Reviewing Message Logs.
	The database engine uses various sessions for its own internal processes, such as for accessing Zen system files, metadata files, dbnames.cfg, and default system databases. These internal sessions do not consume any session counts.

Data In Use MB	Indicates in megabytes (MB) the size of all concurrently open data files. The maximum value is the maximum permitted amount of all concurrently open data files as granted by a license agreement. The maximum is also called the "data in use limit."
	The value for data in use increases when a data file is first opened. Subsequent opens to an already open data file do not add to the total. Data in use also increases if an open file increases in size. Operations on an already open file continue to be permitted even if the size of the open file increases beyond the data in use limit.
	The value for data in use decreases when a data file is closed by the final user to have the file open. Since more than one user can access the same data file, *all* opens must be closed before data in use decreases.
	Messages pertaining to data are logged to the various Zen logging repositories. See Reviewing Message Logs.
	The database engine uses various files for its own internal processes, such as Zen system files, metadata files, dbnames.cfg, and default databases. Files used for internal processes do not increase the value for data in use.
	If a file is immediately closed after you insert a large number of records, Monitor does not immediately reflect the changes in file size. For example, the statistics for "Data in Use MB" are not refreshed for that file until the next time the file is opened for reading or writing.
Transactions	Indicates the number of transactions. The maximum for this resource is unlimited.
Locks	Indicates the number of record locks. The maximum for this resource is unlimited.

Monitoring MicroKernel Communication Statistics

The **MicroKernel Communication Statistics** tab displays information about communication with the MicroKernel Engine. It includes separate sections for Communications Statistics and Resource Usage Information. Communication statistics are calculated in terms of total number of occurrences processed since the database engine was started.

To select columns to monitor, see Select Columns to Display.

Engine Uptime	Lists the amount of time in weeks, days, hours, and minutes that the MicroKernel engine has been running.
	Engine Uptime is not a selectable column for MicroKernel Communications Statistics. It is part of the grid title.

Communications Statistics

Resource	Indicates the type of resource being monitored. See Monitoring MicroKernel Communication Statistics.
Total	Indicates the total number of occurrences processed since the database engine was started.
Delta	Indicates the number of occurrences processed since you last accessed the MicroKernel Communications Statistics tab. To restart the count of the delta number, see Reset the Deltas.

Total Requests Processed	Indicates the number of requests the database engine has handled from workstations or remote, server-based applications.
TCP/IP Requests Processed	Indicates the number of TCP/IP requests the database engine has handled from clients or remote, server-based applications.
Connection Timeouts	Indicates the number of times Auto Reconnect has timed out when attempting to reconnect to Clients. See also Auto Reconnect Timeout.
Connection Recoveries	Indicates the number of times the Auto Reconnect feature has successfully recovered from a connection timeout.

Resource Usage Information

Resource usage information provides current, peak, and maximum values for resource occurrences.

Resource	Indicates the type of resource being monitored. See Monitoring MicroKernel Communication Statistics.
Current	Indicates the current usage by a resource.
Peak	Indicates the highest value for a resource since the MicroKernel was started.
Maximum	Indicates the highest value allowed for a resource.

Communications Threads	Indicates the number of remote requests that the MicroKernel is currently processing. Local requests are not included in this statistic. For the total number of remote and local threads being processed, see Monitoring Resource Usage.
	The database engine dynamically increases the number of communications threads as needed up to the maximum allowed. For Windows and Linux, the maximum is 1024.
	Worker threads are also used to process Monitor requests, so you might not see the number of current worker threads drop below one. This is normal.
Total Remote Sessions	Indicates the number of remote clients connected to the database engine. The maximum number is dynamic and displays as zero.
TCP/IP Remote Sessions	Indicates the number of remote clients connected through the TCP/IP protocol to the database engine.

Monitoring SQL Active Sessions

The **SQL Active Sessions** tab provides information about current connections to the Relational Engine. This tab also allows you to delete a SQL session. See Delete Selected Session. To select columns to monitor, see Select Columns to Display.

User Name	Provides the login name of the user.
Client Host Name	Identifies the name of the Client machine for the selected User Name. If unavailable, this is set to Unknown.
Network Address	Identifies the Client machine IP address for the selected User Name. If unavailable, this is set to Unknown.
	Values displayed include IP, Shared Memory, and Unknown.
Client Application	Identifies the connected application or module. If unavailable, this is set to Unknown.
Data Source Name	Identifies the name of the DSN referenced by the Client application.

Connection Status	Specifies the connection status for the selected User Name. A status can be any of the following:
	• Active – The session has files open. and that Idle means that the session has no files open.
	• Idle – The session has no files open.
	• Dying – A temporary status that indicates an active session has been deleted but has not finished processing the SQL code. At a suitable termination point, the session is no longer listed on the SQL Active Session dialog.
	• Unknown – Status is unavailable.
Active/Idle Period	Displays the duration of time, in milliseconds, since the connection has been active or idle.
Total ConnectionTime	Displays the duration of time, in seconds, since the connection has been established

Command Line Interface Monitor

The following topics cover bmon, the command line version of Monitor:

- Accessing Bmon
- Configuration File Settings
- Bmon Output

Accessing Bmon

The **bmon** command line tool runs on the Windows, Linux and Raspbian platforms supported by Zen. Its executable program **bmon.exe** on Windows and **bmon** on Linux-based systems. On Windows systems, the tool resides in the bin directory of the Zen installation location. On Linux systems it is located in /usr/local/actianzen/bin.

Configuration File Settings

Bmon manages server settings based on a configuration file. Zen provides a default configuration file **monconfig.txt** in the bin directory of the Zen installation location. The settings in the file are documented with comments.

Bmon Output

Output from **bmon** can be directed to the console, a log file, or both. The configuration file specifies where to direct the output. An application could, for example, check the console or log

file for a particular condition, and then take appropriate action. Zen v15 added support for JSON output with the **-runonce** option.

Command Syntax

```
bmon -f [filepath]config_file [-runonce][-JSON]
```

Options

-f	A required parameter to specify that a configuration file is providing input to the tool.
filepath	The path to the configuration file. If omitted, **bmon** checks the local directory for the configuration file.
config_file	The name of the configuration file. The file name is of your choosing. Installation includes the default file monconfig.txt in the Zen installation bin directory, with comments to document all options.
-runonce	Optional parameter to run bmon once and exit. This parameter is particularly useful when **bmon** is called from a batch file. See also Running Bmon Without the Runonce Parameter. **Note:** In command line environments with no stdin, such as a remote PowerShell session, **bmon** uses this option even if it is not specified. To run **bmon** more than once, use the limitrefresh setting in the **bmon** configuration file.
-JSON	Optional to produce output in JSON format, used only with the **-runonce** parameter.

Running Bmon Without the Runonce Parameter

The **-runonce** parameter is optional. Without it, **bmon** uses the settings in the specified configuration file. In the sample file monconfig.txt, the initial refresh rate is set to 5 seconds. At any time during **bmon** execution, you can use **Q** (or **q**) + **Enter** to quit manually.

If the refresh rate is zero, **bmon** pauses until it receives a keyboard response. The **refreshrate** setting in the configuration file specifies how many seconds to pause.

Refresh Options	Notes
refreshrate=5 (pause for 5 seconds and then run bmon again) refreshrate=0 (pause until valid keyboard response received)	The value must be zero or a number 5 or higher. Default value in the sample file monconfig.txt is 5.

Refresh Options	Notes
limitrefresh=0 (manually stop bmon using Q or q + Enter) limitrefresh=*n* (run bmon *n* times and then exit)	Maximum value is 2147483647. Default value in the sample file monconfig.txt is zero. In sessions where stdin would be expected locally but is missing because of a remote connection, bmon exits with the warning "Interactive input failure detected." Setting limitrefresh to 1 or higher suppresses this message.

Monitoring Data File Fragmentation

Over time in a busy database as records are created, updated, or deleted, data can become fragmented, lengthening times for file access and transaction response. This fragmentation differs from file system fragmentation on a hard disk because it occurs within the data file itself. As a developer or DBA, you may know when a file is likely to fragment from heavy use, but in some systems, you may be guessing.

Defragmenter is a tool that helps you solve this problem by finding data fragmentation and enabling you to correct it. Defragmentation rearranges records and rebuilds indexes in data files and removes unused space so that data can be efficiently accessed again. Defragmenting a file does not alter its data in any way, and records can be created, read, updated, or deleted while their files are being defragmented. You can use Defragmenter features during database engine execution with no need for down time or disruption of business operations in most cases.

Opened as a graphical tool in Zen Control Center, Defragmenter shows data files in use, including their number of reads and writes so that you can quickly find ones under heavy use. To add files or tables to the Watch List tab, select them and then drag-and-drop, click a button, or right-click and choose the command. If you know of Btrieve files in other locations, you can browse to them and add them to be watched as well.

Defragmenter also runs as the **dbdefrag** tool.

The following topics cover the use of Defragmenter:

- Deciding When to Defragment
- When Defragmenter Cannot Be Used
- Accessing Defragmenter
- Defragmenter GUI
    - Setting Defragmenter Preferences
    - Setting Automatic Refresh Interval
    - Arranging Tabs
- Defragmenter Tasks
- Command Line Interface Defragmenter

See Automatic Defragmentation for information about using Defragmenter in an unattended mode.

Deciding When to Defragment

Defragmenter helps you analyze your data files for statistics that may explain loss of performance. High statistics for file size and percentage fragmented, unused, and not in order can explain loss of database performance. By defragmenting a file or table, you can reduce all four of these numbers, which are explained in more detail under the Watch List topic. Transactions generally run more quickly against a newly compacted, reindexed file, restoring efficiency, capacity, and performance.

Every database is different, so it isn't possible to make recommendations that apply to everyone. The decision to defragment depends on your knowledge and experience with your own database and its applications. However, some general considerations are worth making:

- Performance remains constant in read-only databases, but as reads and writes occur over time, analysis of watched files will show rising values for statistics. Changes in database behavior may also become noticeable, such as queries and reports running more slowly.

- Bulk delete actions often greatly increase unused space in files, which can be corrected by defragmenting them.

- Reads and writes continue uninterrupted during defragmentation. However, for the following reasons you may want to consider low traffic times for your defragmentation strategy:

  - Defragmentation uses resources that normally are fully devoted to operations on data files, so the extra demands on the engine may cause small effects on performance in high traffic periods.

  - Defragmenting must very briefly lock each data file it runs against. In low traffic periods, this lock may not be noticeable, but in high traffic times, clients may wait slightly for responses.

If you do not see performance improvements after defragmenting, then the problem likely lies elsewhere and requires a different diagnosis and solution.

When Defragmenter Cannot Be Used

In some cases, Defragmenter cannot be used or will be canceled by database activities of higher priority.

- Defragmentation of files on clients is not currently supported. Files can be defragmented only on the local server where the tool runs.

- The disk space needed to defragment a file is displayed in the analysis results. Defragmenter requires free disk space equal to the size of the file, plus a small amount for defragmenting operations. Files undergoing high write activity may require more space.

- If free disk space is too low, then the engine denies attempts to start defragmentation and returns error code 126. If space becomes low on a disk volume while defragmentation is executing there, the engine cancels defragmentation of files on that volume and returns 126.

- If defragmentation starts but then the disk volume where defragmentation is executing becomes full, the engine cancels defragmentation activity on that volume and returns error code 18.

- Use other ZenCC tools to make changes to indexes and metadata, including:

  - Creating, modifying, or deleting indexes

  - Modifying a schema, such as creating, modifying, or deleting a column

  - Deleting databases, tables, or data files

  - Changing database, table, or file names

  - Setting or clearing owner names

  - Creating, modifying, or deleting bound databases

- Backup Agent operations cannot be performed on a file undergoing defragmentation. To perform defragmentation, you must first exit from Backup Agent.

- Continuous operations for data backup cannot be performed on a file undergoing defragmentation because their higher priority cause defragmentation to be canceled. To perform defragmentation, you must wait for these operations to finish.

- Defragmentation is not currently supported for a server engine in an environment where VSS has performed backup operations

- If you need to alter data file properties, such as page size, compression, or file version, use Tools > Rebuild instead of Defragmenter.

- Defragmenter stops and exits automatically if it finds corrupted or erroneous records. In these cases, use Rebuild to recover the data. Freshly rebuilt files do not need defragmenting.

- For DataExchange users: Defragmentation does not change the system data and key used by DataExchange. After defragmenting, you do not need to run the table synchronization and check tool **dxsynctables**.

Accessing Defragmenter

Defragmenter runs in three ways:

- In Zen Control Center, select **Tools** > **Defragmenter**. For details, see Defragmenter GUI.

- At a command prompt, run **dbdefrag**. Note that this differs from **defrag**, the Microsoft tool for defragmenting Windows file systems. For details, see Command Line Interface Defragmenter.

- The Automatic Defragmentation option can be turned on in the Performance Tuning configuration settings for a server engine. For details, see Automatic Defragmentation.

Defragmenter GUI

The following illustration shows the Defragmenter GUI. The table after the image describes the GUI objects. Click an area of the image for which you want more information.
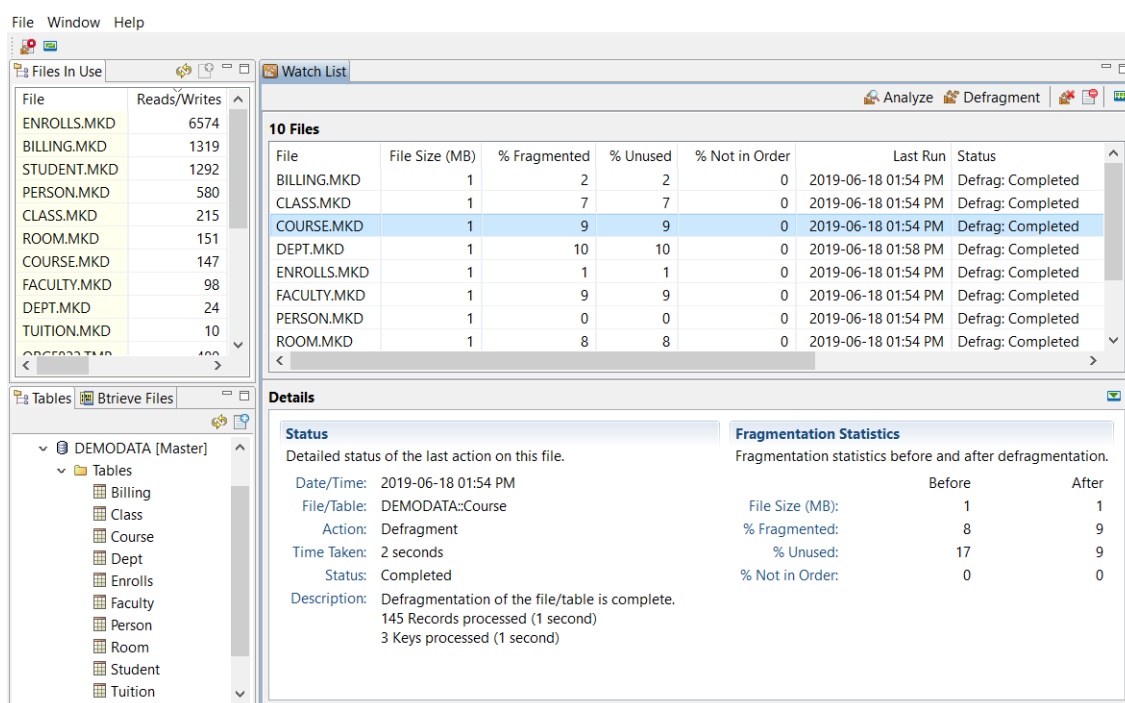
GUI Object	Description
File menu	Allows you to set the refresh rate for the Watch List tab, exit Defragmenter, or exit ZenCC.
Window menu	Allows you to return to the ZenCC window or to set preferences.
Help menu	Allows you to access documentation and logs and to check the Zen version.

Icon	Meaning
Cancel all defragmentation	Allows you to manually stop all analysis or defragmentation. Cancellation means that any file currently being analyzed will show no new statistics. Files being defragmented but not yet finished will be unchanged. Files awaiting analysis or defragmenting will not be done.
	In the upper left corner of the window, click this icon to cancel all active analysis and defragmentation operations. You do not need to select any files to use this option.
	Note: You can cancel all operations from the Defragmenter window even if they were started at a command prompt, and vice versa.
Set the automatic refresh rate	Opens a dialog to set how often the Watch List tab updates progress. The value is in seconds. The default is 5.
Refresh	Allows you to manually refresh the list of items in the Files In Use, Tables, or Btrieve Files tabs. Note that setting the automatic refresh rate affects only the Watch List tab.
Add to Watch List	Enters selected items in the Watch List tab. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.
Analyze file	Gathers and displays fragmentation statistics for items selected in the Watch List tab. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items. Analysis does not change the file being analyzed.
Defragment file	Starts defragmentation for items selected in the Watch List tab. Use Ctrl-click to add an item to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.
Cancel defragmentation	Allows you to selectively stop analysis or defragmentation. Canceling means that files being analyzed show no new statistics. Files being defragmented but not finished are unchanged.
	In the icons to the right above the Watch List, this icon becomes available when files in the Watch List are undergoing analysis or defragmentation. Selecting one or more files and clicking this icon cancels the operation for those files. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.
Remove file	Removes an item from the Watch List tab. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

Icon	Meaning
Select columns to display	Opens a dialog of check boxes to customize the columns in the Watch List tab. The default is for all columns to be displayed.
Show or Hide details	Toggles the display of the Details pane.

Defragmenter Tab Views

Defragmenter presents database tables, files, and other objects in tabs in the Defragmenter window.

- Files In Use
- Tables
- Btrieve Files
- Watch List

The tabs in the window can be rearranged. See Arranging Tabs.

Files In Use

The Files In Use tab lists items currently or recently opened by the Zen engine. You can add files shown in this tab to the Watch List tab to monitor, analyze, and defragment. It is best practice to select only files involved in routine database execution, since they are most likely to contribute to performance issues.

In the Files In Use tab, you can sort the list by clicking the Reads/Writes column heading. Sorting the list helps to identify which files are in heaviest use and possibly more fragmented.

Tables

The Tables tab resembles Zen Explorer, showing nodes for the server where Defragmenter is running. You can add any file shown in this tab to the Watch List tab.

Btrieve Files

The Btrieve Files tab is a file explorer for drives and directories on the file system. In addition to data files, this tab also lists items not involved in routine operations, such as dbnames.cfg and .ddf files. It is possible to add these files to the Watch List tab to analyze and defragment, but unlike data files, attempting to change them while they are in active use returns errors. These metadata objects are at low risk of fragmentation because they generally do not change in a production

environment. In the rare case of their needing defragmenting, it can be done during down-time maintenance.

Watch List

Items in the Watch List tab display statistics after an Analyze or Defragment action has been applied.

Statistic	Measures	Interpretation
File size (MB)	File size in megabytes	The longer a file has been in use, the greater the amount of unused space it may contain, increasing its size. Smaller file sizes typically take less time to defragment, unless they have a large number of indexes. For Cloud Server users, the larger size of fragmented files may be a concern, so defragmentation may help to stay within your license limit.
% Fragmented	Percentage of data residing in blocks separated by unused space	Transactions over time can result in pages containing no data. The more these pages are intermixed with actual data pages, the greater the fragmentation. A lower percentage indicates fewer, larger data blocks stored closer together, allowing more rapid reads and writes. **Note:** This statistic is not supported for Btrieve 6.x and 7.x files. You can analyze and successfully defragment these earlier file versions, but no measurements can be shown for these statistics.
% Unused	Percentage of unused space	Unused space is often created when a file is modified by insert, update and delete operations. The lower this percentage, the more compact the file, allowing more rapid reads and writes. **Note:** This statistic is not supported for Btrieve 6.x and 7.x files. You can analyze and successfully defragment these earlier file versions, but no measurements can be shown for these statistics.

Statistic	Measures	Interpretation
% Not in Order	Percentage of records not stored sequentially	Inserts over time result in a widening mismatch between the logical order of pages and their physical locations, increasing the time needed to locate data.
		A lower percentage generally gives higher performance for actions such as table scans on large files.
		Note: In some files, the lowest possible percentage not in order may be higher than zero. Further defragmentation does not decrease this statistic because records in the file are already stored as efficiently as possible.
Last Run	Date and time of last action	Action can be Analyze or Defragment.
Status	Report from last action	Blank for newly added item. Typically notice of analysis or defragmentation completed.
Table	Logical location	Database and table or file name
Path	Physical location	File system path

Details

Under the Watch List tab, a Details pane summarizes the statistics in the Watch List columns, but with some additional information:

- Time taken to defragment the item

- Before and after comparisons of file size and percentage fragmented, unused, and not in order

Selecting an item in the Watch List tab displays its details. If you select more than one item, details are shown for the first one highlighted.

Setting Defragmenter Preferences

You can set preferences for Defragmenter from either the tool itself or from Zen Control Center. In either GUI, select **Window** > **Preferences** > **Zen** > **Defragmenter** to open the Defragmenter tab of the Preferences dialog box.

The following table shows the types of preferences you can set and what they do.

Preference	When selected...
Remember window layout.	Saves the current arrangement of Defragmenter tabs when you exit the tool, as well as the overall window size, height, and position.
Don't warn about incompatible actions during multi-select operations.	Stops the dialog box from showing when actions will be unevenly applied to items of different types.
Don't warn about canceling defragmentation again.	Stops the dialog box from showing a warning when you cancel an analysis or defragmentation operation. If you selected the check box in the dialog to confirm canceling, this box also becomes checked. To make the confirmation dialog appear again, clear this check box.
Don't warn about canceling all defragmentation again.	Stops the dialog box from showing a warning when you cancel all analysis and defragmentation operations. If you selected the check box in the dialog to confirm canceling, this box also becomes checked. To make the confirmation dialog appear again, clear this check box.

Setting Automatic Refresh Interval

Information in Defragmenter can be refreshed automatically at a configured interval. The default is to refresh every 5 seconds. Be aware that refreshing too many files too often may slow performance.

Arranging Tabs

The Defragmenter tabs can be rearranged and separated for your convenience. To move a tab, drag and drop the tab label where you want the tab to be. If the setting to remember window layout is selected in the Defragmenter preferences, your tab arrangement will be kept.

Defragmenter Tasks

This topic provides steps for using features for various tasks in the Defragmenter GUI.

To set automatic refresh interval

The automatic refresh rate applies to the statistics in the Watch List tab. Refreshing of lists in the other tabs is done manually by clicking the Refresh icon or right-clicking and selecting Refresh.

1. Click **File** > **Set Automatic Refresh Rate** or click the **Set the Automatic Refresh Rate** icon ⬚ to open the refresh rate dialog box.

2. In the dialog box, enter the number of seconds between each refresh. The value must be an integer greater than 1.

3. Click **OK**.

To add a file or table to the Watch List tab

Adding a file or table to the Watch List tab can be done in a number of ways:

- In Defragmenter, select one or more items in the Files In Use, Tables, or Btrieve Files tabs and then click the Add to Watch List icon 📑 at upper right in the tab. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

- In Defragmenter, right-click an item in the Files In Use, Tables, or Btrieve Files tabs and select Add to Watch List.

- In Defragmenter, drag and drop an item from any tab to the Watch List tab.

Items added to the Watch List tab appear at the end, after the items already there.

To analyze a file or table

Analysis reports statistics that determine which items may need defragmenting.

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:

    - In the Watch List tab at top right, click 🔍 **Analyze**.
    - Right-click the selected items and click **Analyze File**.

The Watch List tab reports analysis results by updating its columns. For the first item selected, the Details tab provides individual status, including the free disk space needed to defragment the file.

**To defragment a file or table**

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:

- In the Watch List tab at top right, click 👷 **Defragment**.

- Right-click the selected items and click **Defragment File**.

The Watch List tab reports defragmentation results by updating its columns. For each item selected, the Details tab below provides individual status. For multiple selected items, the first one selected is shown.

To cancel analysis or defragmentation

During analysis or defragmentation, select a file in the Watch List and click the Cancel Defragmentation icon 👷 on the right. In the dialog that appears, click Yes to confirm. The Defragmenter operation continues until you click Yes.

Alternatively, you can click the Cancel All Defragmentations icon 👷 on the left. This icon stops all analysis and defragmentation activity, while the one on the right is for selected files in the Watch List that are undergoing analysis or defragmentation.

After cancellation, status for some items in the Watch List tab may be Analysis: Completed or Defrag: Completed, with new statistics, while others display Analysis: Canceled or Defrag: Canceled and blank statistics. For a canceled item, the Details tab description gives the time that passed until the moment of cancellation.

Successful cancellation means that analysis was stopped or that fragmentation in the file or table is unchanged.

Note: You can cancel all operations from the Defragmenter window even if they were started at a command prompt, and vice versa.

To remove a file or table from the Watch List tab

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:

   - In the Watch List tab at top right, click the **Remove File** icon 👷.

   - Right-click the selected items and click **Remove File**.

To select columns to display

You can set every column in the Watch List tab to be shown or hidden.

1. In the Watch List tab at top right, click the Select Columns to Display icon 🖽.
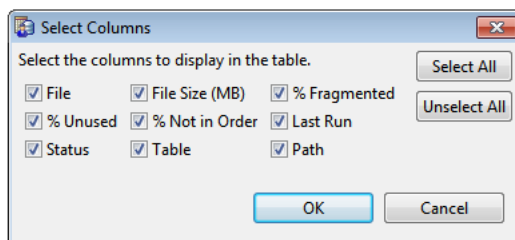
2. By default, all choices are selected. Make any needed changes and click **OK** or click **Cancel**.

To hide or show details of analysis or defragmentation

You can toggle to hide or show the Details tab by clicking the Click to Hide Details icon ▼.

Command Line Interface Defragmenter

The CLI tool **dbdefrag** provides the same functions as its GUI counterpart, except that there is no watch list. Defragmentation actions can be started, checked for status, or canceled from either interface.

The following table gives command line options.

Command	Description
dbdefrag *files*	Start defragmentation on one or more files. You may use asterisks as wildcards. Displays the number of records and keys processed. You can use Ctrl-C to cancel. In addition to path names, files can also be specified as a URI using the syntax brtv://*user@host/dbname?parameters*, as described under Database URIs.
dbdefrag -background *files*	Start defragmentation on one or more files as a background process. You may use asterisks as wildcards.
dbdefrag -cancel *files*	Cancel the currently running analysis or defragmentation for one or more files. You may use asterisks as wildcards. Must be executed at a separate prompt unless the defragmentation is running in the background.
dbdefrag -cancelall	Cancel all active analysis and defragmentation operations. Must be executed at a separate prompt unless the defragmentation is running in the background.
	Note: Note: You can cancel all operations from the Defragmenter command prompt even if they were started in the Defragmenter GUI window, and vice versa.

Command	Description
`dbdefrag -status` *files*	Show defragmentation status for one or more files. You may use asterisks as wildcards.
	Displays the following for the most recently completed defragmentation:
	• Status: Complete
	• Started: *yyyy-mm-dd hh:mm:ss*
	• Finished: *yyyy-mm-dd hh:mm:ss*
	• Time taken: *n*h:*nn*m:*nn*s
	Displays the following for a defragmentation currently running:
	• Defragmentation status: In Progress
	• *n* out of *n* records processed (*n*%)
	• *n* out of *n* keys processed (*n*%)
	Displays the following for a canceled defragmentation:
	• Status: Defragmentation Canceled
	• Time taken: *n*h:*nn*m:*nn*s
	File status remains available until the engine is restarted. After an engine restart, you can find status information in the Zen event log (zen.log).
`dbdefrag -analyze` *files*	Show fragmentation statistics for one or more files. You may use asterisks as wildcards. Returns the following:
	• % Fragmented
	• % Unused
	• % Not in Order
	• File Size (MB)
	• Estimated disk space needed for defragmenting (MB)
	Analysis does not change the file being analyzed.
`dbdefrag -ignore`	Ignore errors during file processing. You can add this option to any command. It is useful when you are analyzing or defragmenting multiple files. Without the -ignore option, if one of the files being analyzed or defragmented returns an error, then the error is logged and analysis and defragmentation operations stop for remaining files. If you use the -ignore option, then the error is logged and analysis and defragmentation continue on the rest of the files.
`dbdefrag -help`	Display the list of command options.

Using dbdefrag

The following example shows a series of steps using `dbdefrag` on a large .mkd file.

1. **Analyzing**

```
C:\ProgramData\Actian\Zen\Examples>dbdefrag -analyze mybtrievefile.mkd
  Fragmented: 9%
      Unused: 4%
Not in Order: 100%
   File Size: 713 MB
```

2. **Defragmenting**

```
C:\ProgramData\Actian\Zen\Examples>dbdefrag mybtrievefile.mkd
1125 out of 1195242 records processed (0%)
87021 out of 1195242 records processed (7%)
170910 out of 1195242 records processed (14%)
255393 out of 1195242 records processed (21%)
339805 out of 1195242 records processed (28%)
404202 out of 1195242 records processed (33%)
487928 out of 1195242 records processed (40%)
572585 out of 1195242 records processed (47%)
655802 out of 1195242 records processed (54%)
716472 out of 1195242 records processed (59%)
800406 out of 1195242 records processed (66%)
883729 out of 1195242 records processed (73%)
947475 out of 1195242 records processed (79%)
1032061 out of 1195242 records processed (86%)
1116015 out of 1195242 records processed (93%)
1185831 out of 1195242 records processed (99%)
1195242 out of 1195242 records processed (100%)
1 out of 2 keys processed (50%)
2 out of 2 keys processed (100%)
Defragmentation complete.
Time taken: 0h:24m:57s
```

3. **Checking Status from Another Prompt or When Running in Background**

```
C:\ProgramData\Actian\Zen\Examples>dbdefrag -status mybtrievefile.mkd
Defragmentation status: In Progress
1053373 out of 1195242 records processed (88%)
0 out of 2 keys processed (0%)
```

4. **Checking Status After Defragmenting**

```
C:\ProgramData\Actian\Zen\Examples>dbdefrag -status mybtrievefile.mkd
    Status: Complete
   Started: 2014-08-08 11:08:28
  Finished: 2014-08-08 11:33:25
Time taken: 0h:24m:57s
```

5. **Analyzing Again**

```
C:\ProgramData\Actian\Zen\Examples>dbdefrag -analyze mybtrievefile.mkd
  Fragmented: 0%
      Unused: 0%
Not in Order: 27%
   File Size: 682 MB
```

Monitoring Performance Counters

A Zen server on Windows provides performance counters for use with the Windows Performance Monitor. (The Zen performance counters are supported only on Windows Vista or later operating systems.) The performance counters measure state or activity of the database engine, which allows you to analyze performance of your application. Windows Performance Monitor requests the current value of the performance counters at specified time intervals.

Zen provides data only for display by the Performance Monitor and cannot modify the counter properties. Performance Monitor controls the following:

- The display of data is in three formats: line graph (default), histogram, and as a text report.

- The display fields are labeled Last, Average, Minimum, and Maximum.

- The scaling of values for display. Zen provides a default scale for each counter. Performance Monitor allows you to change the scaling of individual counters. See To Change a Counter Scale.

The counter values reflect all calls into the database engine regardless of their source. That is, the MicroKernel Engine, Relational Engine, native Btrieve applications, utilities, and so forth, all contribute to the counter values. Counter values are collected for all files. Counters on a per-file basis are not currently supported.

Note that the use of performance counters is an advanced feature intended primarily for application developers and other technical staff. Refer to the Microsoft documentation for details about the Windows Performance Monitor and on the use of counters in general.

Registration During Installation

By default, the Zen installation registers the Zen performance counters with Performance Monitor. The counters are available for use after installation completes.

Note that response to customer needs may result in additional Zen collector sets or counters being installed that are not discussed in this chapter. If so, refer to the description of the collector set or counter provided in Windows Performance Monitor. See Add Sets or Individual Counters To Monitor.

Data Collector Sets

A data collector set organizes multiple counters into a single component that can be used to review or log performance. Zen provides the following data collector sets:

- Zen MicroKernel Btrieve Operations

- Zen MicroKernel Cache

- Zen MicroKernel I/O

- Zen MicroKernel Locks and Waits

- Zen MicroKernel Transactions

- Zen Page Server Activity

Zen MicroKernel Btrieve Operations

These counters are useful for characterizing the behavior of client applications in terms of the Btrieve API. The counters report the types of operations being processed by the database engine at a given point in time.

See also Btrieve API Operations in *Btrieve API Guide*.

Counter	Description	Typical Use
Btrieve Close Operations per Second	The number of Btrieve Close operations per Second	To provide insight into client application behavior. As a first step in troubleshooting issues, you may find it helpful to analyze behavior in terms of Btrieve operations.
Btrieve Get/Step Operations per Second	The number of Btrieve Get and Step operations per second. See Btrieve API Operations in *Btrieve API Guide* for the various Get and Step operations.	
Btrieve Open Operations per Second	The number of Btrieve Open operations per Second	
Btrieve Records Deleted per Second	The number of Btrieve records deleted per second	
Btrieve Records Inserted per Second	The number of Btrieve records inserted per second	
Btrieve Records Updated per Second	The number of Btrieve records updated per second	
Change Operations per Second	The number of Btrieve operations that modify the data files per second	
Operations per Second	The number of Btrieve operations executed per second	

Zen MicroKernel Cache

The database engine uses a two-level memory cache system to increase performance of data operations. The two caches are called Level 1 (L1) Cache and Level 2 (L2) Cache.

The more frequently the engine must read a page from disk to complete a user request, the lower the performance. These counters can be used collectively to view how successfully the database engine avoids disk reads and to determine if any changes need to be made to the cache size settings.

The best performance occurs when all data is stored in the L1 Cache. Because of the sizes of data files, however, the L1 cache cannot always hold all of the data pages, so the database engine also uses a second level cache. Pages in the L2 Cache are stored in a compressed format, allowing for more pages to fit in memory. Consequently, it is lower performing that the L1 Cache, but higher performing than the database engine reading the pages from disk.

See also To calculate the ideal size of the database memory cache.

Counter	Description	Typical Use
Cache Hit Ratio	The percentage of recent cache accesses that resulted in a hit from either the Level 1 or Level 2 Cache.	
L1 Cache Discards/sec	The number of pages discarded from the Level 1 Cache per second in order to make room for a new page. Only occurs when the Level 1 Cache is full. Does not include pages moved from the Level 1 to the Level 2 Cache.	
Level 1 Cache Dirty Percentage	The percentage of the Level 1 Cache in use that contains dirty pages	To help determine if heavily accessed pages are continuously being forced out of the cache, which may adversely affect performance. Dirty pages, ones with changes that have not been written to disk, may only reside in the L1 Cache. Under heavy write loads, the L1 Cache may predominately contain dirty pages. This forces pages out of the L1 Cache and into the L2 Cache, if configured, or out of the L1 Cache entirely. The database engine writes dirty pages to disk at scheduled intervals or when the L1 Cache gets close to full. Frequently writing pages to disk may also adversely affect performance. It may benefit performance to adjust the L1 Cache size so that the percentage of dirty pages is not always high. See also Cache Allocation Size.

Counter	Description	Typical Use
Level 1 Cache Hits per Second	The number of Level 1 Cache hits per second	To help determine how successfully the database engine finds requested pages in L1 Cache. A higher rate of hits-to-misses indicates that the engine is finding pages in L1 Cache rather than needing to access the L2 Cache or physical storage.
Level 1 Cache Hit Ratio	The percentage of recent Level 1 Cache accesses that resulted in a hit.	
Level 1 Cache Misses per Second	The number of Level 1 Cache misses per second	
Level 1 Cache Usage Percent	The percentage of Level 1 Cache currently in use	To aid in adjusting the size of the L1 Cache to fit your application(s). For example, applications that use small or predominately read-only data files may not fill up the L1 Cache as configured by default. The unused memory is not available to the operating system or to other applications. You can change the L1 Cache size to release the memory back to the operating system. Conversely, if you want to have an entire database in memory, you can monitor this value to know when the setting is as desired.
Level 2 Cache Hits per Second	The number of Level 2 Cache hits per second	To help determine how successfully the database engine finds requested pages in L2 Cache. A higher rate of hits-to-misses indicates that the engine is finding pages in L2 Cache rather than needing to access physical storage.
Level 2 Cache Hit Ratio	The percentage of recent Level 2 Cache accesses that resulted in a hit.	
Level 2 Cache Misses per Second	The number of Level 2 Cache misses per second	

Counter	Description	Typical Use
Level 2 Cache Size	The current size of the Level 2 Cache in bytes	To help determine the size of the optional L2 Cache.
Level 2 Cache Usage	The amount of the Level 2 Cache currently in use in bytes	The L2 Cache is one component of the setting for Max MicroKernel Memory Usage. That setting specifies the maximum proportion of total physical memory that the database engine is allowed to consume, which includes L1 Cache, L2 Cache, and all miscellaneous memory usage by the database engine. If the setting for Max MicroKernel Memory Usage is non-zero, the L2 Cache sizes itself to stay within the memory limit of the setting. The L2 Cache monitors memory consumption of the system and resizes itself as needed. The memory used by the L2 Cache may also be swapped out by the operating system.
Level 2 Cache Size Relative to Memory	Level 2 Cache size presented as a percentage of total system memory	To show what percentage of the total system memory the L2 Cache is using.
Level 2 Cache Usage Percent	The percentage of Level 2 Cache currently in use	To show what percentage of the Level 2 Cache is currently being used.

Zen MicroKernel I/O

The counters in this set are useful for understanding the interactions of the database engine and data read and written to physical storage. The pages reported by the counters are data file pages. These counters do not report data for pages in files used for archival logging or for transaction logging.

See also Pages in *Zen Programmer's Guide*.

Counter	Description	Typical Use
Pages Read per Second	The number of pages read from disk per second	To determine the interaction of the database engine and data read and written to physical storage.
Pages Written per Second	The number of pages written to disk per second	

Zen MicroKernel Locks and Waits

Client requests may be delayed by waiting for a resource to become available. These counters give insight into the types of database resources on which a client request may have to wait until the resource is available. As such, these counters may provide insight into the behavior of the database engine when multiple clients access it. A value close to or equal to the number of clients may indicate collisions for the same resources. Any corrective actions that can be done to alleviate these collisions may improve responsiveness.

The counters Waits on Page Buffers and Waits on Page Reads are waits on a global resource. All of the other counters in this grouping apply to multiple clients, but each client may be waiting on resources that differ from client to client.

See also Data Integrity and Supporting Multiple Clients, both in *Zen Programmer's Guide*.

Counter	Description	Typical Use
Client Record Locks	The number of records explicitly locked by clients	To provide insight into the work load of client applications.
Waits on Active Reader Lock	The number of clients waiting on the Active Reader Lock. Multiple clients may hold the Active Reader Lock at the same time; however, the Active Reader Lock and the Active Writer Lock are exclusive. Consequently, a single client that holds the Active Reader Lock prevents any client from obtaining the Active Writer Lock. A single client that holds the Active Writer Lock prevents multiple clients from obtaining the Active Reader Lock. Each file has its own reader (and writer) lock. See also Waits on Active Writer Lock counter.	
Waits on Active Writer Lock	The number of clients waiting on the Active Writer Lock. Only one client may hold the Active Writer Lock for a file at a time. Each file has its own writer (and reader) lock. See also Waits on Active Reader Lock counter.	
Waits on File Locks	The number of clients currently waiting on a file lock	

Counter	Description	Typical Use
Waits on Page Buffers	The number of clients waiting on a page buffer to become available. If a page is not available to service a request, the request blocks until the MicroKernel is able to make a page available.	To indicate whether or not the database engine has a page buffer available in the cache. Use this value along with the memory cache counters to decide if the caches are sized appropriately for the work load. Increasing the cache size will increase the total number of available pages, which can reduce the waits on page buffers. Three things may cause this value to spike when pages are not in cache: • A data file was recently opened. • First time or infrequent access of a data page. • The caches may be too small to contain all of the pages frequently accessed and modified. The spike for the first two items cannot be avoided because of accessing a file for the first time. The third item can be avoided by using a larger cache. If the caches are full and the cache misses are high, it is possible that the caches may be too small to contain all the pages frequently accessed and modified. See also Zen MicroKernel Cache.
Waits on Page Locks	The number of clients currently waiting on page locks	To provide insight into the work load of client applications.
Waits on Page Reads	The number of clients waiting to read a page from disk. If a client is already in the process of reading the page, other clients must wait for the in-progress read to complete.	To help determine the number of clients trying to read the same page of the same file at the same time.
Waits on Record Locks	The number of clients currently waiting on record locks	To provide insight into the work load of client applications.

Zen MicroKernel Transactions

These counters are useful for understanding the behavior of client applications in terms of transactions. For example, a few long-lasting transactions that involve many changes cause a different behavior than many short-lived transactions.

See also Begin Transaction (19 or 1019), End Transaction (20), and Abort Transaction (21) in *Btrieve API Guide*, and MicroKernel Engine Fundamentals in *Zen Programmer's Guide*.

Counter	Description	Typical Use
System Transactions in Progress	The number of system transactions in progress. A system transaction is a special type of transaction that prepares data file changes then persists the changes to the file.	To help determine if system transactions are occurring too frequently or not often enough. The database engine writes changes to the data files during a system transaction. The frequency at which a system transaction occurs is determined by two server data integrity configuration properties – Initiation Time Limit and Operation Bundle Limit. The server also automatically adjusts the frequency when the amount of free space in the L1 Cache is low. In general, running the system transaction too frequently or not often enough adversely affects performance. Typically, you will notice that the number of page writes per second may increase, the number of Btrieve operations that modify records may decrease, and the number of clients waiting on the Active Writer Lock may increase. It may take experimentation to determine an ideal interval for a particular work load.
Transaction Commits per Second	The number of commits executed per second	To determine the number of application transaction commits. See also End Transaction (20) in *Btrieve API Guide*.

Zen Page Server Activity

The counters in this set provide statistics on the page server.

Counter	Description
Page Server Pass-Thru Operations/sec	The number of pass-through operations received by the page server per second from cache engines.
Page Server Page Requests/sec	The number of requests for a data file page received by the page server per second from cache engines.

| Counter	Description
Page Server Invalid Page List Requests/sec	The number of requests for an invalid page list received per second from cache engines.
Page Server Invalid Page List Piggy-backs/ sec	The number of invalid page lists piggy-backed by the page server per second on responses to other cache engine requests.
Page Server Invalid Page List Length	The current length of the page server invalid page list.
Page Server Invalid Page List Adds/sec	The number of committed pages added to the page server invalid page list per second, counting duplicates.
Page Server Invalid Page List Removals/ sec	The number of entries removed from the page server invalid page list per second.
Page Server Level 1 Cache Hits/sec	The number of Level 1 Cache hits by the page server per second.
Page Server Level 1 Cache Misses/sec	The number of Level 1 Cache misses by the page server per second.
Page Server Level 1 Cache Hit Ratio	The percentage of recent Level 1 Cache accesses by the page server that resulted in a hit.
Page Server Level 2 Cache Hits/sec	The number of Level 2 Cache hits by the page server per second.
Page Server Level 2 Cache Misses/sec	The number of Level 2 Cache misses by the page server per second.
Page Server Level 2 Cache Hit Ratio	The percentage of recent Level 2 Cache accesses by the page server that resulted in a hit.
Page Server Cache Hit Ratio	The percentage of recent cache accesses by the page server that resulted in a hit from either the Level 1 or Level 2 Cache.
Page Server Level 1 Cache Usage Percent	The percentage of the Level 1 Cache currently in use only by the page server.
Page Server Level 2 Cache Usage Percent	The percentage of the Level 2 Cache currently in use only by the page server.

Zen Cache Engine Activity

The counters in this set provide statistics on Zen cache engines.

Counter	Description
Cache Engine Pass-Through Ops/sec	The number of pass-through operations requested by the cache engine per second.
Cache Engine Local Ops/sec	The number of local operations performed by the cache engine per second.
Cache Engine Page Requests/sec	The number of data file pages requested by the cache engine per second.
Cache Engine Page Invalidations/sec	The number of pages invalidated by this cache engine per second.
Cache Engine FCR Invalidations/sec	The number of FCRs invalidated by this cache engine per second.
Cache Engine Level 1 Cache Usage Percent	The percentage of the Level 1 Cache currently in use only by the cache engine.
Cache Engine Level 1 Cache Hits/sec	The number of Level 1 Cache hits by the cache engine per second.
Cache Engine Level 1 Cache Misses/sec	The number of Level 1 Cache misses by the cache engine per second.
Cache Engine Level 1 Cache Hit Ratio	The percentage of recent Level 1 Cache accesses by the cache engine server that resulted in a hit.
Cache Engine Level 2 Cache Usage Percent	The percentage of the Level 2 Cache currently in use only by the cache engine.
Cache Engine Level 2 Cache Hits/sec	The number of Level 2 Cache hits by the cache engine per second.
Cache Engine Level 2 Cache Misses/sec	The number of Level 2 Cache misses by the cache engine per second.
Cache Engine Level 2 Cache Hit Ratio	The percentage of recent Level 2 Cache accesses by the cache engine that resulted in a hit.
Cache Engine Cache Hit Ratio	The percentage of recent cache accesses by the cache engine server that resulted in a hit from either the Level 1 or Level 2 Cache.

Using Windows Performance Monitor

This topic provides some rudimentary instructions on using Windows Performance Monitor to get started with the Zen performance counters. See the Microsoft document for complete details on using Windows Performance Monitor.

The following steps assume that the Zen performance counters have been registered with Windows Performance Monitor by the Zen installation.

Display Zen Data Collector Sets

1. Start Windows Performance Monitor. The steps vary depending on the operating system, but generally the tool can be started from **Control Panel** > **Administrative Tools**. You may also try the command **perfmon** in the Run window (**Start** > **Run**).

2. In the tree on the left, click Performance Monitor, then click the plus sign on the right.
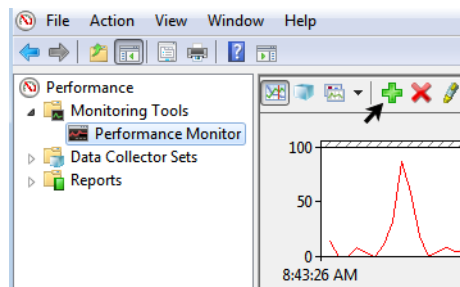
3. In the Available Counters group, scroll to the Zen data collector sets.
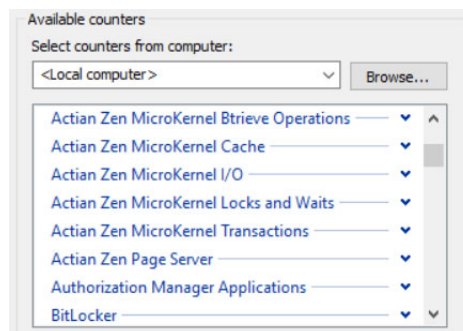
Add Sets or Individual Counters To Monitor

1. Do one of the following:

- To add an entire set, click the desired set in the Available Counters group.
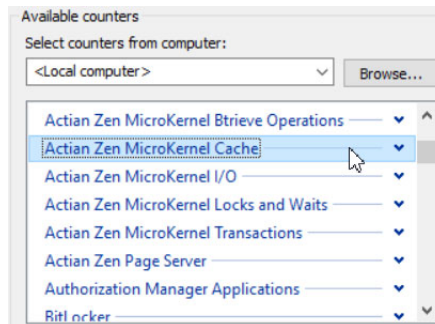
- To add individual counters, expand the desired set and click the desired counters.
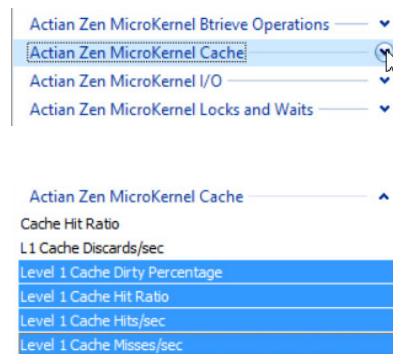
To view a description of the counter, ensure that the **Show description** option is selected.
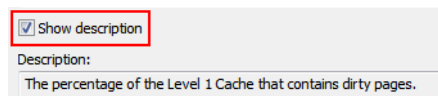
2. Click **Add**, then **OK**.

To Change a Counter Scale

1. After a counter has been added to the list in the bottom of the main window, you can right-click it and select **Properties**.

2. On the Data tab, use the Scale list to select a value.
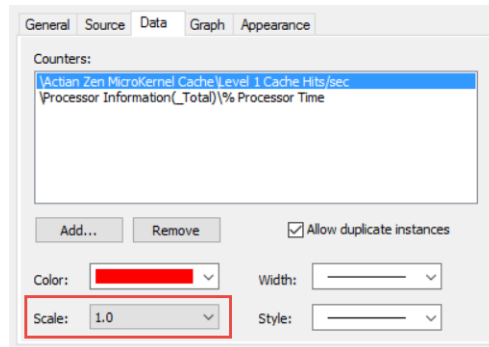
You may need to adjust the Scale to display two or more counters on the same graph that have vastly different ranges. The counter value is multiplied by the Scale value before the data is graphed. For example, assume that one counter outputs values of 53 and 99, and a second counter outputs 578 and 784. You may want to set the Scale for the first counter to 10 so that its output is 530 and 990. This lets you look at the data from both counters more comparably (530, 990, 578, and 784).

3. Click **OK**.

    Note that the scale on the display changes from its original value ("1" in this example) to the new value ("10" in this example):
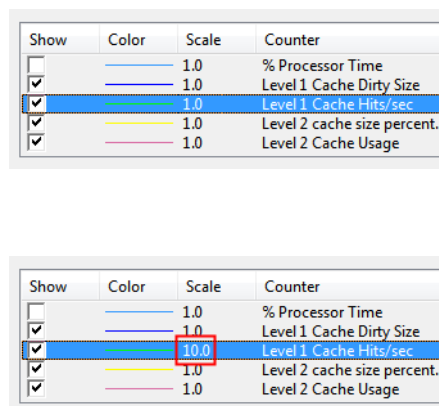
4. On the Graph tab, set the desired values for **Maximum** and **Minimum** of the Vertical scale.

    You may want to change the vertical scale if what is being graphed is very small (or very large) so you can easily see when the values change. For example, if the counter values are always under 20, you may want to change the vertical scale to be Maximum 20 and Minimum 0.

5. Click **OK**.

Monitoring License Usage

The Zen products use different license models depending on the product. Zen Enterprise Server and Workgroup are licensed under a concurrent user count license. Cloud Server is licensed under a capacity-based license.

User Count License	The user count license model works well for traditional client-server applications in which many users or devices constantly add, update, and delete records from distinct individual desktops. Each product key specifies a licensed user count. A user count allows the specified number of concurrent connections to the Zen database engine. Users are counted by network address.
Capacity-based License	A capacity-based model shifts the emphasis from how many users to how much work the database server processes. The model is based on capacity to accommodate license enforcement in service bureau, software-as-a-service, or other multiplexed environments. For example, each instance of Cloud Server has capacity limits on both the number of sessions in use and the data in use.

Much of the monitoring pertaining to license usage involves user count, number of sessions, or data in use. For example, you may want to determine the current value, increase the current value, or better determine your capacity needs.

The four primary utilities used to monitor license usage are Monitor (see Monitoring Resource Usage), Capacity Usage Viewer, License Administrator, and Notification Viewer.

Capacity Usage Viewer

ZenCC provides Capacity Usage Viewer to monitor concurrent sessions and data usage for all database engines. This is especially useful when you are considering migrating from Zen Enterprise Server to Zen Cloud Server, because of the difference in licensing.

Capacity Usage Viewer includes two graphs, one for the number of concurrent sessions and one for the amount of data. Each graph includes a *usage level bar*, a heavy horizontal line across the graph, to help you determine what volume of usage is normal and what volume is uncommon for your business. The Capacity Usage Viewer also displays peak usage statistics.

The graphs use the peak values that are recorded each day. For any day on which the engine is not used, they use a value of zero. They require a minimum of two days' data to be generated. Otherwise, Capacity Usage Viewer displays an error message.

This section contains the following topics:

- To access Capacity Usage Viewer

- Capacity Usage Viewer GUI

- Zooming

To access Capacity Usage Viewer

In ZenCC, right-click the engine you want to examine and select **Capacity Usage Viewer**.

Capacity Usage Viewer GUI

The following image shows the Capacity Usage Viewer. The table below the image describes its features. Click an area of the image for which you want more information.
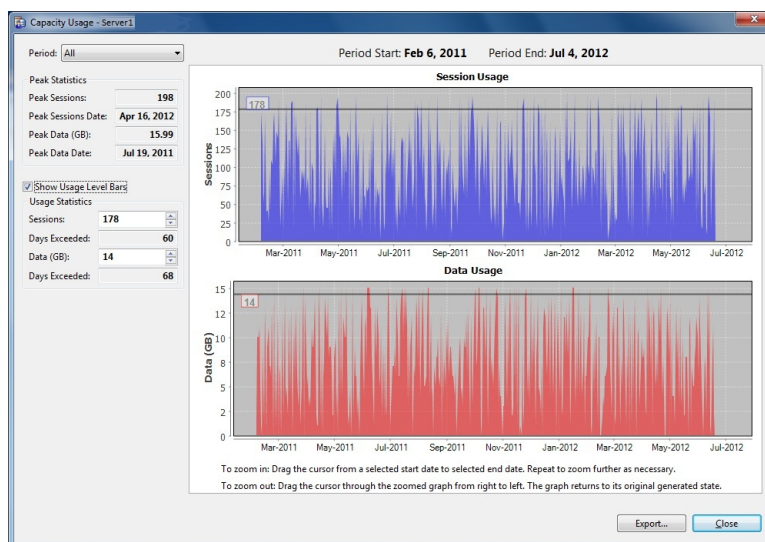
GUI Object	Description
Title Bar	Identifies the engine you selected.
Statistical Indicators	Displays meaningful statistics and enables you to select what you want to display in the graphs. See Statistical Indicators.
Time Designation	Displays starting and ending dates of the period for the data displayed in the graphs.
Session Usage Graph	Displays, graphically, the number of sessions that occurred concurrently during the selected time period.

GUI Object	Description
Data Usage Graph	Displays, graphically, the amount of data used during the selected time period.
Usage Level Bars	Enables you to determine how often your usage exceeds a selected level. When selected, they appear across each graph at a default level that is 90% of peak usage. The number at the left of each usage level bar identifies its level (amount of data or number of sessions). You can move the usage level bar to whatever level you need, either by using the spin boxes or by dragging with the cursor. The two usage level bars are independent of each other.
Zoom Instructions	Describes the general procedures for zooming in and out of a graph. For detailed procedures, see Zooming.
Export Button	Enables you to export the data to a .csv file, if you find it useful to save the data for additional analysis. The **Export** button opens a **Browse for Folder** dialog, where you can select a location for data storage.

Statistical Indicators

The following image shows Statistical Indicators information. The table below the image describes its features. Click an area of the image for which you want more information.

GUI Object	Description
Period	Enables you to select the time period for the data you want the graphs to display. When the window opens, the graphs display, by default, the data from the period that was selected when the window was last closed. You can select a different time period: • All • Last week • Last 30 days • Last 90 days • Last 180 days You can also select a time period by zooming the graphs. When you zoom a graph, the **Period** drop-down menu displays **Custom** as the selected time period.
Peak Statistics Group Box	Contains fields that display statistics for maximum use of data and maximum number of concurrent sessions during the time period displayed in the graphs.
Peak Sessions	Displays the greatest number of concurrent sessions that occurred during the time period displayed in the **Session Usage** graph.
Peak Sessions Date	Displays the date on which the greatest number of concurrent sessions occurred. If that number of sessions occurred more on more than one day, the most recent date is displayed.
Peak Data (GB)	Displays the maximum amount of data, in gigabytes, used at one time during the time period displayed in the **Data Usage** graph.
Peak Data Date	Displays the date on which the maximum amount of data was used. If that amount of data was used on more than one day, the most recent date is displayed.
Show Usage Level Bars	Displays or hides usage level bars across the graphs, depending on whether it is checked or not.
Usage Statistics Group Box	Contains spin boxes for moving the usage level bars up and down and fields that display the statistics that result from moving the usage level bars.
Sessions	Sets the value at which to position the usage level bar in the **Session Usage** graph.
Days Exceeded	Displays the number of days on which the number of concurrent session in use was greater than the level at which the usage level bar is set.

GUI Object	Description
Data (GB)	Sets the value at which to position the usage level bar in the **Data Usage** graph.
Days Exceeded	Displays the number of days on which the amount of data in use was greater than the level at which the usage level bar is set.

Zooming

If you need to view a particular time period besides the selections provided in the **Period** pull-down menu, you can select and zoom a segment of a graph. When you zoom one graph, the other zooms simultaneously. The two graphs are always set to the same period.

To zoom a graph

1. Place the cursor at the start of the period you want to display.

2. With your left mouse button pressed, drag the cursor to the end of the period you want to display. In the graph, a rectangular black outline appears around the selected part of the graph. As you move the cursor, the rectangle enlarges accordingly.
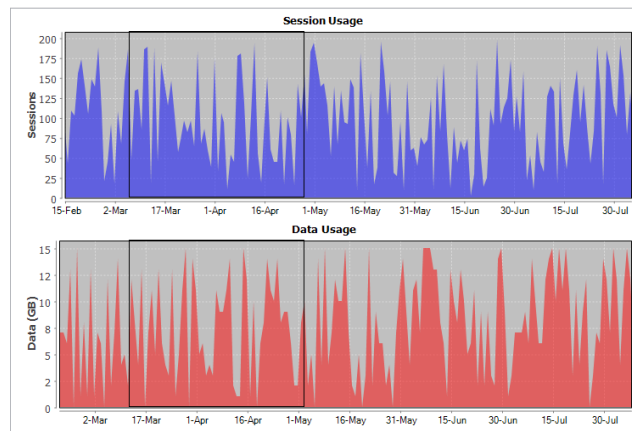
3. Release the mouse button. The graph is refreshed to display the period you selected. The **Period** field displays the setting **Custom**.

4. Repeat the process to zoom further.

5. To zoom out, place the cursor anywhere in the graph and, with your left mouse button pressed, drag the cursor from right to left. The graph returns to its original period, that is, the period it displayed when the window opened.

License Administrator

License Administrator is the tool used to manage keys for licensing. It is fully documented in License Administration in *Zen User's Guide*. The following is a summary of common monitoring tasks that you can conduct with the tool, with links to *Zen User's Guide*.

Item to Monitor	Using the Graphical User Interface	Using the Command Line Interface
License information (All license information for an authorized key, such as the product, product key, status of a key), platform to which the license applies, license type, user count, session count, data in use, expiration date of a license, vendor software that installed the license, and application to which the license applies)	To Display Information about a Key	To Display Information about a Key
User count (The number of allowed concurrent connections to the Zen database engine permitted by a product key)	To Determine a Total User Count	
Session count limit (The maximum permitted number of concurrent sessions as granted by a license agreement)	To Determine the Session Count Limit	
Data in use limit (The maximum permitted total size of all concurrently open data files as granted by a license agreement)	To Determine the Data in Use Limit	
Number of authorizations remaining for a key	To Display Remaining Authorizations	To Display Remaining Authorizations

Monitoring Database Access

Actian Corporation provides a companion product for Zen called AuditMaster that is used to monitor access at the database level. AuditMaster allows you to monitor changes data for auditing purposes, such as the following:

- Who accessed a record or performed a change.

- What access or change occurred, when, and where it originated.

- How the change was made.

AuditMaster provides a detailed audit trail and offers query and alert capabilities.

For product details, see the AuditMaster documentation.

Reviewing Message Logs

Zen provides various logging repositories for messages to assist you with troubleshooting. The logging falls into two broad categories:

- All messages. These messages include status, error, warning, and information messages. They can originate from any Zen component, including the license administration components.

- Licensing messages. These messages alert you about licensing issues and provide troubleshooting information. They originate from license administration components.

The following table summarizes the repositories.

Repository	Written To By
Notification Viewer	License administration components
Operating System Event Log	License administration components (Windows)
	Zen components writing error messages to Zen Event logs (Windows)
	All Zen components (Linux and Raspbian)
Zen Event Log (zen.log)	All Zen components (Windows)

Licensing Messages

Several of the logging repositories emphasize licensing messages. If a key is determined to be invalid, the key changes state from active to failed validation. The database engine functions normally for a certain number of days so that you have ample time to correct the validation failures.

If you do not correct the failures before the number of days ends, the key changes state again to disabled. The key is no longer valid and the database engine cannot access data files.

Because you need to attend to a failed validation in a timely manner, the state change of the key is brought to your attention as soon as possible. For example, a message is logged to all of the message repositories. The most evident of these is Zen Notification Viewer. License Administrator also displays the state of keys, which you can check at any time by initiating a validation action. See License Administration in *Zen User's Guide*.

Change in State of Key

The following table explains the type of message returned based on the change in state of the key.

Message Type	Change in State of Key		Scenario
	From	**To**	
Warning	Active	Failed Validation	A validation action detects that the key has failed validation. For example, the machine signature cannot be determined, no longer matches the key, or one or more licensing components cannot be loaded.
			The database engine functions normally for a certain number of days so that you have ample time to correct the validation failures.
Warning	Disabled	Failed Validation	Only some of the conditions that resulted in a disabled key have been corrected, but at least one condition still needs to be addressed.
			Because the key has changed state to failed validation, the database engine functions normally for a certain number of days so that you can correct the remaining validation failures.
Error	Failed Validation	Disabled	The number of days provided to correct the failed validation have expired. The conditions causing the failed validation were not corrected before the expiration. The key is no longer valid and the database engine cannot access data files.
Information	Failed Validation	Active	The conditions that caused the failed validation is corrected. The key is valid and the database engine has all functionality.
Information	Disabled	Active	The condition(s) that disabled the key is corrected. The key is valid and the database engine has all functionality.

Note that no messages are logged for keys in the expired state (which applies only to temporary keys) or the inactive state (which applies to keys still registered on the machine from previous versions of Zen).

Logging Frequency

The following table lists the frequency with which licensing messages are logged for particular actions.

Initiating Action	Logging Frequency	Logging Repository[1]
Key changes state as described under Change in State of Key.	Immediately	• Notification Viewer • Operating System Event Log • Zen Event Log
Key remains in failed validation state	Once a day reminder	• Notification Viewer • Operating System Event Log • Zen Event Log
A validation action invoked programmatically through API call See To Display Remaining Authorizations in *Zen User's Guide*, PvValidateLicenses() in *Distributed Tuning Interface Guide*, and ValidateLicenses in *Distributed Tuning Objects Guide*.	Immediately	• Operating System Event Log • Zen Event Log
Warning or error messages originating from the Zen licensing server	Immediately	• Operating System Event Log • Zen Event Log

[1] Message logging follows a one-way hierarchy: any licensing message logged to Notification Viewer is also logged to the Operating System Event Log and to the Zen Event Log. Similarly, any licensing message logged to the Operating System Event Log is also logged to the Zen Event Log.

Notification Viewer

Notification Viewer is tool for displaying messages logged by the licensing components. Its purpose is to inform you of noteworthy licensing messages (see Logging Frequency) in a noticeable but unobtrusive manner.

By default, Notification Viewer is installed with Zen Enterprise Server and Cloud Server, 32- and 64-bit, on Windows and Linux and with Zen Workgroup on Windows. Also by default on Windows platforms, Notification Viewer restarts when you restart Windows.

On Windows platforms, the executable is named **notifyviewer.exe**. It provides a single running instance for a user. An attempt to start Notification Viewer when it is already running brings the GUI to the front of the application displays.

On Linux and Raspbian, the tool is a shell script named **notifyviewer**. Each time the shell script executes it starts another instance of Notification Viewer. If you restart the operating system, you must restart Notification Viewer. The shell script is not automatically executed upon restart.

Command Line Options

You can specify how you want the tool to start with the following command line options.

Option	Meaning
(no option)	If you start the tool without specifying an option, the GUI opens and a tray icon appears if the operating system supports a system tray.
-tray	Starts the tool with the GUI hidden and displays a tray icon. If the operating system does not support a system tray, the GUI opens instead of remaining hidden.

Notification Viewer provides two interfaces: system tray icons and a graphical user interface.

System Tray Icons Interface

By default on Windows, Notification Viewer starts with the GUI hidden and displays its system tray icon. On Linux, Notification Viewer starts as a GUI and displays its system tray icon if the distribution supports a system tray. After starting, the tool begins monitoring licensing messages.

If Notification Viewer detects unread messages, the tray icon visibly changes to indicate unread messages. See Tray Icons.

Notification Viewer also displays two types of tooltips. The mouse-over tooltip displays the number of important unread messages (if any), the total number of unread messages, or the name of the tool if all messages have been read. A balloon tooltip displays when Notification Viewer detects messages that need to be brought to your attention. On Windows, the balloon tooltip remains visible until you dismiss it directly or perform a keyboard or mouse operation. On Linux, you must click the balloon tool tip to dismiss it.

Popup Menu

The popup menu for the tray icon contains two menu items: Open which opens the GUI, and Exit which closes the tool. Right-click the tray icon to display the menu.

Tray Icons

The following table explains the meaning of the tray icons.

Icon	Meaning
	Notification Viewer is running and monitoring licensing messages. This icon indicates a normal condition in which all messages have been read.
	Notification Viewer contains unread messages. This icon remains visible until all unread messages are read. See Left panel.

Graphical User Interface

You can open Notification Viewer GUI by double-clicking the tray icon or by right-clicking the tray icon and clicking Open. By default on Linux, Notification Viewer starts as a GUI and displays its system tray icon. If you want to change the startup behavior, pass the -tray option to the notifyviewer shell script. If the Linux distribution does not support a system tray, Notification Viewer displays the GUI but no system tray icon. In that case, start Notification Viewer by running the shell script.

When the GUI is visible, unread messages are immediately added to the GUI. In addition, the tray icon tooltip is shown and the icon changes to indicate unread messages.

Zen tracks which records in the notification file are read or unread for each user. That is, each user displaying the GUI sees all of the messages, but whether a particular message is read or unread varies by user.

If a system tray is not supported by an operating system, the Close command for Notification Viewer terminates the tool. If a system tray is supported, then closing hides the GUI and the tray icon becomes the only visual indication that the tool is still running.

Toolbar and Panels

The Notification Viewer GUI provides a toolbar and two main panels as shown in the following figure.
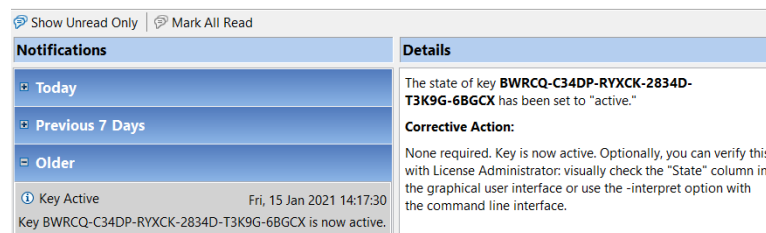
GUI Element	Description
Toolbar	Provides two options: • Show only unread messages (toggle on, toggle off) • Mark all messages as read
Left panel	Contains a scrollable list of messages. They are sorted with the latest messages on top and arranged in three groups: Today, Previous 7 days, and Older. You can expand and collapse each group. Each message shows an image for the type of information, a caption string, the message date and a brief description. All text is in a bold typeface if the message is unread. To mark an individual message as read, click the message to select it.
Right panel	Shows the message details, which provide a full description of the message, and when applicable, suggestions to resolve any issues.

Operating System Event Log

Zen writes the following entries to operating system event logs:

• Zen license administration messages

• Zen error messages written to the Zen Event Log (zen.log)

• All Zen component messages on Linux and Raspbian, including the above messages

For more information, see the following topics:

• Windows Platform Event Logs

• Linux and Raspbian Distribution Event Logs

Windows Platform Event Logs

Windows operating systems provide a method to log events categorized as Application, Security, or System. Zen logs general error messages and selected licensing messages to the Windows Application Event Log when they are written to zen.log.

For licensing messages, any event categorized as an error or warning is logged. This includes messages that result from a change in the state of a key as well as other warning and error messages (see Logging Frequency). In addition, certain information messages are logged, such as those listed under Change in State of Key.

Viewing Event Logs

Windows operating systems provide a graphical user interface tool called Event Viewer to view and manipulate event logs. It can be accessed through the Windows PC Settings or Control Panel or by executing the command eventvwr.msc from a command interface.

Zen displays the following for an event:

- Date and Time – Date and time of event

- Source – Actian Zen

- Task Category – Zen

- Type/Level – Level of the event: Information, Warning, or Error

- Event ID – 1000

- User – N/A

- Computer – Name of the computer

In addition, the Keyword column displays "Classic" and the Log column displays "Application." Event Viewer allows you to display additional columns, but Zen provides no data for them.

Linux and Raspbian Distribution Event Logs

On Linux and Raspbian distributions, all Zen components write messages to the standard logging system, syslog. By default on Linux and Raspbian, syslog writes to the file /var/log/messages. Optionally, for SQL Connection Manager only, you can also log messages to the event.log file.

Event.log File and Bti.ini

Bti.ini is a Zen configuration file used on Linux and Raspbian distributions. By default, the file is located in /usr/local/actianzen/etc.

The file lets you configure settings for the SQL Connection Manager (the SQLManager section in the .ini file). One of the settings, LogEvent, determines the type of event messages logged to the event.log file. By default, event.log is located in /usr/local/actianzen/bin.

Bti.ini Parameters for SQLManager Section	Description
MgrPort	Sets the port number used by the SQL Connection Manager. The default is 1583.
MgrUseTransport	Sets the type of protocol used by the SQL Connection Manager. This must be set to TCP.

Bti.ini Parameters for SQLManager Section	Description
LogEvent=*msg_type*	Specifies one of the following values for *msg_type* to indicate the type of messages logged to event.log (the default is 1): • 0 – no logging • 1 – errors only • 2 – errors plus warnings • 3 – errors plus warnings plus information messages • 4 – errors plus warnings plus information messages plus connect.log
InstallDirectory=/usr/local/actianzen	Activates the connection log: /usr/local/actianzen/connect.log

Zen Event Log (zen.log)

On Windows platforms, all Zen components write status, error, warning, and information messages to the Zen event log. On Linux and Raspbian distributions, Zen does not use an exclusive event log. Instead, all Zen components write messages to the standard logging system, syslog. See Linux and Raspbian Distribution Event Logs.

The Zen event log is zen.log, located by default in *application_data_directory*\actianzen\logs. All Zen components on Windows platforms write to this log file. If two or more applications using the Zen database engine are running on the same machine, they share zen.log.

Zen.log Fields

The contents of zen.log consists of text messages in the format described in the following table.

Field	Contents
Date	Automatic date stamp in *mm/dd/yyyy* format.
Time	Automatic time stamp in *hh:mm:ss* format. Also indicates AM or PM.
Component	File name of component returning the error (prefix only, no extension).
Process	Instance ID of the component, which is the process ID of the component.
Process Name	Path and name of the component, truncated to the last 15 characters.
Computer Name	Name assigned to the machine hosting the process, truncated to the first 15 characters.

Field	Contents
Type	A single character: I for Information, W for Warning, or E for Error.
Message	The message text which may be either a string retrieved from a resource associated with the calling component or a text string passed directly from the calling component.
	Some message text may contain numeric values, which may be in decimal or hexadecimal format. The characters "0x" precede any hexadecimal values to distinguish them from decimal values.
	Some message text may also contain information specific to an OEM application, such as a link to a vendor's website and troubleshooting information.

An entry may be followed by binary data in standard hexadecimal format. The binary data has no length limit.

Zen.log Example Entry

The following shows an example of the type of data contained in zen.log.

Date	Time	Component	Process	Process Name
5/10/2019	9:53:06 AM	LicenseMgr	9048	zenengnsvc.exe

Computer Name	Type Category	Message
USRegion2Svr	W	License failed validation. Remaining Days: 14

Receiving Email Notification of Messages

Zen does not include email notification of event messages because products from other vendors that provide such functionality are readily available. This topic lists some of those products and also discusses use of the event log to monitor actions around Zen licensing and product keys.

Products That Provide Email Notification of Monitored Events

The following table is a partial list of products that can provide email notification of events in the operating system event log. The products are listed alphabetically. Actian Corporation does not endorse one product over another. Since the products listed are from other vendors, the products and the discussion about them may vary from what is presented here.

Note that such products typically require either an agent to be installed or a remote access mechanism like Windows Management Instrumentation (WMI) or secure shell (SSH) to be enabled.

Product	Cost	Platform	Discussion
Hyperic www.hyperic.com	Contact Hyperic	Windows	Requires installing an agent on the monitored machine and opening a firewall port for the agent.
		Linux and Raspbian	Requires enabling SSH on the monitored machine and opening a firewall port for SSH.
Nagios www.nagios.org/	Free	Windows and Linux	Requires installing an agent on the monitored machine and opening a firewall port for the agent.
Spiceworks www.spiceworks.com	Free	Windows	Requires enabling remote WMI on the monitored machine and opening a firewall port for remote WMI.
		Linux	Requires enabling SSH on the monitored machine and opening a firewall port for SSH.
System Center Configuration Manager (SCCM) www.microsoft.com	Contact Microsoft	Windows	Requires either an agent or enabling remote WMI on the monitored machine. The firewall is automatically adjusted when WMI is enabled. If an agent is used, also requires opening a firewall port for the agent. This product is available only for Windows operating systems.

Product	Cost	Platform	Discussion
ZenOSS www.zenoss.com	Contact ZenOSS	Windows	Requires installing an agent on the monitored machine and opening a firewall port for the agent.
		Linux and Raspbian	Requires enabling SSH on the monitored machine and opening a firewall port for SSH.

Event Log Content to Monitor for License Event Messages

All of the items listed under Products That Provide Email Notification of Monitored Events have the ability to identify content in the operating system event log. The identification methods differ between Windows platforms and Linux and Raspbian distributions. The intent of this information is not to discuss the specific monitoring methods of each product. Refer to the vendor documentation for product details.

Windows Platforms

The following table identifies the properties in the operating system event log specific to Zen licensing. Configure your monitoring product to send email notifications based on the values of those properties.

Operating System Event Log Property	Value to Monitor
Source	Actian Zen
Task Category	Product Keys
Type	Warning, Error

Examples

Suppose that you want to receive email notifications for all Zen licensing event messages that are warnings or errors. Configure your product to monitor the following conditions.

- Source="Actian Zen"
- Task Category="Product Keys"
- Type="Warning" | Type= "Error"

Linux and Raspbian Distributions

The Linux and Raspbian syslog does not contain the same properties as the Windows operating system event log. You must configure the notification product to identify events and strings in the syslog. Based on events and string comparisons, the product can then initiate the desired action, such as send an email notification.

Note that all messages – whether licensing related or not – written to the syslog by the Zen database engine contain the string `mkded`. The following list identifies some of the strings in the syslog specific to Zen licensing. One way to monitor the syslog is to configure a string comparison based on content that contains `mkded` and one of the following strings:

- `disabled`
- `failed validation`
- `for key`
- `validation of key`
- `capacity for session count`
- `capacity for data in use`
- `user count`

Testing Btrieve Operations

The following topics cover use of the Function Executor tool:

- Function Executor Concepts
- Function Executor Graphical User Interface
- Function Executor Tasks

Function Executor Concepts

This section contains the following topics:

- Overview
- What Function Executor Can Do
- Function Executor Features
- Automatic Mode in Function Executor
- Where to Learn More

Overview

Function Executor runs on Windows. With this interactive tool, you can learn how Btrieve operations work. Btrieve operations are the same as MicroKernel Engine operations.

By allowing execution of Btrieve operations one at a time, Function Executor enables application developers to simulate the operations of a Btrieve application. This simulation can isolate the database calls from the rest of your application, which can help in testing and debugging your program.

Function Executor is primarily a tool for application developers. This chapter assumes that you have a basic knowledge of Btrieve operations. For more information about Btrieve operations, see the *Btrieve API Guide*.

What Function Executor Can Do

- Perform Btrieve operations while monitoring the contents of memory structures.
- Allow you to capture a series of Btrieve operations and save them as a history file for later playback
- Display the Btrieve version for clients, and local and remote engines.
- Display the Btrieve characteristics of data files and allow you to save those characteristics as a template (description file) or create a new file based on those characteristics. See File Statistics for more information.
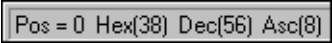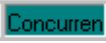
Function Executor Features

Function Executor features include the following:

Editor Status Bar

The status bar contains the following elements:

8	The last/current status code is shown in the editor window's status bar at the bottom of the window for the open file, and appears red if the last status was not zero.
Operation B_GET_NEXT (6) returned status 8 The current positioning is invalid.	Placing the mouse cursor over the status code shows a description of the status and what operation caused it. You can also click the display in red in order to display the full help for the status code. See To get help for a status code for more information.
Pos = 0 Hex(38) Dec(56) Asc(8)	When your cursor is in the input area on the main window of Function Executor, the status bar displays the offset within the buffer: the hex, the decimal, and the ASCII value of the byte you are presently on.
1 of 1	The status bar also indicates how many operations have been performed when there are multiple items in the queue. Normally this displays 1 of 1, but if you are executing multiple operations, it will display the count as the operations are executed.
Concurren	The status bar also displays when you are in a transaction.

Statistics

Clicking the **File Statistics** icon displays a dialog box listing statistics for the currently open file. You can print these statistics to a text file or save them to a description file usable by a `butil -create` command. You may also create a new blank file with the same characteristics.

File Size	1,077,248	Page Size (bytes)	4096
File Version	9.5	Number of Records	1500
Record Length	425	Unused Preallocated Pages	0
Number of Keys / Segments	3/9	Unused Linked Duplicate Pointers	0
Open Mode	Normal		

Flags

Record Compression	☐	Free Space Threshold	5%
Page Compression	☐	Key Only	☐
Variable Records	☐	Index Balancing	☐
Truncate Blanks	☐	Page Preallocation	☐
Contains VATs	☐		

D = Dups M = Mod R = Rep Dups A / L = Null < = Desc I = Caseins

Key #	Seq	Unique	Pos	Len	Attributes	Data Type
0	1	1500	1	8	M	Unsigned Binary
1	1	1498	26	1	D M R <	??
1	2	1498	27	26	D M R I	ZString
1	3	1498	9	1	D M R <	??
1	4	1498	10	16	D M R I	ZString
2	1	896	117	1	D M R <	??
2	2	896	118	3	D M R I	ZString
2	3	896	85	1	D M R <	??
2	4	896	86	31	D M R I	ZString

Get and GetExt

From the **Get** menu, you can retrieve the First, Next, Previous, and Last records in the table. The **GetExt** menu includes the **Goto Percent, Get Position**, and **Find Percent** commands.

The **Get** and **GetExt** commands are available from both the menu bar and toolbar. The toolbar offers **Step (Physical)** and **Get (Logical)**, allowing you to move either through the natural order of the file (physical) or in a specific order (logical).

Goto Percent allows you to choose whether to jump to a point within the physical layout of the file, or down any key path, limited to the keys defined in the file. You can also set lock biases using the option buttons in the Locks group box.

Find Percentage is the opposite of **Goto Percent.** It tells you how far into the data you are, depending on whether you are stepping through the file logically or physically.
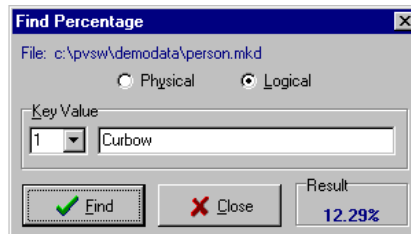
Transaction Toolbar
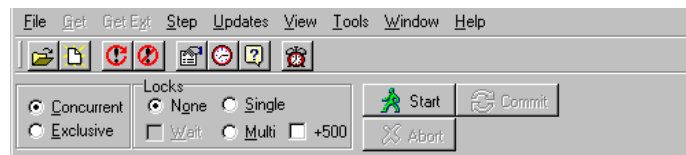
The Transaction toolbar lets you start, stop, and abort transactions. You can set all aspects of the Transaction API through this toolbar, and the operation is executed immediately. The Transaction status also appears on the main window status bar, since it affects all open files for the client ID.

Login Dialog

The Login dialog box allows you to perform the Btrieve login operation via a GUI interface. For more information, see topics about database security and its configuration.

History Log

When you perform operations using the Function Executor tool, they are recorded in a History log. You can use this log to perform the operations contained therein, or save the history as a file that you can later reload and step through.
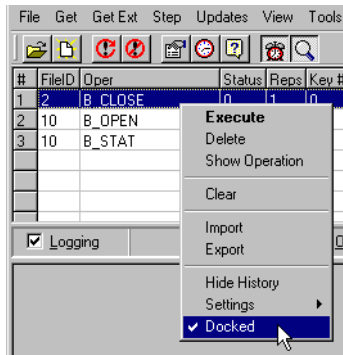
See the following topics for more information about the History log:

- [History](#)
- [History Tasks](#)

Viewing as Any Data Type

When a file is open, you can right-click any position in the buffer and select **Show As**. A dialog box appears in which you can view the bytes at the chosen buffer position as any data type.
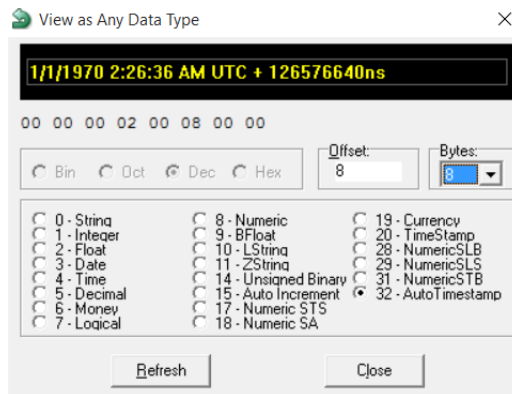
The following controls are available in Function Executor:

Control	Description
Repeat	Allows you to repeat commands.
Create	Allows you to create a new file.
File Statistics	Gives information from the BSTAT function. You can print these statistics.
MDI	The Multiple Document Interface permits the opening of multiple files.

Control	Description
Reset	Reset Client ID
Stop	Btrieve Stop
Version	Btrieve Version

Automatic Mode in Function Executor

For each open file (see Application Window), you have the option of performing Btrieve operations with or without the assistance of Function Executor. You do not need to make any configuration changes to use one method or the other. Whether you use automatic mode or manual mode depends on which GUI controls you use.
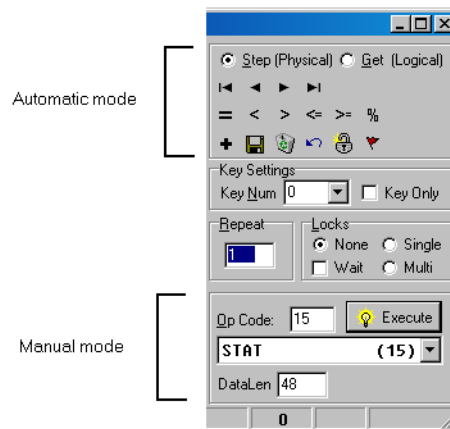
Note: Selections from the menus (see Application Window) also are part of the automatic mode.

When you click a button in the automatic mode area, the tool provides the following assistance:

• Data buffers and data lengths are set automatically to prevent status code 22.

• Information is requested appropriate to the operation.

Where to Learn More

Function Executor is a valuable tool for program developers, but it assumes you have a working knowledge of Btrieve fundamentals. See the following topics to understand all tool features:

• MicroKernel Engine fundamentals in *Zen Programmer's Guide*

• *Btrieve API Guide*

• Various security topics in the advanced user documentation

Function Executor Graphical User Interface

This topic describes the objects on Function Executor graphical user interface (GUI).

- Application Window

- Main Window

- Login and Logout

- Open File Dialog

- Create New Btrieve File Dialog

- Transactions Toolbar

- File Statistics

- History

Application Window

The table below the following image explains the window components. Click an area of the image for which you want more information.
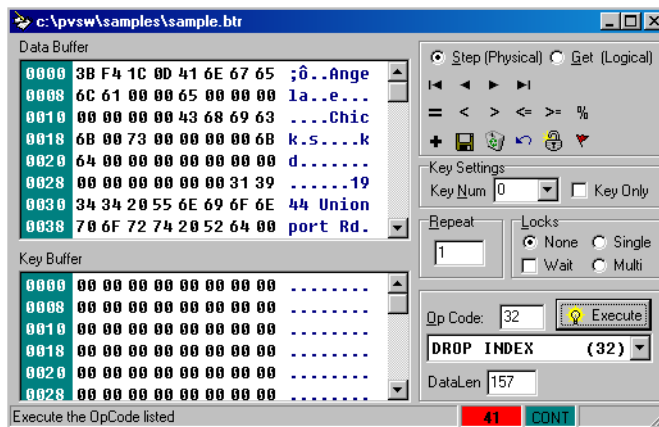
GUI Object	Description	Related Information
File menu	Allows you to perform the following commands and Btrieve operations: • Login and Logout • Open and Close • New • Print Setup • Set Owner Name and Clear Owner Name • Start Continuous Operations or End Continuous Operations • Reset, Stop, and Exit	Performing Operations Tasks Opening a File Tasks
Get menu	Allows you to perform the following Btrieve operations: • Get First and Get Next • Get Previous and Get Last • Get Greater Than and Get Greater or Equal • Get Less and Get Less or Equal	Performing Operations Tasks
GetExt menu	Allows you to perform the following Btrieve operations: • Goto Percent • Get Position • Find Percent	Performing Operations Tasks
Step menu	Allows you to perform the following Btrieve operations: • Step First and Step Next • Step Previous and Step Last	Performing Operations Tasks
Updates menu	Allows you to perform the following Btrieve operations: • Insert, Update, and Delete You can also release locks using this menu.	Performing Operations Tasks
View menu	Allows you to display GUI elements: • Toolbars (main and transactions) • History window • File statistics window • Engine version using the Btrieve Version operations	History Tasks
Tools menu	Allows you to perform the following Btrieve operations: • Get Directory and Set Directory	

GUI Object	Description	Related Information
Window menu	Allows you to perform windowing operations: • Cascade windows and Tile windows • Select from a list of open windows	
Help menu	Allows you to select from a list of help resources for this tool.	To get help for a status code
Open	Displays a dialog box from which you select a Btrieve file to open.	Opening a File Tasks
Create	Displays a dialog box with which you can create a new Btrieve file.	Creating a Btrieve File Tasks
Reset	Resets the current client connection (Btrieve operation 28) and closes all files opened with that client ID.	
Stop	Ends transactional operations (Btrieve operation 25) and closes all open files.	
Statistics	Displays a dialog box listing the currently opened file's statistics. You can print these statistics to a text file or save them to a description file usable by BUTIL -CREATE	
Version	Displays information about the version of Zen (using Btrieve operation 26) that you are running. If no file is open, you will see information about the requester DLLs and any local MicroKernel engine. If you open a file on a remote server, you will see information about the server and requester DLLs.	
Show help	Toggles whether a pop-up dialog box displays when a non-zero status code is received.	
Show history	Toggles whether the History window is displayed.	To display the History window History History Log
Operation count	Indicates the current operation number. This is used when you set the Repeat to a value greater than one.	Performing Operations Tasks

GUI Object	Description	Related Information
Transaction state ![Exclusive]	Indicates the current state of any transactions. Can be: • Blank, if no transaction is in effect • Concurrent • Exclusive	Performing Operations Tasks Transactions Toolbar

Main Window

A main window displays for every open file. Click an area of the image for which you want more information.

GUI Object	Description	Related Information
Title Bar	Lists the full path of the open data file.	To open a data file with Function Executor
Data Buffer	Specifies a data value. For read and write operations, the Data Buffer contains records. For other operations, the Data Buffer contains file specifications, filtering conditions, and other information the database engine needs for processing the operation. This control corresponds with the Data Buffer parameter.	
Key Buffer	Specify the path for the data file for which you want to perform a Btrieve operation.	

GUI Object	Description	Related Information
Step vs. Get	Toggles between Step and Get operations	
Get/Step First	Performs the Get or Step Next operation	Performing Operations Tasks
Get/Step Prev	Performs the Get or Step Previous operation	Performing Operations Tasks
Get/Step Next	Performs the Get or Step Next operation	Performing Operations Tasks
Get/Step Last	Performs the Get or Step Last operation	Performing Operations Tasks
Get Equal	Performs the Get Equal operation	Performing Operations Tasks
Get Less Than	Performs the Get Less Than operation	Performing Operations Tasks
Get Greater Than	Performs the Get Greater Than operation	Performing Operations Tasks
Get Less or Equal Than	Performs the Get Less or Equal Than Operation	Performing Operations Tasks
Get Greater or Equal Than	Performs the Get Greater or Equal Than Operation	Performing Operations Tasks
Get/Find Percent	Performs the Get or Find Percent operations	Performing Operations Tasks
Insert	Performs the Insert operation	Performing Operations Tasks
Update	Performs the Update operation	Performing Operations Tasks
Delete	Performs the Delete operation	Performing Operations Tasks
Cancel	Cancels the recent changes	
Unlock	Releases any locks.	
Set or Goto Bookmark	Sets or positions at a previously defined bookmark.	
Key Num	For most Get operations, specifies a key number, or index path, to follow for the current operation. For other operations, specifies such information as file open mode, encryption, or logical disk drive. This control corresponds with the Key Number parameter.	
Key Only	Specifies to get key only, not data.	
Repeat	Repeats the operation the number of times you specify.	
Locks	Specifies the locking behavior you want in for the current operation.	

GUI Object	Description	Related Information
Op Code	Specifies the current operation code plus its bias (if any). The default is **0**. If you are familiar with Btrieve operation codes, you can enter the desired code. Otherwise, use the **List** box to specify an operation. This control corresponds with the **Operation Code** parameter.	Performing Operations Tasks
Execute button	Performs the currently specified operation.	Performing Operations Tasks
Operations list	Lists all Btrieve operations and their codes. The default is **Open** (0). You can move quickly through the list by entering the first letter of the operation you want to perform.	Performing Operations Tasks
DataLen	Specifies the length (in bytes) of the Data Buffer. The default is 1024. For every operation that requires a data buffer, you must specify a buffer length. On many operations, the database engine returns a value to the Data Length. Generally, you should always specify a Data Length before you execute an operation. This control corresponds with the Data Buffer Length parameter.	Performing Operations Tasks
Status Code Indicator	Displays a numeric status code returned by the database engine and a brief message explaining the result of a Btrieve operation. For detailed information about these status codes and messages, refer to the *Status Codes and Messages* manual.	To get help for a status code
Continuous Operations Indicator	Displays the following on the bottom row of the file window if the file is in Continuous Operations mode (operation 42).	Using Continuous Operations

Login and Logout

The Login and Logout dialog boxes allows you to perform the Btrieve login and logout operations for the database engine. Click an area of either image for which you want more information.

Login Dialog

Logout Dialog

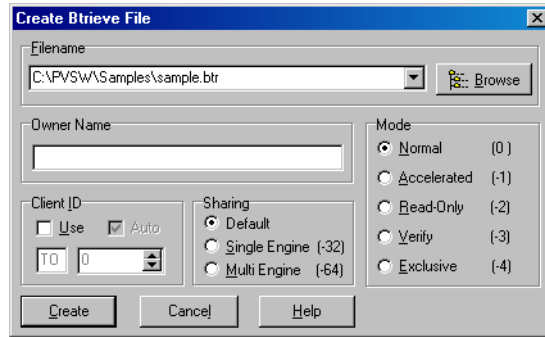| GUI Object	Description
Server	Specifies the server where the database resides to log in to or log out from. Note that a server name is required to access a database on a Linux operating system. See also Database URIs in *Zen Programmer's Guide*.
Database	Specifies the database on the server to which you want to authenticate.
User Name	The user name you want to authenticate against the database.
Password	The password for the user name.
Client ID	If you want this login or logout to apply to a specific client ID, click Use and specify the range. Otherwise, leave as is. See Client ID in *Btrieve API Guide*.
URI String	As you enter information in the forms, the resulting URI appears here.

Open File Dialog

This dialog box allows you to open a file. Click an area of the image for which you want more information.

GUI Object	Description	Related Information
Filename	Sets the location and name of the file that you want to open.	Opening a File Tasks
Owner name	Serves as a password for the Btrieve file, which is required to gain access to the file. A short owner name can be up to 8 bytes. The length of a long owner name depends on the file format. For more information, see Owner Names.	Opening a File Tasks
Mode	Sets the state of the file when it is opened. Based on the state, the database engine knows certain conditions that apply to the opened file. For example, a condition could be that the file can be read but not updated (read-only).	Opening a File Tasks See *Btrieve API Guide* for an explanation of modes.
Client ID	If you want this login to apply to a specific client ID, click Use and set the range. Otherwise, leave as is.	Opening a File Tasks
Sharing	The database engine ignores the Sharing options. The Sharing options applied only to a legacy version of the engine, Btrieve 6.15.	Opening a File Tasks

Create a Btrieve File

This dialog box allows you to create a Btrieve file based on an already open file. Click an area of the image for which you want more information.
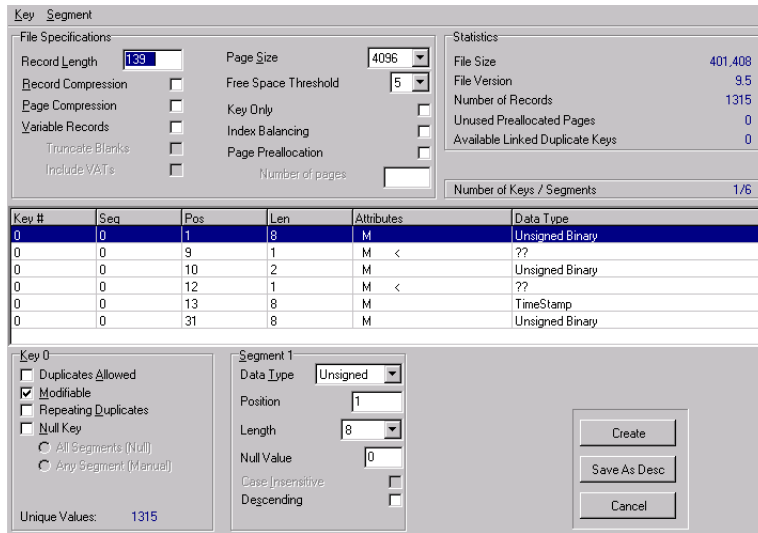
GUI Object	Description	Related Information
Filename	Sets the location and name of the file that you want to create.	Creating a Btrieve File Tasks
Owner name	Serves as a password for the Btrieve file, which is required to gain access to the file. A short owner name can be up to 8 bytes. The length of a long owner name depends on the file format. For more information, see Owner Names.	Creating a Btrieve File Tasks
Mode	Sets the state of the file when it is opened. Based on the state, the database engine knows certain conditions that apply to the opened file. For example, a condition could be that the file can be read but not updated (read-only).	Creating a Btrieve File Tasks See Open (0) in *Btrieve API Guide* for an explanation of modes.
Client ID	If you want this login to apply to a specific client ID, click Use and set the range. Otherwise, leave as is.	Creating a Btrieve File Tasks
Sharing	The database engine ignores the Sharing options. The Sharing options apply only to a legacy version of the database engine, Btrieve 6.15.	

Create New Btrieve File Dialog

This dialog box allows you to specify additional characteristics for the file being created. Click an area of the image for which you want more information.
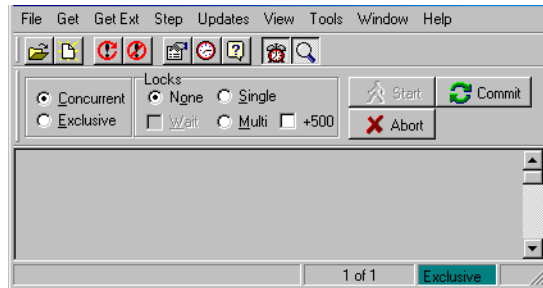
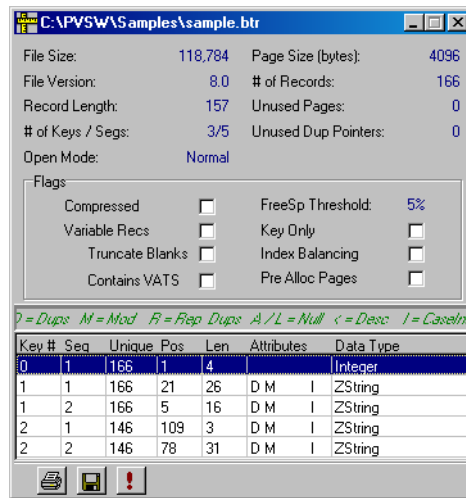| GUI Object	Description	Related Information
Key and Segment commands	Allow you to add or delete a key or to add, insert, or delete a key segment.	Creating a Btrieve File Tasks
File Specifications	Displays and allows you to modify characteristics of the file.	Creating a Btrieve File Tasks Record and Page Compression
Statistics	Provides read-only information about the file.	Creating a Btrieve File Tasks
Key and segment matrix	Displays the keys and segments for the file, listing the starting position, length, attributes, and data type for them. Clicking a row allows you to see, and modify if you want, information in the Key and Segment blocks.	Creating a Btrieve File Tasks
Key	Displays and allows you to modify characteristics of a key.	Creating a Btrieve File Tasks
Segment	Displays and allows you to modify characteristics of a key segment.	Creating a Btrieve File Tasks
Command buttons	Allow you create a data file, a description, or to cancel the dialog box.	Creating a Btrieve File Tasks

Transactions Toolbar

When a file is open, you can show the Transactions toolbar by selecting View > Toolbars > Transactions. This toolbar allows you to manually control transactions. Click an area of the following image to learn about what you can do with this toolbar.

GUI Object	Description	Related Information
Main application window	Main features of the program.	Application Window
Transaction type	Specifies whether you want an exclusive or concurrent transaction.	
Lock information	Specifies the type of locking you want in the transaction.	
Start transaction button	Starts a transaction.	
Commit transaction button	Commits the active transaction.	
Abort transaction button	Cancels the active transaction and rolls back any changes made.	
File area	This area contains one or more open files.	Main Window
Transaction indicator	Indicates whether you are currently in a transaction. If you are in a transaction, this display will show either Concurrent or Exclusive.	

File Statistics

When a file is open, you can show the File Statistics window by selecting View > File Statistics, clicking the File Statistics icon in the tool bar, or pressing Ctrl+S. It allows you to see the statistics for the file. Click an area of the following image to learn about what you can do in this window.

GUI Object	Description
File information	Displays statistical information about the file.
Flags	Allows you to view the flags set for this file. Any flag that is set will have a check mark next to its listing in the dialog box.
Keys	Allows you to view the keys defined in this file.
Key legend	Describes the single letter values that indicate Key attributes for the file. • D = DUP (Duplicatable) • M = MOD (Modifiable) • R = REPEAT_DUPS_KEY (repeating duplicates key) • A= NUL (All segment Null key) • L = MANUAL_KEY (any segment null key) • < = DESC_KEY (descending key value) • I = NOCASE_KEY (case-insensitive key) See also Key Attributes in *Zen Programmer's Guide*.
Print button	Allows you to print the file statistics. Set up the printer using the File menu.

GUI Object	Description
Save button	Allows you to save the current file statistics to a description file, which you can later use to make a new file with the same characteristics.
Exclamation point (create new file)	Allows you to create a new file based on these file statistics.

History

This dialog box allows you to see all the operations you have performed. Click an area of the image for which you want more information.
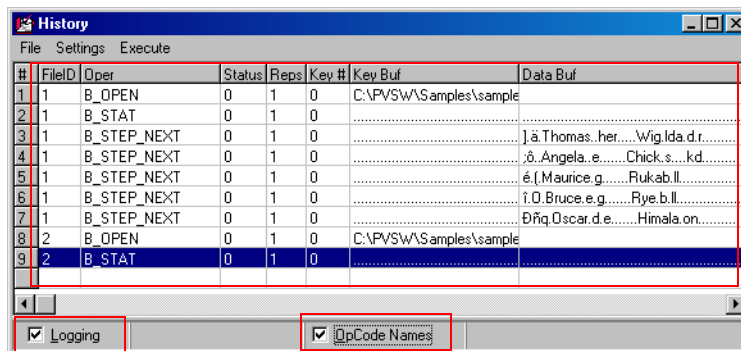
GUI Object	Description	Related Information
File menu	Allows you to perform the following operations: • Import a history file • Export a history file • Close the history window	

GUI Object	Description	Related Information
Settings menu	Allows you to specify the visual attributes of the History window. • Save on Exit - Specifies whether you want your customization of the History window to be saved between sessions. • Defaults - Resets the History window to default settings. This removes any settings you specified. • Docked - Toggles the state of the History window between a separate window and one that is attached to the Application Window. • Stay On Top - Toggles the state of the History window between one that can lose focus and one that cannot.	• To toggle the docking status of the History window • To toggle the Always On Top status of the History window • To reset the History window to default settings
Execute command	Loads the History Playback window	
List of operations	Lists operations that you have recently performed. Each operation is logged with the following information: • FileID • Operation name or number depending on the status of the OpCodeNames check box. • Status code when that operation was performed • Number of times that operation was performed. • Key number set for the operation. • Contents of the key buffer. • Contents of the data buffer.	• History Tasks • History
Logging	Toggles the inclusion of future operations in the history list.	
OpCode Names	Toggles the display in the Operation column between operation code names (such as B_OPEN) and operation code numbers (such as 0 for B_OPEN)	

Function Executor Tasks

Function Executor tasks are grouped into the following categories:

- Starting Function Executor Tasks

- Performing Operations Tasks

- Opening a File Tasks

- Creating a Btrieve File Tasks

- History Tasks

Starting Function Executor Tasks

To start Function Executor

1. Access **Function Executor** from the operating system **Start** menu or **Apps** screen or from the **Tools** menu in Zen Control Center.

2. The main window appears.
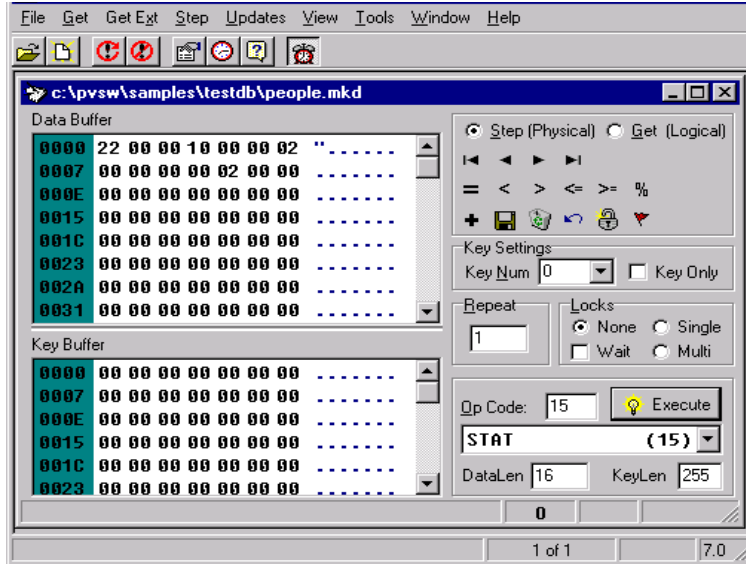
Performing Operations Tasks

Because Btrieve provides many operations, this chapter cannot explain them all. The following sections discuss some common operations as well as some new ways of performing them with the Function Executor.

Note: Selecting options from all menus performs the intended operation immediately. It does not fill in the grid and wait for you to execute the command as in previous versions. Also, closing the form closes each open file. No longer do you need to manually perform the close operation.

General Operations-Related Tasks

*   To get help for a status code

For other tasks, see these sections:

*   Opening a File Tasks

*   Creating a Btrieve File Tasks

*   History Tasks

To get help for a status code

1. When a status code is received using Function Executor, it is displayed on the status bar of the open file.

    Move your mouse so it hovers over the status code that is displayed in red.
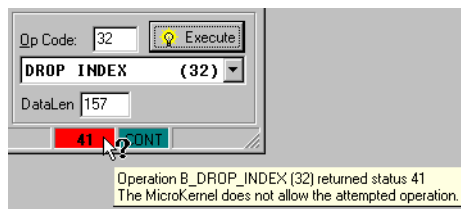
2. If you click the status code indicator, the full documentation for the status code is displayed as shown in this example:
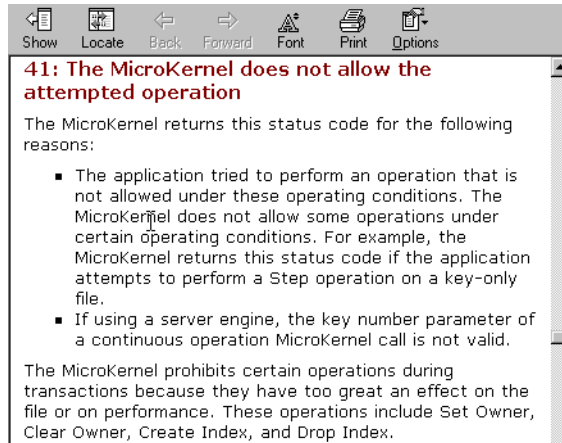
Opening a File Tasks

To open a data file with Function Executor

1. From the **File** menu, select **Open**. The following dialog box appears:
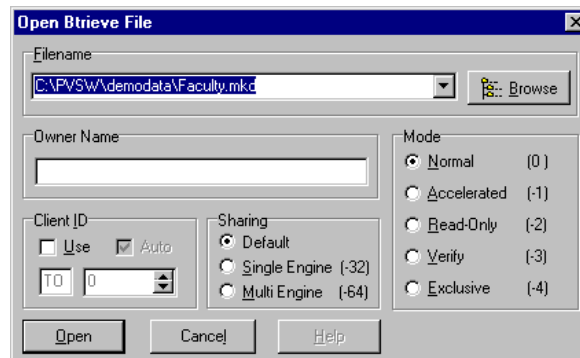
Note: Client ID: If you have Use disabled, Function Executor will use a BTRV() function call. If Use is enabled, it will use a BTRVID() function call for each operation you execute on this file. With Auto enabled, Function Executor will generate a client ID for you. If you have Auto disabled, then you may enter values manually.

2. Click **Browse**.

3. Double-click the desired file name.

Other Ways to Open a File with Function Executor

•   You can drag a file from Windows Explorer into the Open dialog box. This step fills in the file name for you.

•   You can drag one or more files into the main window.

•   After opening a file to display the editor window, you can use the OpCode 0 to open another file. The file appears in a new window.

•   You can run Function Executor from a command prompt and specify a list of file names to open. For example:

```
WBExec32 person.mkd billing.mkd
```

You can even use wildcard characters:

```
WBExec32 *.mkd
```

Also, in Windows you can associate files with .btr, .dat, .mkd, or any other extension with Function Executor. Then when you double-click the file in Windows Explorer, Function Executor opens it.

Creating a Btrieve File Tasks

There are two options in creating a Btrieve file with Function Executor. If a file is already open, you can clone it. Otherwise, you can start from scratch.

Caution! The Btrieve engine uses file names in ways that ignore extensions. Because of this, no two files in the same directory should share a file name and differ only in their file name extension. For example, do not name one data file invoice.btr and another invoice.mkd.

Method 1: Using a current file as a template

1.   From the **File** menu, select **New**. The following dialog box will appear:
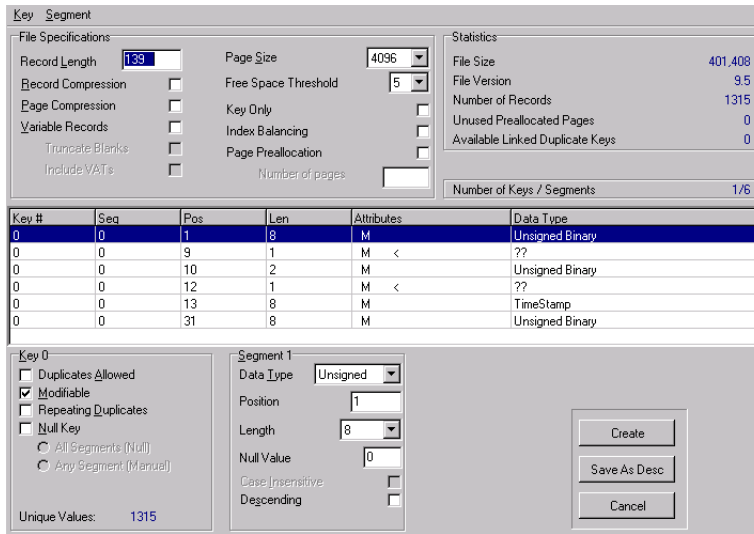
2. You can manipulate keys from this dialog box as well. You can **Add, Create**, or **Insert Segments** from the **Key** menu. You can also save the new file as a description for use with BUtil create. Select **Save As Desc** and indicate the name and location where you would like the file saved.

3. To create the file, click **Create**. This will open the file and display a message indicating success.

Method 2: Creating a new file from scratch

1. Click the **Create** icon on the main toolbar; or, if no file is open yet, you may click **File** and then **New**, as before.

2. If a file is already open on screen, a selection box will appear. Choose **Create New File from Scratch**.

3. The same dialog box as before will appear, but it will be blank, allowing you to input brand new values.

4. Start by adding a new Key using the **Key** menu, or press Ctrl-A.

5. Fill in the attributes for the key in the lower section of the dialog box.

6. Continue adding or removing new keys and segments as desired, using the menus or right-clicking the key in the list.

7. Now click the **Create** button to execute the B_Create (14) operation. This will automatically open the file on screen as well.

History Tasks

The following tasks are related to the History feature:

- To display the History window
- To toggle the docking status of the History window
- To toggle the Always On Top status of the History window
- To reset the History window to default settings

To display the History window

1. Click **View > History** or click the **History** button.

To toggle the docking status of the History window

1. If the History window is not visible, display it. (see To display the History window).

2. In the history items window, click the display to check or clear the Docked option as shown in the following figure:

When docked, the History window is connected to the application window as shown here. When not docked, the History window is a distinct window.

To toggle the Always On Top status of the History window

1. If the History window is not visible, display it (see To display the History window).

2. This feature only applies to the History window when it is in the undocked state (see To toggle the docking status of the History window).

3. In the history items window, right-click the display and check or clear the **Stays On Top** selection.

To reset the History window to default settings

1. If the History window is not visible, display it (see To display the History window).

2. In the history items window, right-click the display and select **Settings** > **Defaults**.

Manipulating Btrieve Data Files with the Maintenance Tool

The following topics are covered here:

Maintenance Utilities Overview

Zen provides both an interactive Maintenance GUI and a command line Maintenance tool. Both versions perform the following file and data manipulations:

- Create new data files based on file and key specifications you define.
- Provide file and key specifications for existing data files.
- Set and clear owner names for data files.
- Create and drop indexes on data files.
- Import and export ASCII sequential data.
- Copy data between Zen data files.
- Recover changes made to a file between the time of the last backup and a system failure.

While both utilities provide the same core functionality, minor differences exist. For example, the interactive Maintenance tool allows you to create description files based on file and key specifications you define. The command line Maintenance tool allows you to start and stop continuous operation on a file or set of files locally on the server.

Before you use either Maintenance tool, you should be familiar with Btrieve fundamentals, such as files, records, keys, and segments. For information about these topics, see *Zen Programmer's Guide*.

Note: The Zen product provides two categories of maintenance tools: Btrieve and SQL. The SQL Maintenance tool supports data source names (DSNs), which are used for relational access through ODBC.

Btrieve Interactive Maintenance Tool

The interactive Maintenance tool is a Windows application that runs on Windows 32-bit and 64-bit platforms. Use it if you prefer a graphical user interface or if you want to create a description file. This section contains the following major topics:

- File Information Editor
- Managing Owner Names
- Statistics Report
- Indexes
- Data

Each major topic contains tasks specific to that topic.

Extended File Support

The size of a MicroKernel data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the database engine file size limit because of the differences in the physical format.

When you are exporting large files, the interactive Maintenance tool detects when the unformatted file exceeds a 2 GB file size limit and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. Note that file size limit varies depending on the operating system and file system. The 2 GB size is the limit supported by the database engine.

The extension file uses a naming scheme in which the file names are similar to the base file name. In contrast to native MicroKernel Engine extension files which use a caret "^" to indicate extension file status, the unformatted extension files use a tilde "~" to avoid overwriting any existing extended engine files with the same base file name. The first export extension file is the same base file name with ".~01" extension. The second extension file is ".~02" and so on. These extensions are appended in hexadecimal format.

The naming convention supports up to 255 extension files, thus supporting files as large as 256 GB.

Additionally, when you import data from an unformatted file, the tool detects whether the file has extensions and loads the data from the extension file.

Long File Names and Embedded Spaces Support

Long file name support, including support for embedded spaces is available in all supported operating system environments. All references to files can contain embedded spaces and be longer than 8 bytes.

Older versions of Btrieve allowed spaces to be added at the end of a file name in path-based operations such as Open and Create. This is still the default behavior. Existing applications will not break. However, if you want to take advantage of file and directory names with embedded spaces, set the **Embedded Spaces** configuration setting for the client requester to **On**. Note that On is the default setting.

Even when you turn off the option, an application that accesses a file having a name with embedded spaces can enclose that name in double quotation marks while making the BTRV/ BTRVID/BTRCALL/BTRCALLID call to open or create the file.

Record and Page Compression

Zen provides two types of data compression: record and page. These two types may be used separately or together. The primary purpose for both compression types is to reduce the size of the data files and to provide faster performance depending on the type of data and on the type of data manipulation.

Record Compression

Record compression requires a file format of 6.0 or later. Record compression can result in a significant reduction of the space needed to store records that contain many repeating characters. The database engine compresses five or more of the same contiguous characters into 3 bytes.

When creating a file, the database engine automatically uses a page size larger than what is specified to allow room for the specified record length. If the uncompressed record length is too large to fit on the largest available page, the database engine automatically turns on record compression.

Because the final length of a compressed record cannot be determined until the record is written to the file, the database engine creates a file with record compression as a variable-length record file. Compressed images of the records are stored as variable-length records. Individual records may become fragmented across several file pages if your application performs frequent insertions, updates, and deletions. The fragmentation can result in slower access times because the database engine may need to read multiple file pages to retrieve a single record.

See also Choosing a Page Size, Estimating File Size, and Record Compression, all in *Zen Programmer's Guide*.

Page Compression

Page compression requires a file format of 9.5 or later. Internally, a Zen data file is a series of different types of pages. Page compression controls the compression and decompression of data pages within a file.

As a file is read from physical storage, data pages are decompressed and held in a memory cache. Record reads and updates are performed against the uncompressed data in the memory cache. When a write action occurs, the data page is compressed then written to physical storage. Depending on cache management, the compressed page is also retained in memory until accessed again.

If the type of data cannot be significantly compressed, the database engine writes the data to physical storage uncompressed.

Deciding When To Use Compression

The benefits obtained by using record compression, page compression, or both depends entirely on the type of data being compressed. Given that, the following table discusses some *general* factors to consider when deciding to use data compression or not.

Compression		Factors to Consider
Record	Page	
X		Record compression is most effective for the following conditions: • Each record has the potential for containing a large number of repeating characters. For example, a record may contain several fields, all of which may be initialized to blanks by your task when it inserts the record into the file. Record compression is more efficient if these fields are grouped together in the record, rather than being separated by fields containing other values. • The computer running the database engine can supply the extra memory required for compression buffers. • The records are read much more frequently than they are changed. If the fixed length portion of a record is longer than the page size minus overhead, compression is used automatically. Note that you cannot use record compression for key-only files or for files that use blank truncation.

Compression		Factors to Consider
Record	Page	
	X	Page compression is most effective for the following conditions: • Data is highly compressible using a ZIP-type compression algorithm. When the file size can be significantly decreased because of page compression, such as 4 to 1 compression, file performance can be increased significantly. • The pages are read much more frequently than they are inserted, updated, or deleted. Note that the database engine writes data pages to physical storage uncompressed if the data cannot be significantly compressed.
X	X	The use of record compression and page compression is most effective when records contain a large proportion of blank space and the pages are read much more frequently than they are inserted, updated, or deleted.

The Btrieve Maintenance Tool Interface

Access **Maintenance** from the operating system **Start** menu or **Apps** screen or from the **Tools** menu in Zen Control Center. The Maintenance main window looks like the following.

Menu Options

The interactive Maintenance tool provides the following menus:

Options	Allows you to display the **File Information Editor**, set and clear an owner name, generate statistics reports, and exit the tool.
Index	Allows you to create and drop indexes.
Data	Allows you to load data from ASCII files, save data to ASCII files, copy records between data files, and perform a roll forward operation to recover changes made to a data file between the time of the last backup and a system failure.
Help	Provides access to the Maintenance tool help system.

Getting Help

To access the Maintenance tool help system, click **Help** in the dialog box for which you want help, or choose a command from the **Help** menu, as follows:

Contents	Provides a list of Maintenance tool help topics.
About	Displays copyright information and the product version number.

File Information Editor

This section provides general information about the File Information Editor with which you can create new files based on file and key specifications you construct. Because this Editor allows you to load information based on an existing file, it is also useful for viewing file and key specifications on existing data files. You can also create a new file based on the file and key specifications of an existing file (similar to the **clone** command for **butil**, the command line Maintenance tool).

Caution! No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not name a data file Invoice.btr and another one Invoice.mkd in the same directory. This restriction applies because the database engine uses the file name for various areas of functionality while ignoring the file name extension. Since only the file name is used to differentiate files, files that differ only in their file name extension look identical to the database engine.

Open the File Information Editor by clicking **Options** > **Show Information Editor**.
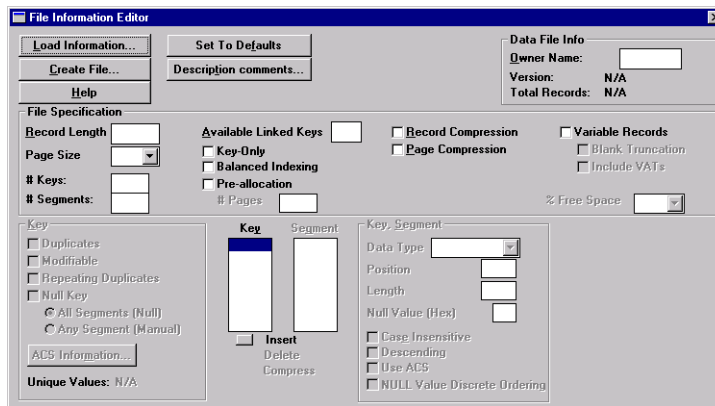
File Information Editor Dialog Elements

At the top of the Editor, the following buttons appear:

Load Information	Loads information for an existing file. When you load file information, you are not editing the file. Instead, you are loading a copy of the information about that file.
Create File	Creates a new file based on current information in the dialog box.
Set To Defaults	Sets the controls to default values.

Description Comments	If you are creating a description file, allows you to enter notes about the file.
Help	Displays help for the **File Information Editor** dialog box.

Data File Info

The **Data File Info** area, also at the top of the File Information Editor, contains the following controls:

Owner Name	Displays the file owner name, if one exists, as a string of asterisks.
Version	Earliest version of the database engine that can read all of the attributes of the file. For example, if you created a file using the 9.5 database engine but did not use any attributes specific to 9.5, the Maintenance tool displays 9.0 as the version number. See File Version Notes for additional information about file format versions.
Total Records	Total number of records in the file.

File Specification

The **File Specification** area is in the middle of the File Information Editor. The following table describes the controls in this box.

Control	Description	Range	Default
Record Length	Specifies the logical data record length (in bytes) of the fixed-length records in a file. For information about record length and overhead, see "Record Length" in *Zen Programmer's Guide*.	Minimum is 4 bytes. Maximum is variable. If the record length specified exceeds the page size minus overhead, the database engine automatically tries the next available page size for the file format. If the record length exceeds the maximum page size minus overhead, the engine turns on record compression.	100

Control	Description	Range	Default
Page Size	Specifies the physical page size (in bytes) for the file.	• 512–4096 for file versions prior to 9.0 (a multiple of 512 bytes up to 4096) • 512, 1024, 1536, 2048, 2560, 3072, 3584, 4096, or 8192 for file version 9.0. • 1024, 2048, 4096, 8192, or 16384 for file version 9.5. • 4096, 8192, or 16384 for file versions 13.0 and 16.0.	4096
# Keys	Indicates the number of distinct keys (as opposed to key segments) currently defined in the Editor. Reflects the number of keys in the Key list.	• 0–119	0
# Segments	Indicates the number of key segments currently defined in the Editor. Reflects the number of segments in the Segment list.	• 0–119 for file versions before 9.5 • 0–420 for file version 9.5 • 0–378 for file versions 13.0 and 16.0	0
Available Linked Keys	Specifies how many 8-byte place holders you want to reserve for future linked-duplicatable keys. If you are loading information based on an existing data file, this value reflects the number of place holders currently available in that file. (The number of originally reserved place holders is not stored in the file.)	• 0–119	3
Key-Only	Indicates whether the file is key-only. Not applicable if you turn Record Compression on, if you turn Variable Records on, or if you define more than one key for the file.	On or Off	Off
Balanced Indexing	Specifies that the file uses the balanced indexing method of managing key pages.	On or Off	Off
Pre-allocation	Specifies that the file uses preallocated pages.	On or Off	Off

Control	Description	Range	Default
# Pages	Specifies the number of pages you want preallocated when you create the file. Applicable only if Pre-allocation is turned on. If you are loading information based on an existing data file, this value reflects the number of unused, preallocated pages left in that file. (The number of originally preallocated pages is not stored in the file.)	1–65535	0
Record Compression	Specifies that the file uses record compression. Not applicable for key-only files or files that use blank truncation. See also Record and Page Compression.	On or Off	Off
Page Compression	Specifies that the file uses page compression. See also Record and Page Compression.	On or Off	Off
Variable Records	Specifies that the file can contain variable-length records.	On or Off	Off
Blank Truncation	Specifies whether the file uses blank truncation on variable records to conserve disk space. Applicable only if Variable Records is turned on.	On or Off	Off
Include VATs	Specifies whether the file supports Variable-tail Allocation Tables for faster access to data in very long records. Applicable only if Variable Records is turned on.	On or Off	Off
% Free Space	Specifies the amount of unused space a file's variable pages must have available before the database engine creates a new variable page. Applicable only if Record Compression or Variable Records are turned on.	5, 10, 20, or 30	5

Key

At the bottom left in the dialog box is the **Key** group box. The following table describes the controls in this area. These controls are specific to the key highlighted in the **Key** list, not just to the current key segment. When you change the setting for one of these controls, the change affects *all* segments of the specified key.

Control	Description	Default
Duplicates	Specifies that the key can have duplicate values (linked duplicates). See Methods for Handling Duplicate Keys.	On
Modifiable	Specifies that the key value can be modified after creation. Allowing modification of key values does not affect performance. Key pages are only updated if the actual key value changes, not if non-key fields in a particular record are changed.	On
Repeating Duplicates	Specifies that the database engine uses the repeating duplicates method of storing duplicate key values. See Methods for Handling Duplicate Keys.	Off
Sparse Key (Null Key)	A sparse key contains fewer key values than the number of record in the file. To specify which key values are excluded from the index, see the next two controls. Applicable only to keys that contain nullable segments.	Off
All Segments (Null)	Specifies that if all key segments in the record contain a null value, the database engine does not include that record in the index. Applicable only if Sparse Key (Null Key) is turned on. Equivalent to key flag 0x0008. Whether a segment is evaluated as null is determined solely by the null indicator segment for that field; the contents of the field are not evaluated.	Off
Any Segment (Manual)	Specifies that if one or more key segments contains a null value, the database engine does not include that record in the index. Applicable only if Sparse Key (Null Key) is turned on. Equivalent to key flag 0x0200. Whether a segment is evaluated as null is determined solely by the null indicator segment for that field; the contents of the field are not evaluated.	Off
ACS Information	Allows you to specify an alternate collating sequence (ACS) for the key. Applicable only if the **Use ACS** check box is selected for a segment of the key.	Off
Unique Values	Indicates the number of unique key values in the file. Applicable only if you are loading information based on an existing data file.	N/A

Key List and Segment List

At the bottom middle of the dialog box, the **Key** list shows the key numbers defined in a file. (For 6.x and later files, these key numbers do not have to be consecutive; they can have gaps between them.) The Maintenance tool displays the highlighted key's specifications in the **Key** box at the bottom left of the dialog box.

Also at the bottom middle of the dialog box, the **Segment** list shows the key segment numbers defined for the key highlighted in the **Key** list. The Maintenance tool displays the highlighted segment's specifications in the **Segment** box at the bottom right of the dialog box.

In addition, the following buttons appear under the **Key** and **Segment** lists:

Insert	Defines a new key or segment.
Delete	Removes the highlighted key or segment specification.
Compress	Renumbers the keys consecutively. You can use this button to remove gaps that result from deleting a key specification.

Because these buttons control key specifications for a file you want to create, you cannot use them to operate on keys in an existing file. If you want to create or drop an index on an existing file, refer to Index Tasks.

Key Segment

At the bottom right in the dialog box is the **Key Segment** group box. The next table describes the controls in this area. These controls are specific to the segment highlighted in the **Segment** list),

Control	Description	Default
Data Type	Specifies a data type for the key segment. The NULL data type indicates that the index is one byte Null indicator segment. It must be in a multisegment key and it must precede another key segment that is not a NULL type. The number used in the Btrieve API for this key type is 255.	String
Position	Specifies by number the relative starting position of the beginning of this key segment in the record. The value cannot exceed the record length.	1
Length	Specifies the length (in bytes) of the key segment. This value cannot exceed the limit dictated by the data type for the segment. The total of key position and key length cannot exceed the record length.	10

Control	Description	Default
Null Value (Hex)	Specifies the null character value (in hexadecimal) for the key segment. Applicable only if the **Null Key** check box is selected for the key.	Binary zero
Case Insensitive	Specifies whether the segment is sensitive to case. Applicable only for STRING, LSTRING, and ZSTRING data types or for keys that do not use an ACS.	On
Descending	Specifies that the database engine sort the key segment values in descending order (that is, from highest to lowest).	Off
Use ACS	Specifies that the segment uses the alternate collating sequence defined for the key. Applicable only for `string`, `lstring` and `zstring` data types that are case sensitive.	Off
NULL Value Discrete Ordering	NULL Value Discrete Ordering is used for the null indicator segment (NIS) to determine whether the MicroKernel Engine should treat the NIS as a boolean value, where any non-zero value is considered NULL, or as a one byte integer, where zero is considered non-null and all other values are considered different types of null. In this case they are sorted as discrete values. The Btrieve API uses the NO_CASE flag, 0x0400, to indicate discrete ordering should be performed, because that flag was previously unused for integer values.	Off

Methods for Handling Duplicate Keys

Multiple records may carry the same duplicated value for index keys. The two methods to keep track of the records with duplicate key values are called linked duplicates and repeating duplicates.

Linked Duplicates

The linked duplicates method uses a chain technique in which each record in the group connects to its neighbors by means of pointers. Each entry on an index page contains a pair of record pointers that indicate the first and last links in the chain of records that duplicate that key's value. This makes each key page entry 4 bytes longer than a repeating duplicates index. In addition, each record on the data page requires an extra 8 bytes of overhead for each linked duplicates index. These 8 bytes consist of two record pointers that point to the next and previous records in the chain.

The first record pointer holds the address of the first, or oldest, record stored. The second pointer holds the address of the most recent, or newest record. After the first record is written but before any others are added, both pointers on the key page entry hold the first record's address. Subsequent records cause the second pointer to be changed to point to each record as it is added.

This permits the record pointer for the last record to be used as the previous-record link of the chain built in the data page when the record is added, and also to be used to locate that previous record.

Repeating Duplicates

With the repeating duplicates method, each duplicate key value is stored on both the index page and within the record on the data page. Each key value has only one record pointer instead of two. This method requires no chaining within the data records and saves the 8 bytes of overhead per index within each record. Since the key value is repeated for each duplicate record, the indexes affected increase in size.

Method Comparisons

The linked duplicates and repeating duplicates methods can be compared based on the following criteria:

- Ordering
- Storage
- Performance
- Concurrency

Ordering

A linked duplicates index retrieves duplicates in the order in which they were inserted. A repeating duplicates index retrieves duplicates in the order in which they are located within the file. Since location with a file cannot be controlled, the ordering must be considered as random.

Storage

A linked duplicates index requires 12 more bytes for the first occurrence of each duplicate key value. That includes 8 extra bytes on each record and 4 extra bytes for the key page entry. But each duplicate record requires no additional space in the key page, and adds only 8 bytes per record. Therefore, as the number of duplicates per key value increases, and as the size of the key value increases, linked duplicate indexes can save significant storage space used by key pages. However, storage space can increase if your file contains very few records with duplicate keys, the key length is very short, or both.

The following figure is an example of the amount of storage space saved using linked duplicate indexes. Note that linked duplicate indexes take more space if duplicate records per key value are

few. As the number of duplicate records per key value increases, however, linked duplicate indexes require less pages, providing significant space savings.

Page Counts for
Linked Duplicate vs Repeating Duplicates
Page Size=4096, 100,000 records of length 100
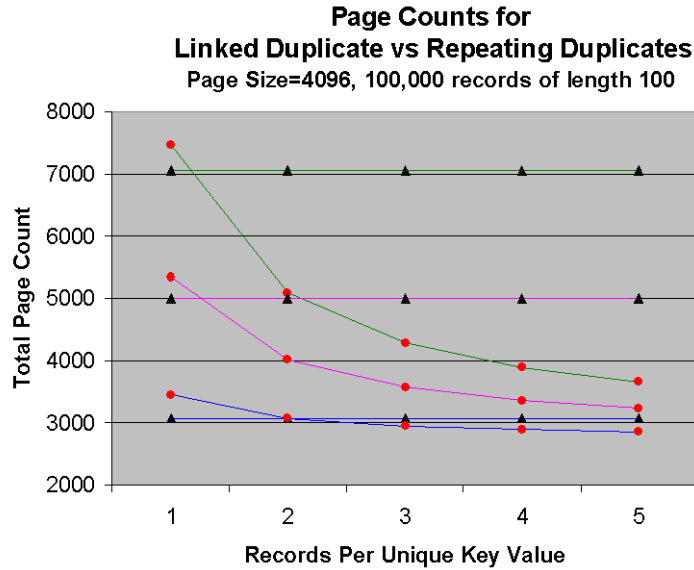
Legend:	● = linked duplicates	▲ = repeating duplicates

Top two lines represent a key length of 100
Middle two lines represent a key length of 50
Bottom two lines represent a key length of 4

Performance

Faster performance results when fewer pages are involved in an index search because fewer pages must be read from disk. The linked duplicates method generally uses less physical storage space and therefore provides faster performance. The repeating duplicates method provides a performance advantage if only a small number of keys have duplicates.

Concurrency

The database engine provides page-level concurrency when several concurrent transactions are active on the same file at the same time. This applies to most changes to key pages and for all changes to data pages. The concurrency means that the same page can contain pending changes from separate transactions at the same time, and the transactions can be committed in any order. Repeating duplicate indexes take the most advantage of this concurrency.

Linked duplicate indexes add another limitation on concurrency that does not exist with repeating duplicates. When a new duplicate is created, the new record is linked to another record at the end of the list. This record linking causes two records to be locked instead of one. Since all duplicates are added to the end of the chain of linked records, only one duplicate can be inserted at a time.

Such a record lock conflict usually causes other clients to wait until the first transaction is committed. In a concurrent environment, if all new records use the same duplicate value, then concurrency can effectively be reduced to one transaction at a time. And if transactions are large or long lasting, this serialization can affect performance tremendously.

Performance is typically better if you use repeating duplicate indexes for databases that are updated in a concurrent environment. Therefore, unless you have a compelling reason to use the linked duplicates method, you should use repeating duplicate indexes for databases that are updated in a concurrent environment.

Information Editor Tasks

You perform the following tasks with the File Information Editor:

- Loading Information from an Existing Data File
- Creating a New File
- Compacting Btrieve Data Files
- Specifying a Alternate Collating Sequence for a Key

Loading Information from an Existing Data File

When you load information from an existing file, you are not editing the existing file. Instead, you are loading a copy of the information about that file. Generally, you want to load a data file before performing other tasks with the File Information Editor, but this is not mandatory.

To load information from an existing data file into the File Information Editor

1. Click **Load Information** at the top of the File Information Editor. The **Select File** dialog box appears.
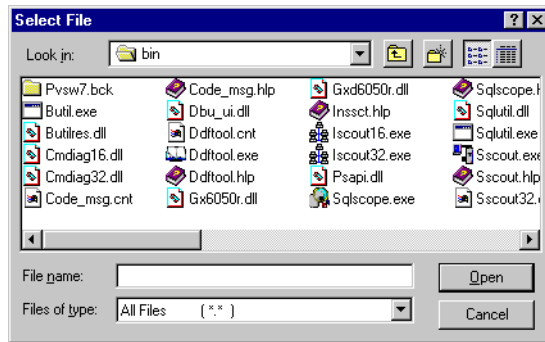
2. Specify the name and path of the file for which you want to load information. By default, data files have the file extension .mkd, but others are possible, or none.

The Maintenance tool first attempts to open the specified file as a data file. If the file requires an owner name, the tool prompts for one. If the specified file is not a data file, the tool then attempts to open the file as a description file.

Creating a New File

You can create a new file based on the current information in the File Information Editor or on new information you provide.

To create a new file based on the current information in the File Information Editor

1. Click **Create File** at the top of the **File Information Editor** dialog box. The **Create File** dialog box appears.
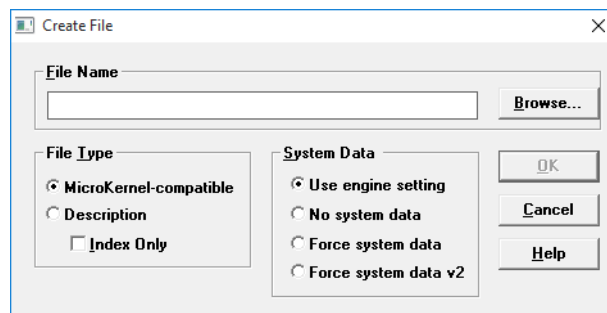
2. Specify the controls in the **Create File** dialog box, which are described in this table:

Control	Description	Default
File Name	Specifies a name and path for the file. By default, data files have the .mkd extension.	N/A
File Type	Specifies the type of file to create. If you are creating a description file, you can use the **Index Only** option, which creates a description file you can use with the **butil** tool to add an index to an existing data file. (For more information, refer to Creating Indexes.)	MicroKernel-compatible
System Data	Determines whether the tool includes system data in the file. If you choose **Use engine setting**, the tool uses the setting for the system data configuration option described. If you choose **No system data**, the tool does not create system data, regardless of the engine configuration. If you choose **Force system data** or **Force system data v2**, the tool creates system data regardless of the engine configuration. System data v2 requires a 13.0 or 16.0 file version.	Use Engine Setting

Adding Comments to a Description File

The comments are written to the top of the description file when you create the description file. For example, the comment, "This is my file," appears at the top of the description files as `/* This is my file */`. If you add additional comments after creating the description file, you need to create the file again to include the additional comments.

To add comments to a description file

1. Click **Description Comments**. The **Description File Comments** dialog box appears.

2. Enter a block of comments up to 5120 characters long.

3. Click **OK** when you are finished entering comments.

Compacting Btrieve Data Files

You can compact a Btrieve data file to remove unused space in it, which typically decreases the file size. You can also perform this procedure using the command line Maintenance tool (see To compact a Btrieve data file).

To compact a Btrieve file

1. Click **Load Information** in the File Information Editor and select the file you want to compact.

2. Click **Create File**, give the file a new name (which creates a clone) in the **Create File** dialog box, and click **OK**.

3. From the **Data** menu on the main window, select **Save**. In the **Save Data** dialog box, enter the name of the original file in the **From MicroKernel File** box and then specify a name for the output file (for example, `<original file>`.out) in the **To Sequential File** box.

4. Click **Execute**. The **Save Data** dialog box displays the results of the save. Click **Close**.

5. From the **Data** menu, select **Load**. In the **Load Data** dialog box, enter the name of the sequential data file you just saved in the **From Sequential File** box. Then enter the name of the clone file you created in Step 2 in the **To MicroKernel File** box.

6. Click **Execute**. The **Loading Data** dialog box displays the results of the load. Click **Close**.

   You can now compare the size of the original file to the clone file to verify the reduction in size.

Specifying a Alternate Collating Sequence for a Key

You can use an alternate collating sequence (ACS) to sort string keys (types STRING, LSTRING, and ZSTRING) differently from the standard ASCII collating sequence. By using one or more collating sequences, you can sort keys as follows:

- By your own user-defined sorting order, which may require a sorting sequence that mixes alphanumeric characters (A-Z, a-z, and 0-9) with non-alphanumeric characters (such as #).

- By an international sorting rule (ISR) that accommodates language-specific collations, including multibyte collating elements, diacritics, and character expansions and contractions.

Files can have a different ACS for each key in the file, but only one ACS per key. Therefore, if the key is segmented, each segment must use either the key's specified ACS or no ACS at all. For a file in which a key has an ACS designated for some segments but not for others, Btrieve sorts only the segments that specify the ACS.

The ISR tables are provided with Zen and are based on ISO-standard locale tables. ISR tables are stored in the collate.cfg file, which is installed with the Zen database engine. Multiple data files can share a single ISR.

To specify a key's alternate collating sequence

1. Click **ACS Information**.

    The Maintenance tool displays the **Specify ACS Information** dialog box.
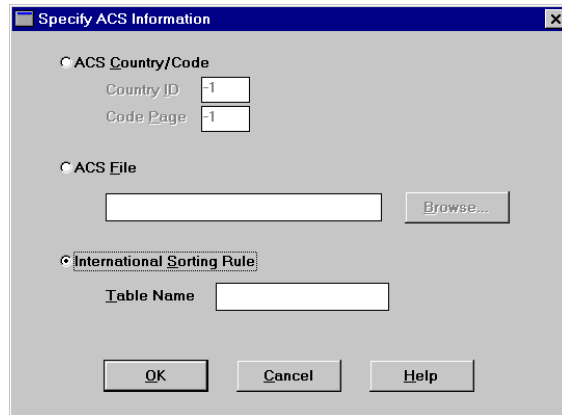
2. You can specify either an ACS File name or an International Sorting Rule (ISR) as follows:

Control	Description	Default
ACS Country/Code	No longer used.	N/A
ACS File	Specifies the fully qualified file name of the alternate collating sequence file.	N/A
International Sorting Rule	Allows you to select an ISR table for sorting international data. Zen provides a set of already generated ISR tables, which are listed in the programmer's guide.	

3. When you specify an ACS file name for a data file, the database engine copies the contents of the ACS file into the data file. (That is, the data file does not contain the file name of the ACS file.) The ACS identifies itself using an eight-digit name (such as UPPER). Subsequently, when you view the ACS information for a data file, the Maintenance tool displays this eight-digit name, not the file name of the original ACS.

4. When you specify an ACS File for a description file, the Maintenance tool copies the actual path and file name of the ACS file into the description file. Subsequently, when you view the ACS information for a description file, the Maintenance tool attempts to locate the specified ACS file.

    To specify an ACS that sorts string values using an ISO-defined, language-specific collating sequence, you must specify an ISR table name. The **Table Name** field is limited to 16

characters. For more information on ISRs, see the *Zen Programmer's Guide* in the Developer Reference.

Managing Owner Names

The MicroKernel gives you the option of restricting access to an individual data file by setting an owner name in the file. Only users who provide the owner name are able to read from or write to the file. This topic covers use of the Maintenance tool to manage owner names. For more information, see the Owner Names topic in *Advanced Operations Guide*.

An ODBC error results if you attempt relational access of a table that is restricted by an owner name without providing that string value, such as trying to delete the table in ZenCC. You can specify owner names for one or more tables in a session with the GRANT or SET OWNER statement. Use the GRANT statement to authorize access for a particular user or group, and then manipulate the table relationally through ODBC. The Master user must supply the GRANT statement with the correct owner name. For more information, see GRANT in *SQL Engine Reference*.

Use SET OWNER to provide one or more owner names to enable file access during the current database session. For more information, see SET OWNER in *SQL Engine Reference*.

Setting or Clearing an Owner Name

Setting an owner name restricts access to a data file. Clearing the owner name removes the restriction.

To set or clear an owner name

1. In the Maintenance tool window, select **Options > Set - Clear Owner** to open the Set - Clear Owner Name dialog.

2. In the **MicroKernel File** box, enter the file for which you want to set or clear an owner name. Then, to clear the owner name, click **Clear Owner** and enter the owner name in the **Current Owner** field to authorize its removal.

3. To set the owner name, click **Set Owner**, enter the new owner name in the **New Owner** field, then select any desired options.

- Select **Permit read-only access without an owner name** to allow all users read-only access to the data file.

- Select **Encrypt data in file** to ensure that unauthorized users do not examine your data using a debugger or file dump tool. Select this option only if data security is important to your environment, since encryption and decryption add to processing time.

- Select **Long Owner Name** to create an owner name longer than the 8-bytes length of short owner names. The length of a long owner name depends on the file format. For more information, see the Owner Names topic in *Advanced Operations Guide*.

4. Click **Execute** to apply the options.

Note: You can also use a GRANT statement to supply an owner name. See Granting Access Using Owner Names in *SQL Engine Reference*.

Statistics Report

Generating a statistics report is a good way to determine whether a file can be logged by the database engine's transaction durability feature. The report shows whether the file has system data and if a key is unique. (A unique key lacks the D flag, which indicates that duplicates are allowed.) The statistics report provides metadata about the file. This information can be used when you troubleshoot problems or to help you create similar files.

Statistics Report Tasks

The following task lists the steps to create a statistics report.

To create a statistics report for an existing data file

1. Click **Options > Create Stat Report** from the menu on the main window. The Maintenance tool displays the **Statistics Report** dialog box.
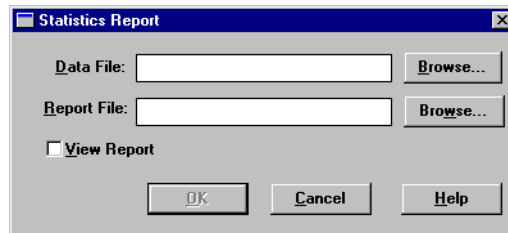
2. Specify a data file to use and a report file name. If you want to view the report when it is created, select the **View Report** check box.

   If you choose to view the report, the Maintenance tool displays a report like the following example in the View File window:

   ```
   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
   ```

   The informational headings in a status report correspond to the controls in the File Information Editor, which is described in File Information Editor.

The legend at the bottom of the statistics report explains the symbols used in the key/segment portion of the report. This information includes items such as the number of keys and key segments, the position of the key in the file, and the length of the key:

```
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
```

Indexes

An *index* is a structure that sorts all the key values for a specific key. Btrieve access permits overlapping indexes (an index that includes a partial column). Relational access through ODBC does not permit overlapping indexes. (You can create an overlapping index with the File Information Editor, which you can display by clicking the Goto Editor button.)

Index Tasks

You perform the following tasks pertaining to indexes:

*   Creating Indexes
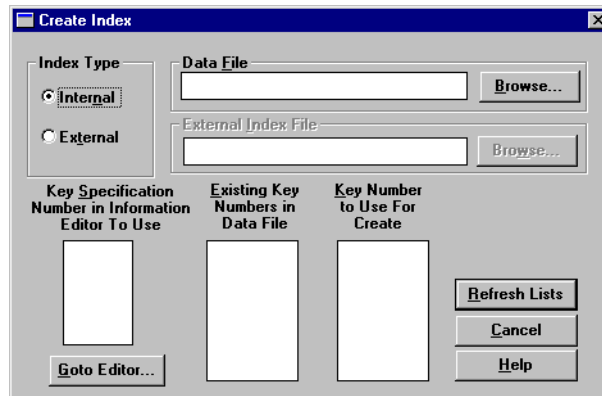*   Dropping Indexes

Creating Indexes

You cannot create an index for a file unless the file has at least one key defined. You can create a key with the File Information Editor (see File Information Editor).

To create an index

1. Click **Index > Create** from the main menu, which opens the **Create Index** dialog box.

2. Complete the following options in the **Create Index** dialog box.

Index Type	Specify whether to create an internal or external index. Internal indexes are dynamically maintained as part of the data file. External indexes are separate files you generate as needed. An external index file is a standard data file that contains records sorted by the key you specify. Each record consists of the following: • A 4-byte address identifying the physical position of the record in the original data file • A key value
Data File	Specify the name of the data file for which you want to create the index.
External Index File	Specify the name of the file to generate for an external index. Not applicable for internal indexes.
Key Specification Number in Information Editor to Use	Lists the key numbers defined in the **File Information Editor**.
Existing Key Numbers in Data File	Click **Refresh Lists** to display the key number defined for the file. If the file contains a system-defined log key, this list includes SYSKEY.
Key Number to Use For Create	Click **Refresh Lists** to display the key numbers available (that is, not defined for the file). Highlight the key number you want to use when creating the index. If the file contains a system-defined log key (also called system data) but the key has been dropped, this list includes SYSKEY, which you can select to add the system-defined log key to the file again.

3. You can click **Go To Editor** to display the **File Information Editor** dialog box, which shows more complete information about the key. You can click **Refresh Lists** to read key information from the data file and refresh the **Existing Key Numbers in Data File** and **Key Number to Use For Create** lists. You must click **Refresh Lists** before you can create an index.

4. When you have completed the **Create Index** dialog box, click **Execute** to create the index. The amount of time required to create the index depends on how much data the file contains.

Dropping Indexes

Ensure that you understand the access performed by an application program before dropping an index. Certain functions fail (such as GET NEXT) if a required index is missing. This can result in an application program not functioning correctly.

To drop an index

1. Click **Index > Drop** from the main menu. The **Drop Index** dialog box appears.

2. Complete the following options in the **Drop Index** dialog box.

MicroKernel File	Specify the name of the data file from which you want to drop the index.
Existing Key Numbers	Click **Refresh List** to display the key number defined for the file. Highlight the number of the key whose index you want to drop. If the file contains a system-defined log key, this list includes **SYSKEY**, which you can select to drop the system-defined log key from the file.
Renumber Keys	Renumbers the keys consecutively. Select this check box to remove gaps that result from deleting an index.

3. Click **Refresh List** to get the key information from the file you have specified.

Data

The commands in the Data menu allow you to import, export, and copy records in data files. You can also recover data after a system failure with the Roll Forward feature.

Importing and Exporting ASCII File Format

When you save data, records in the ASCII file have the following format. You can use an ASCII text editor to create files that you can load, as long as they adhere to these specifications. Note that most text editors do not support editing binary data.

- The first field is a left-adjusted integer (in ASCII) that specifies the length of the record. (When calculating this value, ignore the carriage return/line feed that terminates each line.) The value in this first field matches the record length specified in the data file.

    - For files with fixed-length records, the length you specify should equal the record length of the data file.

    - For files with variable-length records, the length you specify must be at least as long as the fixed-record length of the data file.

- A separator (a comma or a blank) follows the length field.

- The record data follows the separator. The length of the data is the exact number of bytes specified by the length field. If you are creating an import ASCII file using a text editor, pad each record with blank spaces as necessary to fill the record to the appropriate length.

- A carriage return/line feed (0D0A hexadecimal) terminates each line. The Maintenance tool does not insert the carriage return/line feed into the data file.

- The last line in the file must be the end-of-file character (CTRL+Z or 1A hexadecimal). Most text editors automatically insert this character at the end of a file.

Figure shows the correct format for records in the input ASCII file. For this example, the data file has a defined record length of 40 bytes.
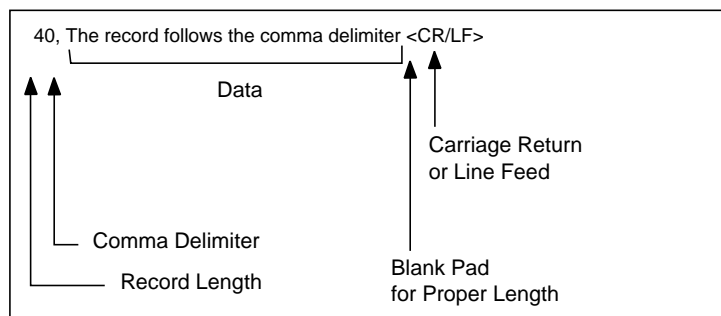
Data Tasks

You can perform the following data tasks with the Maintenance tool:

- To import ASCII data

- To export ASCII records

- To copy records between MicroKernel data files

- To recover (Roll Forward) changes made to a data file between the time of the last backup and a system failure, see Logging, Backup, and Restore in *Advanced Operations Guide*.

Importing Records From an ASCII File

You can use the Maintenance tool to import records from an ASCII file to a standard data file. This operation does not perform any conversions on the data. You can create an import file using a text editor or the Maintenance tool (see Exporting Records to an ASCII File).

To import ASCII data

1. Click **Data > Load** from the main menu. The **Load** dialog box appears.
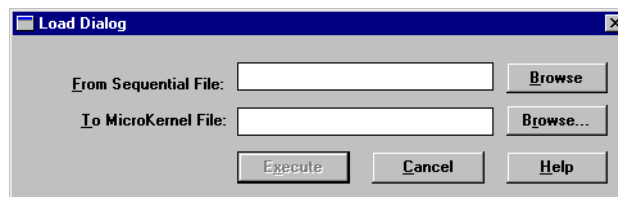
   The ASCII file you specify must adhere to the specifications explained in Importing and Exporting ASCII File Format. The record length of the standard data file you specify must be compatible with the records in the ASCII file.

2. Click **Execute** to import the records.

   While importing data, the Maintenance tool shows the number of records being imported, the percentage of records imported, and a status message. You can continue working in the Maintenance tool (for example, you can open another **Load** dialog box).

Exporting Records to an ASCII File

You can use the Maintenance tool to export records from a data file to an ASCII file.

To export ASCII records

1. Click **Data > Save** from the main menu. The **Save Data** dialog box appears.
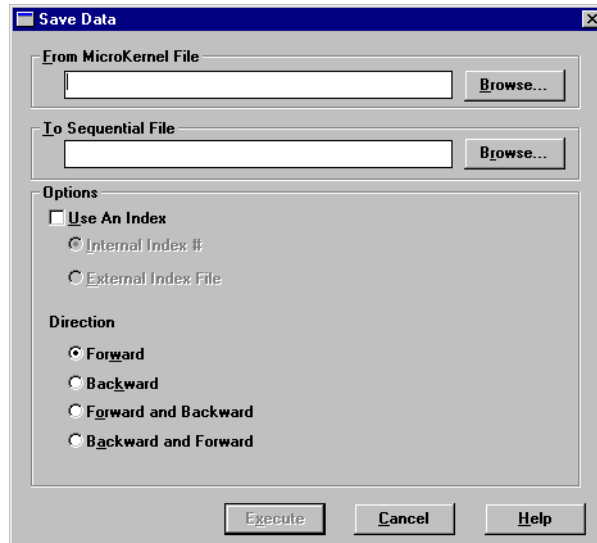
2. In the **Save Data** dialog box, specify the following options.

From MicroKernel File	Specifies the name of the existing MicroKernel-compatible file you want to save.
To Sequential File	Specifies the name of the sequential file to create.
Use An Index	Uses a specified index when sorting the records for export. By default, the Maintenance tool does not use an index, meaning that records are exported according to their physical position in the data file.
	Internal Index #: Uses the specified key number. Click **Refresh Index List** to update the available indexes if you change file in the **From MicroKernel File** box.
	External Index File: Uses the specified external index. (To create an external index, refer to Creating Indexes.)

Direction	Forward: This is the default setting and indicates the tool recovers the file from the beginning.
	Backward: This option recovers data from the end of the file.
	Forward and Backward: This option reads the file forward until it fails. Then it starts at the end of the file and reads the file backward until it reaches the record that failed previously or encounters another failure.
	Backward and Forward: Indicates the tool reads the file backward until it fails. Then it starts at the beginning of the file and reads the file forward until it reaches the record that failed previously or encounters another failure.

3. Click **Execute** to export the data. The Maintenance tool creates the specified ASCII file using the format described in Importing and Exporting ASCII File Format. You can then edit the ASCII file and use the **Load** command to import the edited text to another standard data file

Copying Records Between Data Files

You can use the Maintenance tool to copy data from one MicroKernel data file to another. The record lengths for both data files you specify must be the same.

To copy records between MicroKernel data files

1. Click **Data > Copy** from the main menu. The **Copy Data** dialog box appears.
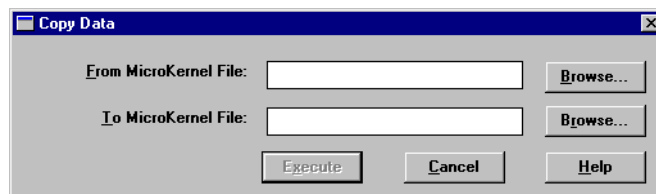
2. Enter the name of the file you want to copy in the **From MicroKernel File** box and then specify the path where you want to copy the file in the **To MicroKernel File** box.

   The record lengths for both data files you specify must be the same.

Recovering (Roll Forward) Changes to a Data File

See Logging, Backup, and Restore in *Advanced Operations Guide*.

Btrieve Command Line Maintenance Tool (Butil)

Use **butil** if you prefer a command line interface or if you want to start or stop continuous operation. Its executable program is **butil.exe** on Windows and **butil** on Linux-based systems. You can manually execute **butil** commands at a command prompt or as an executable batch script. Before you run **butil** commands, we recommend you understand the concepts and syntax covered here.

The Btrieve command line Maintenance tool performs the following file and data manipulations:

- Importing and Exporting Data

- Creating and Modifying Data Files

- Managing the Page Cache for Files

- Viewing Data File Statistics

- Displaying MicroKernel Engine Version

- Unloading the MicroKernel Engine and Requester (DOS only)

- Performing Continuous Operations

- Performing Archival Logging

Return Codes

When **butil** finishes executing, it returns an exit code or DOS error-level return code to the operating system. The return codes are shown in the following table.

Code	Meaning
SUCCESS_E = 0	Requested operation succeeded.
PARTIAL_E = 1	Requested operation completed, but with errors.
INCOMPLETE_E = 2	Requested operation did not complete.
USAGE_E = 3	Syntax error in input, display usage screen and exit.

Commands

The next table lists the commands used with **butil**. Links in the table lead to more detailed information.

Command	Description
Cache	Preloads pages for a file into cache, returning when either the file is fully cached or the cache is full. Counterpart to the Purge command.
Clone	Creates a new, empty data file using specifications for an existing file.
Close	Sends a request to override the File Close Delay configuration option for files kept open by this setting.
Clrowner	Clears the owner name of a data file.
Copy	Copies the contents of one data file to another.
Create	Creates a data file.
Drop	Drops an index.
Endbu	Ends continuous operation on data files defined for backup.
Index	Creates an external index file.
Load	Loads the contents of an unformatted file into a data file.
Purge	Flushes all unneeded cached pages for a file. Returns immediately if the file has open handles. Counterpart to the **cache** command.
Recover	Reads data sequentially from a data file and writes the results to an unformatted file. (The DOS version does not support **rollfwd**.) Use this command if you have a damaged file.
Rollfwd	Recovers changes made to a data file between the time of the last backup and a system failure. See Performing Archival Logging.
Save	Reads data along a key path and writes the results to a sequential file.
Setowner	Assigns an owner name to a data file.
Sindex	Creates an index.
Startbu	Starts continuous operation on files defined for backup. See Logging, Backup, and Restore in *Advanced Operations Guide*.
Stat	Reports statistics about file attributes and current sizes of data files.
Stop (DOS only)	Unloads the MicroKernel Engine and Requester.

Command	Description
Ver	Displays the version of the database engine and requester that is loaded at the server.

Viewing Command Usage Syntax

To view a summary of each command usage, enter the **butil** command at a prompt on the file server.

Command Format

The format for **butil** is as follows:

```
butil [-command [parameter ...]] | @commandFile
```

`-command`	A Maintenance tool command, such as **copy**. You must precede the command with a dash (–), and you must enter a space before the dash.
`parameter`	Information that the command may require. Discussions of the individual commands provide details when applicable.
`@commandFile`	Fully qualified file name of a command file.

Command Files

You can use a command file to do the following:

• Execute a command that is too long to fit on the command line.

• Execute a command that you use often (by entering the command once in the command file and then executing the command file as often as you want).

• Execute a command and write the output to a file, using the following command format:
`butil` @*commandFile [commandOutputFile]*

For each command executed, the resulting output file shows the command followed by its results. All messages appear on the server console screen, as well.

• Execute multiple commands sequentially.

Command files contain the same information as that required on the command line.

Rules for Command Files

Observe the following rules when creating a Maintenance tool command file:

- You cannot split a single parameter across two lines.

- You must end each command with `<end>` or `[end]`. You must also end each command with `<end>` when trying to execute multiple commands. The `<end>` or `[end]` must be lowercase.

Command File Example

The following is an example command file, copycrs.cmd. the file calls the **butil -clone** command to create the newcrs.mkd file by cloning the course.mkd file, and the **-create** command to create the newfile.dta file by using the description provided in the newfiles.des description file.

```
-clone newcrs.mkd course.mkd <end>
-create newfile.dta newfiles.des <end>
```

The following command uses the copypats.cmd file and writes the output to the copypats.out file:

```
butil @copypats.cmd copypats.out
```

Description Files

Description files are ASCII files that contain descriptions of file and key specifications that the Maintenance tool can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. For more information about the description file format, see Description Files in *Advanced Operations Guide*.

Extended File Support

The size of the database engine data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the database engine file size limit because of the differences in the physical format.

When large files are exported, the Interactive Maintenance tool detects when an unformatted file has exceeded the operating system file size limit (2 GB) and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. The extension file uses a naming scheme in which the file names are similar to the base file name. In contrast to native MicroKernel Engine extension files which use a caret "^" to indicate extension file status, the unformatted extension files use a tilde "~" to avoid overwriting any existing extended MicroKernel Engine files with the same base file name. The first export

extension file is the same base file name with ".~01" extension. The second extension file is ".~02," and so on. These extensions are appended in hexadecimal format.

While the naming convention supports up to 255 extension files, the current maximum number of extension files is 64, thus supporting files as large as 128 GB.

To Save or Recover huge files to unformatted files, see the respective command. Also, when you import data from an unformatted file, the tool detects if the file has extensions and loads the data from the extension file.

Owner Names

The MicroKernel allows you to restrict access to individual files by specifying an owner name for the file. The owner name string is also used for encryption.

Redirecting Error Messages

Be sure that you specify a fully qualified file name, including a drive letter or UNC path, when redirecting error messages.

To redirect error messages to a file

•   Use the following command format.

```
butil -command commandParameters > filePath
```

ASCII File Format

See Importing and Exporting ASCII File Format in the Interactive Maintenance tool section.

Rules for Specifying File Names on Different Platforms

When you run **butil** on Windows or Linux, you do not need to specify the name of the path if the data file is in the current directory.

Importing and Exporting Data

This topic provides detailed information on importing and exporting data using the following **butil** commands: **Copy**, **Load**, **Recover**, and **Save**.

Command	Description
Copy	Copies the contents of one data file to another.
Load	Loads the contents of a sequential file into a data file.
Recover	Reads data sequentially from a data file and writes the results to a sequential file.
Save	Reads data along a key path and writes the results to a sequential file.

Note: To export data retrieved by a SQL statement, see the data export utility **deu** in *Zen User's Guide*.

Copy

The **copy** command copies the contents of one MicroKernel file to another. **Copy** retrieves each record in the input data file and inserts it into the output data file. The record size must be the same in both files. After copying the records, **copy** displays the total number of records inserted into the new data file.

Note: **Copy** performs in a single step the same function as a **Recover** command followed by a **Load** command.

Using the **copy** command, you can create a data file that contains data from an old file, but has new key characteristics.

To copy a MicroKernel data file

1. Use the **Create** command to create an empty data file with the desired key characteristics (key position, key length, or duplicate key values).

   or

   Use **Clone** to create an empty data file using the characteristics of an existing file.

2. Use the **copy** command to copy the contents of the existing data file into the newly created data file.

Format

```
butil -copy sourceFile outputFile [/O< owner1 | /PROMPT>
[/O<owner2 | /PROMPT>]] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	The fully qualified name of the data file from which to transfer records. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
outputFile	The fully qualified name of the data file into which to insert records. The output data file can contain data or be empty. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
/Oowner1	The owner name of the source data file, if required. If only the output data file requires an owner name, specify **/O** followed by a blank for *owner1* (as shown in the example). Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/Oowner2	The owner name of the output data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/UID<name> /UIDuname	Specifies the name of the user authorized to access a database with security enabled.
/PWD<word> /PWDpword	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB<name> /DBdbname	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Examples

The following command copies the records in course.mkd to newcrs.mkd. the course.mkd input file does not require an owner name, but the newcrs.mkd output file uses the owner name Pam.

```
butil -copy course.mkd newcrs.mkd /O /OPam
```

If you omit the first /O from this example, the tool assumes that the owner name Pam belongs to the input data file, not the output data file.

When copying a file requires an owner name for both the original file and the copied file, the -copy option specifies both owner names, as shown in this example:

```
butil -copy originalFile copiedFile /Od3ltagamm@ /OV3rs10nXIII
```

The first owner name d3ltagamm@ is required to open originalFile. The second owner name V3rs10nXIII is used to create copiedFile.

If the owner names are provided interactively, the command resembles the following example:

```
butil -copy originalFile copiedFile /PROMPT /PROMPT
```

Upon execution, the user is first prompted to enter the owner name to access *originalFile*. Once that file is open, the user is prompted for the owner name to assign to *copiedFile*.

Load

The **load** command inserts records from an input ASCII file into a file. The input ASCII file can be a single file or an extended file (the base file plus several extension files). **Load** performs no conversion on the data in the input ASCII file. After the tool transfers the records to the data file, it displays the total number of records loaded.

Note: The **load** command opens the output file in Accelerated mode; during a load operation, the database engine does not log the file. If you are using archival logging, back up your data files again after using the **load** command.

Extended files: If the tool finds the next extension file, it continues loading. Do not delete any extension file created earlier by the **save** and **recover** commands. If the file has three extensions and the user deletes the second one, **load** stops loading records after processing the first extension file.

If **save** or **recover** created three extension files and a fourth one exists from a previous **save** or **recover**, **load** reads the records from the fourth extension and inserts them into the database engine file. If a fourth file exists, then you need to delete it before starting the **load** process.

Before running the **load** command, you must create the input ASCII file and the data file. You can create the input ASCII file using a standard text editor or an application. The input ASCII file must have the required file format (see Importing and Exporting ASCII File Format). You can create the data file using either the **Create** or the **Clone** command.

Format

```
butil -load unformattedFile outputFile [/O<owner> | /PROMPT] [/UIDuname /PWDpword [/DBdbname]]
```

unformattedFile	The fully qualified name of the ASCII file containing the records to load into a data file. For Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
outputFile	The fully qualified name of the data file into which to insert the records. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.

/O*owner*	The owner name for the data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Example

The following example loads sequential records from the course.txt file into the course.mkd file. The owner name of the course.mkd file is Sandy.

```
butil -load course.txt course.mkd /OSandy
```

Recover

The **recover** command extracts data from a MicroKernel file and places it in an ASCII file that has the same format as the input ASCII file that the **Load** command uses. This is often useful for extracting some or all of the data from a damaged MicroKernel file. The **recover** command may be able to retrieve many, if not all, of the file's records. You can then use the **load** command to insert the recovered records into a new, undamaged MicroKernel file.

Note: The Maintenance tool performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

Format

```
butil -recover sourceFile unformattedFile [/O<owner> | /PROMPT] [/Q] [/J] [/I]
[/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	The fully qualified name of the data file from which to recover data. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
unformattedFile	The fully qualified name of the ASCII file where the tool should store the recovered records.
/O*owner*	The owner name for the data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.

/Q	Indicates whether to replace an existing unformatted file. By default, the Maintenance tool overwrites the existing files. If you specify this option and a file with the same name exists, the tool returns an error message.
	The tool also checks whether the database engine file to be recovered is extended. If the file is extended, the tool checks for files with the same name as the potential unformatted extension file. If one of those files exists, the tool returns an error message.
/J	Indicates backward reading of the file. If you specify this option, the tool recovers data from the database engine file using step last and previous operations. The default is forward reading, using step first and next operations.
/I	Indicates forward reading of the file. Although the default is forward reading, you can use this option to indicate forward and backward reading. This means that if you specify both **/I** and **/J**, respectively, the tool reads the file forward until it fails. Then it starts at the end of the file and reads backwards until it reaches the record that failed previously or encounters another failure.
	If you specify **/J** first, the tool reads backwards and then reads forward.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

For each record in the source file, if the **recover** command receives a variable page error (Status Code 54), it places all the data it can obtain from the current record in the unformatted file and continues the recovery process.

The tool produces the following messages:

- Informs you about the name of the last extension file created.

- Checks if the next extension file exists, and if so, tells you to delete it.

- If you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files.

Example

The following statement extracts records from course.mkd and writes them to course.txt.

```
butil -recover course.mkd course.txt
```

Save

The **save** command retrieves records from a MicroKernel file using a specified index path and places them in an ASCII file that is compatible with the required format for the **load** command. You can then edit the ASCII file and use the **load** command to store the edited data in another data file. (See Importing and Exporting ASCII File Format for more information about the ASCII file format.)

Save generates a single record in the output ASCII file for each record in the input data file. Upon completion, **save** displays the total number of records saved.

Note: The Maintenance tool performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

Format

```
butil -save sourceFile unformattedFile [Y indexFile | N <keyNumber | -1>] [/O<owner1 | /PROMPT> [/
O<owner2 | /PROMPT>]] [/Q] [/J] [/I] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	The fully qualified name of the data file containing the records to save. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
unformattedFile	The fully qualified name of the ASCII file where you want the tool to store the records.
indexFile	The fully qualified name of an external index file by which to save records *if* you do not want to save records using the default of the lowest key number.
keyNumber	The key number (other than 0) by which to save records *if* you do not want to save records using the default of the lowest key number.
-1	The specification for saving the records in physical order using the Btrieve Step operations.
/O*owner1*	The owner name for the source file, if required. If only the index file requires an owner name, specify **/O** followed by a blank for *owner1*. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/O*owner2*	The owner name for the index file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.

/Q	Indicates whether to replace an existing unformatted file. By default, the Maintenance tool overwrites the existing files. If you specify this option and a file with the same name exists, the tool returns an error message.
	The tool also checks whether the database engine file to be saved is extended. If the file is extended, the tool checks for files with the same name as the potential unformatted extension files. If one of those files exists, the tool returns an error message.
/J	Indicates backward reading of the file. If you specify this option, the tool recovers data from the database engine file using get last and previous operations. The default is forward reading, using get first and next operations.
/I	Indicates forward reading of the file. Although the default is forward reading, you can use this option to indicate forward and backward reading. This means that if you specify both **/I** and **/J**, respectively, the tool reads the file forward until it fails. Then it starts at the end of the file and reads backwards until it reaches the record that failed previously or encounters another failure.
	If you specify **/J** first, the tool reads backwards and then reads forward.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

The tool produces the following messages:

- Informs you about the name of the last extension file created.

- Checks if the next extension file exists, and if so, tells you to delete it.

- If you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files.

Examples

The following two examples illustrate how to use the **save** command to retrieve records from a data file.

This example uses a newcrs.idx external index file to retrieve records from the course.mkd file and store them in an unformatted text file called course.txt:

```
butil save course.mkd course.txt newcrs.idx
```

The following example retrieves records from the course.mkd file using key number 3 and stores them in an unformatted text file called course.txt:

```
butil -save course.mkd course.txt n 3
```

Creating and Modifying Data Files

This topic includes detailed information on creating, modifying, and managing data files using the butil commands shown in the following table. The removal of unused space in a Btrieve data file is discussed under Compacting Btrieve Data Files.

Command	Description
Clone	Creates a new, empty data file using the specifications of an existing file.
Close	Overrides the File Close Delay setting.
Clrowner	Clears the owner name of a data file.
Create	Creates a data file.
Drop	Drops an index.
Index	Creates an external index file.
Setowner	Assigns an owner name to a data file.
Sindex	Creates an index.

Caution! No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not name a data file Invoice.btr and another one Invoice.mkd in the same directory. This restriction applies because the database engine uses the file name for various areas of functionality while ignoring the file name extension. Since only the file name is used to differentiate files, files that differ only in their file name extension look identical to the database engine.

Clone

The **clone** command creates a new, empty file with the same file specifications as an existing file (including any supplemental indexes, but excluding the owner name). The new data file includes all the defined key characteristics (such as key position, key length, or duplicate key values) contained in the existing file.

The **clone** command ignores all MicroKernel configuration options that affect file statistics (such as system data) *except* file version. The **clone** command creates a new file using the database engine file version you specify with the **Create File Version** option.

Format

```
butil -clone outputFile sourceFile [/O<owner> | /PROMPT] [/pagecompresson | /pagecompressoff] [/
recordcompresson | /recordcompressoff] [/UIDuname /PWDpword [/DBdbname]] [/S]
```

outputFile	The fully qualified file name to use for the new, empty data file. When you run **butil** for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
sourceFile	The fully qualified file name of the existing data file to replicate.When you run **butil** for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
/O*owner*	The owner name, if any, for the source data file. Note that an owner name in the source file is not cloned to the output file. If an owner name is needed in the new file, it must be added separately. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/pagecompresson	Turns on page compression for *outputFile* provided that the Create File Version in the compatibility properties for the database engine is 9.5 or later.
/pagecompressoff	Turns off page compression for *outputFile*. This parameter has no effect if *sourceFile* does not use page compression.
/recordcompresson	Turns on record compression for *outputFile*.
/recordcompressoff	Turns off record compression for *outputFile*. This parameter has no effect if *sourceFile* does not contain record compression.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Remarks

Btrieve 6.0 and later allows a maximum of 23 key segments in a data file with a page size of 1024 bytes. Therefore, the **clone** command sets the page size in the new data file to 2048 bytes if the existing data file contains 24 key segments and has a page size of 1024 bytes. This occurs if the existing data file has a format earlier than 6.0 and the database engine was not loaded with the **Create File Version** option set to 5.x or 6.x.

If you are cloning a pre-7.x file, ensure that the database engine is configured to create the file format version that you want the new file to be. For example, if you want to clone a 6.15 file in 9.5 format, in the database engine Compatibility properties, set Create File Version to 9.5.

Note: If your source file is in 8.x, 9.5, 13.0, or 16.0 format and contains no system data, your output file also will contain no system data, regardless of the database engine configuration. To add system data to an existing file, see *Getting Started with Zen.*

If you are trying to recover from receiving status code 30 (file specified is not a MicroKernel file) and you suspect that the header page of the source file might be damaged, try creating the new MicroKernel file using the **Create** command with a description file.

Example

The following command creates the newcrs.mkd file by cloning the course.mkd file.

```
butil -clone newcrs.mkd course.mkd
```

Close

The **close** command sends the database engine a request to override the File Close Delay setting for files being kept open after their last client connection is closed. These open files are then closed immediately. You can issue the command for all files, a single file, or a list of files. Executing **close** with no file parameter applies to all open files awaiting a delayed closing. Executing with no server parameter assumes localhost.

Format

```
butil -close [sourceFile | @listFile] [/Sserver]
```

sourceFile	The fully qualified name of a single file to be closed. When you run butil on Windows platforms, no path is needed if the file is in the current directory.
@listFile	The fully qualified name of a text file containing the fully qualified names of files to be closed. Separate these paths with a new line to put them on their own lines. If butil encounters an error in the list file, it stops processing files in the list.
/Sserver	The name or IP address of a remote server that hosts a database engine. If this parameter is not provided, localhost is assumed.

You can use this command to ensure files are closed before certain actions such as file copying. For more information, see the File Close Delay performance tuning property.

Clrowner

The **clrowner** command clears the owner name of a MicroKernel file.

Format

```
butil -clrowner sourceFile </Oowner | /PROMPT> [/UIDname /PWDword [/DBname]]
```

sourceFile	The fully qualified file name of the data file. When you run **butil** on Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
/Oowner	The owner name to clear. Use of the `/PROMPT` option generates an interactive prompt for an owner name upon execution.
/UIDname	The name of the user authorized to access a database with security enabled.
/PWDword	The password for the user identified by `/UIDname`. If `/UIDname` is given, then `/PWDword` must be supplied.
/DBname	The name of the database on which security is enabled. If omitted, the default database is assumed.

Example

The following command clears the owner name for the file tuition.mkd. The owner name is Sandy.

```
butil -clrowner tuition.mkd /OSandy
```

Create

The **create** command generates an empty MicroKernel file using the characteristics you specify in a description file. Before you can use the **create** command, you must create a description file to specify the new key characteristics. For more information, see Description Files in *Advanced Operations Guide*.

Format

```
butil -create outputFile descriptionFile [< Y | N >] [/UIDuname /PWDpword [/DBdbname]]
```

outputFile	The fully qualified file name of the database engine file to create. If the file name is the name of an existing MicroKernel file, this command creates a new, empty file in place of the existing file. Any data that was stored in the existing file is lost and cannot be recovered. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
descriptionFile	The fully qualified name of the description file containing the specifications for the new MicroKernel file.
Y | N	Indicates whether to replace an existing file. If you specify **N** but a MicroKernel file with the same name exists, the tool returns an error message. The default is **Y**.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Example

The following command creates a file named course.mkd using the description provided in the create.des description file.

```
butil -create course.mkd create.des
```

Sample Description File for the Create Command

The sample description file shown in the following figure creates a MicroKernel formatted file. The file is specified to have a page size of 512 bytes and 2 keys. The fixed-length portion of each record in the file is set to 98 bytes. The file specifies variable-length records with no blank truncation, record compression, and variable-tail allocation tables (VATs). The free space

threshold is set to 20 percent. Allocation is set to 100 pages. The MicroKernel preallocates 100 pages, or 51,200 bytes, when it creates the file.

```
record=98 variable=y truncate=n compress=y       ┐
key=2 page=512 allocation=100 replace=n          ├ File Specification
fthreshold=20 vats=y                             ┘

position=1 length=5 duplicates=y                 ┐
modifiable=n type=string alternate=y             ├ Key 0
nullkey=allsegs value=20 segment=y               ┘ Segment 1

position=6 length=10 duplicates=y                ┐
modifiable=n type=string alternate=y             ├ Key 0
nullkey=allsegs value=20 segment=n               ┘ Segment 2

position=16 length=2 duplicates=n                ┐
modifiable=y type=numeric descending=y           ├ Key 1
nullkey=n segment=n                              ┘

name=c:\myacsfiles\upper.alt
```

Key 0 is a segmented key with two duplicatable, nonmodifiable string segments and a null value of 20 hexadecimal (space) specified for both segments. Key 0 uses the collating sequence upper.alt.

Key 1 is a numeric, nonsegmented key that does not allow duplicates but permits modification. It is sorted in descending order.

Drop

The **drop** command removes an index from a file and adjusts the key numbers of any remaining indexes, subtracting 1 from each subsequent key number. If you do not want to renumber the keys, you can add 128 to the key number you specify to be dropped. This renumbering feature is available only for 6.0 and later files.

Format

```
butil -drop sourceFile < keyNumber | SYSKEY | SYSKEY2> [/Oowner | /PROMPT] [/UIDname /PWDword
[ DBname]]
```

sourceFile	The fully qualified name of the file from which you are dropping the index. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
keyNumber	The number of the key to remove. To preserve the original key numbers, add a 128 bias to the key number you specify.

SYSKEY	Instructs the tool to drop the system-defined log key (also called system data). Dropping the system-defined log key does not delete values from the records; the database engine still assigns unique system-defined log key values to newly inserted records.
	However, the database engine cannot perform logging for a file where the system-defined log key has been dropped if no user-defined unique keys exist. For this reason, you should use this option only if you suspect that the system-defined log key is corrupt and you intend to add it back.
	The Sindex command allows you to reuse the system-defined log key once you have dropped it.
SYSKEY2	Instructs the tool to drop the second system key (124) for system data v2. Dropping this key does not delete system data from the records, and the engine will continue to maintain the values for new and updated records.
	However, without the index, you cannot efficiently find recently modified records. For this reason, use this option only if you suspect that the key is corrupt and you intend to add it back. The Sindex command allows you to reuse the system key once you have dropped it.
/O*owner*	The owner name for the file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/UID*name*	The name of the user authorized to access a database with security enabled.
/PWD*word*	The password for the user identified by *uname*. *Pword* must be supplied if *uname* is used.
/DB*name*	The name of the database on which security is enabled. If omitted, the default database is assumed.

Examples

In both of the following examples, course.mkd has three keys. The original keys in the file are numbered 0, 1, and 2.

In the first example, the **butil -drop** command drops key number 1 from the course.mkd file and renumbers the remaining key numbers as 0 and 1.

```
butil -drop course.mkd 1
```

In the following example, the **butil –drop** command drops key number 1, but does not renumber the keys. The key numbers remain 0 and 2.

```
butil -drop course.mkd 129
```

Index

The **INDEX** command builds an external index file for an existing MicroKernel file, based on a field not previously specified as a key in the existing file. Before you can use the **INDEX** command, you must create a description file to specify the new key characteristics. For more information about description files, see Description Files in *Advanced Operations Guide*.

The records in the new file consist of the following:

*   The 4-byte address of each record in the existing data file.

*   The new key value on which to sort.

Note: If the key length you specify in the description file is 10 bytes, the record length of the external index file is 14 bytes (10 plus the 4-byte address).

Format

```
butil -index sourceFile indexFile descriptionFile [/O<owner> | /PROMPT] [/UIDname /PWDword [DBname]]
```

sourceFile	The fully qualified name of the existing file for which to build an external index. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
indexFile	The fully qualified name of the index file in which the database engine should store the external index.
descriptionFile	The fully qualified name of the description file you have created containing the new key definition. The description file should contain a definition for each segment of the new key.
/O*owner*	The owner name for the data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/UID*name*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*word*	The password for the user identified by *name*. *Word* must be supplied if *name* is used.
/DB*name*	The name of the database on which security is enabled. If omitted, the default database is assumed.

Remarks

The **INDEX** command creates the external index file and then displays the number of records that were indexed. To retrieve the data file's records using the external index file, use the **Save** command.

Sample Description File for the INDEX Command

The sample description file shown in the following illustration defines a new key with one segment. The key begins at byte 30 of the record and is 10 bytes long. It enables duplicates, is modifiable, is a STRING type, and uses no alternate collating sequence.

```
position=30 length=10 duplicates=y modifiable=y
type=string alternate=n segment=n
```

Example

The following command creates an external index file called newcrs.idx using a data file called course.mkd. The course.mkd file does not require an owner name. The description file containing the definition for the new key is called newcrs.des.

```
butil -index course.mkd newcrs.idx newcrs.des
```

Setowner

The **SETOWNER** command sets an owner name for a data file.

Format

```
butil -setowner sourceFile /O<owner | /PROMPT> level [/L] [/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	The fully qualified name of the data file. When you run butil on Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
/Oowner	The owner name to be set. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.

level	The type of access restriction for the data file. The possible values for this parameter are as follows: Note that *level* must immediately follow the /O parameter.
	0: Requires an owner name for any access mode (no data encryption)
	1: Permits read access without an owner name (no data encryption)
	2: Requires an owner name for any access mode (with data encryption)
	3: Permits read access without an owner name (with data encryption)
/L	Designates a long owner name. Owner names are case sensitive and can be short or long. A short owner name can be up to 8 bytes long. The length of a long owner name depends on the file format. For restrictions pertaining to long owner names, see the Procedure topic in *Btrieve API Guide* for Set Owner (29).
/UID<*name*> /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD<*word*> /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB<*name*> /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Examples

The following example creates a short owner for the course.mkd data file. The owner name is Sandy, and the restriction level is 1.

```
butil -setowner course.mkd /OSandy 1
```

The following example creates a long owner name for the billing.mkd data file, encrypts the owner name and file, and restricts all access modes.

```
butil -setowner billing.mkd /Ohr#Admin$945k7YY%svr 2 /L
```

Sindex

The **SINDEX** command creates an additional index for an existing MicroKernel file. By default, the key number of the new index is one higher than the previous highest key number for the data file, or you can instruct the database engine to use a specific key number. An exception is made if a **DROP** command previously removed an index without renumbering the remaining keys, thus producing an unused key number. In this case, the new index receives the first unused number.

You can instruct the database engine to use a specific key number for the new index with the key number option. The key number you specify must be a valid key number that is not yet used in the file. If you specify an invalid key number, you receive status code 6.

If you do not use the **SYSKEY** or **SYSKEY2** option with this command, you must create a description file that defines key specifications for the index before you can use the **SINDEX** command. For more information, see Description Files in *Advanced Operations Guide*.

Format

```
butil -sindex sourceFile <descriptionFile | SYSKEY | SYSKEY2> [keyNumber] [/O<owner> | /PROMPT]
[/UIDuname /PWDpword [/DBdbname]]
```

sourceFile	The fully qualified name of the existing file for which to build an external index. When you run butil for Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
descriptionFile	The fully qualified name of the description file you have created containing the new key definition. The description file should contain a definition for each segment of the new key.
SYSKEY	Instructs the tool to add back the system key (125) on a file in which the system key was dropped.
SYSKEY2	Instructs the tool to add back the system key (124) for system data v2.
/Oowner	The owner name for the data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/UID<name> /UIDuname	Specifies the name of the user authorized to access a database with security enabled.
/PWD<word> /PWDpword	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB<name> /DBdbname	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Examples

The following example adds an index to the course.mkd file. The name of the description file is newidx.des.

```
butil -sindex course.mkd newidx.des
```

The following example adds the system-defined key to the course.mkd file. The system-defined key was dropped.

```
butil -sindex course.mkd syskey
```

Compacting Btrieve Data Files

You can use several commands in the butil (**Clone**, **Recover**, and **Load**, respectively) to remove unused space in a data file to decrease its size.

To compact a Btrieve data file

1. Rename your data file and then use the **Clone** option to create a blank data file using the original file name.

2. Use **Recover** to save the data from the clone file to an unformatted text file in sequential order.

3. Use **Load** to load the recovered data into the clone.

   Every record containing data is loaded into the newly created data file without blank records. You can also perform this operation in the Btrieve Interactive Maintenance tool.

Managing the Page Cache for Files

To improve performance, butil allows you to manage page caching for a file using cache and purge commands.

Notes

When butil -cache and -purge commands are executed from a client cache or reporting engine, their actions apply only to the file in the local cache.

In order for a file to remain in the client cache, one of two things must be true:

- No other machines must try to write to the file while it is closed.

- At least one application must keep the file open on the client machine.

When the client cache engine is installed as an application, it shuts down and empties the cache shortly after the client closes all files. This is not an issue for the client cache engine installed as a service or for the reporting engine.

Cache

The **CACHE** command preloads pages for a file into cache, returning when either the file is fully cached or the cache is full.

Format

```
butil -cache <sourceFile | @listFile>
```

`sourceFile`	The fully qualified name of the data file to be preloaded into cache. For Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
`@listFile`	The fully qualified name of a text file containing the fully qualified names of files to be preloaded into cache. Separate these paths with a new line to put them on their own lines. If butil encounters an error in the list file, it stops processing files in the list.

Purge

The **PURGE** command flushes all unneeded cached pages for a file. Returns immediately if the file has open handles.

Format

```
butil -purge <sourceFile | @listFile>
```

sourceFile	The fully qualified name of the data file to be purged. For Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
@listFile	The fully qualified name of a text file containing the fully qualified names of files to be purged. Separate these paths with a new line to put them on their own lines. If butil encounters an error in the list file, it stops processing files in the list.

Viewing Data File Statistics

This topic includes information about generating a report using STAT on characteristics and statistics of a data file.

Stat

The **STAT** command generates a report of defined characteristics for a data file and statistics about file contents. The **STAT** command does not distinguish between keys defined in a file by the Create Index and Create operations.

Format

```
butil -stat <sourceFile> [/O<owner> | /PROMPT] [/UIDuname /PWDpword [/DBdbname] /JSON]]
```

sourceFile	The fully qualified name of the data file for which to report statistics. For Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
/Oowner	The owner name for the data file, if required. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution.
/PROMPT	Indicates that an interactive prompt will be presented for the user to enter the owner name.
/UID<name> /UIDuname	Specifies the name of the user authorized to access a database with security enabled.
/PWD<word> /PWDpword	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB<name> /DBdbname	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.
/JSON	Returns output in JSON format.

File Statistics and Error Message Formats

Butil -stat statements offer two formats for file statistics and error message output. Text output is the default, while using the /json option returns the same information in JSON format.

The following examples illustrate these options:

- File Statistics as Text
- File Statistics as JSON

- [Error Messages as Text](#)
- [Error Messages as JSON](#)

File Statistics as Text

The following example shows statistics for the file patients.dta as text.

```
butil -stat D:\ClinicDB\patients.dta
```

Which returns the following:

```
**************************************************

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
```

This example shows that the file version of patients.dta is 8.0, indicating the earliest Btrieve version that can read the file format. The file has a page size of 2048 bytes and no preallocated pages. This is not a key-only file, nor an extended file.

Sixteen records have been inserted into the file. The file was defined with a record length of 104 bytes, does not use record compression, and does not allow variable-length records.

The file has no available linked duplicate keys. It does not use balanced indexing. The MicroKernel performs logging using Key 1, and the file contains no system-defined data. It has three keys comprised of four key segments.

Note: Indexes created with Sindex are designated with the letter R by default *unless* you specified the Reserved Duplicate Pointer element.

The statistics report also provides information about specific keys. For example, the report shows that Key 0 consists of two segments, allows duplicates, and is modifiable. Furthermore:

- The first segment starts in position 21, is 20 characters long, allows duplicates, is modifiable, and will be sorted as a string type. The dashes indicate that a null value was not defined. The Unique Values column indicates that 16 unique values were inserted for this segment. This segment uses the upper.alt alternate collating sequence file.

- The second segment starts in position 7, is 12 characters long, allows duplicates, is modifiable, and will be sorted as a string type. Sixteen unique values were inserted for this segment. This segment uses the upper.alt alternate collating sequence file.

Key 1 is the key the database engine uses in logging this file. It consists of one segment. It starts in position 1, is 6 characters long, does not allow duplicates, is modifiable, and will be sorted as a string type. Sixteen unique values were inserted for this key. It uses the upper.alt alternate collating sequence file.

Key 2 consists of one segment. It starts in position 83, is 10 characters long, allows duplicates, is modifiable, and will be sorted as a string type. Seven unique key values were inserted for this key. It uses the upper.alt alternate collating sequence file.

File Statistics as JSON

The following example shows statistics for the file patients.dta as JSON.

```
butil -stat D:\ClinicDB\patients.dta /json
```

Which returns the same content as the default -stat option shown under File Statistics as Text:

```
{

    "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",
```

```json
    "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":"The command completed successfully."
}
```

Error Messages as Text

The following example shows an error message as text.

```
butil -stat D:\ClinicDB\wrongdir\patients.dta
```

Which returns the following errors:

```
Btrieve Maintenance Utility 15.00.034.000

Copyright (C) Actian Corporation 2020

All Rights Reserved.

BUTIL-14: The file that caused the error is D:\ClinicDB\wrongdir\patients.dta.

BUTIL-100: MicroKernel error = 35. The application encountered a directory error.
```

```
BUTIL-9: The command did not complete due to an unrecoverable error.
```

Error Messages as JSON

The following example shows an error message as JSON.

```
butil -stat D:\ClinicDB\wrongdir\patients.dta /json
```

Which returns the same content as the default -stat option shown under Error Messages as Text:

```
{
"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: The file that caused the error is
D:\ClinicDB\wrongdir\patients.dta.",

        "BUTIL-100: MicroKernel error = 35. The application encountered a directory
error."

        ],

    "execution_status":"BUTIL-9: The command did not complete due to an unrecoverable
error."

}
```

System Data v2

If the file contains system data v2, the output contains two additional entries:

```
System Data v2 = Yes
```
```
SYSKEY v2 Status = Present
```

File Version Notes

When asked for the file format version of a file, the database engine gives the earliest engine version that can read the file. For example, if you have a file created in 5.*x* format, it may be reported as a version 3.*x* file because it uses no 4.*x* or 5.*x* features. Starting with the 6.*x* format, the file itself contains a version stamp. Before 6.*x*, the only way to determine file format version was to inspect the features used in the file, since each new file version has features not available in previous ones.

For version 5.*x* and earlier files, the following table shows the features used to determine the file format version.

This file format version is reported as ...	... if one or more of these features are in use
5.x	• Compressed records • Key-only file
4.x	• Extended key types • Variable length records • Key added by Create Index operation
3.x	None of the above

Displaying MicroKernel Engine Version

This section includes detailed information about displaying the version of the MicroKernel Engine using the **VER** command.

Ver

The **VER** command returns the version number of both the MicroKernel Engine and the Requester (the access module).

Format

```
butil -ver
```

Remarks

When you run the **VER** command, the tool displays messages similar to the following:

```
The Btrieve Requester version is 14.00.
```

```
The Btrieve Version is 14.00.
```

Unloading the MicroKernel Engine and Requester (DOS only)

Stop

Use the **STOP** command to unload the MicroKernel Engine and, if applicable, the Requester.

Format

```
butil -stop
```

Performing Continuous Operations

The commands pertaining to continuous operations, **startbu** and **endbu**, are discussed in the topic Logging, Backup, and Restore in *Advanced Operations Guide*.

Performing Archival Logging

The Maintenance tool (GUI or butil command line) provides a way to roll forward archival log files into the data files. See also Logging, Backup, and Restore in *Advanced Operations Guide*.

The **butil rollfwd** command recovers changes made to a data file between the time of the last backup and a system failure. If a system failure occurs, you can restore the backup copy of your data file and then use the butil rollfwd command, which applies all changes stored in the archival log to your restored data files. Do not use this command unless you have restored data files from backup.

Note: You cannot take advantage of the rollfwd command unless you both enable the MicroKernel Archival Logging Selected Files option *and* back up your files *before* a system failure occurs.

You can also use the rollfwd command to produce an output file of logged operations. The rollfwd command can produce the output file either before you roll changes forward or at the same time as the roll forward.

You can roll forward a single file, all data files on a volume, all data files on a drive, or a list of files, volumes, and/or drives.

Using the GUI

1. Access **Maintenance** from the operating system **Start** menu or **Apps** screen or from the **Tools** menu in Zen Control Center.

2. Within the **Maintenance** window, click **Data > Roll Forward** The **Roll Forward** dialog box appears.
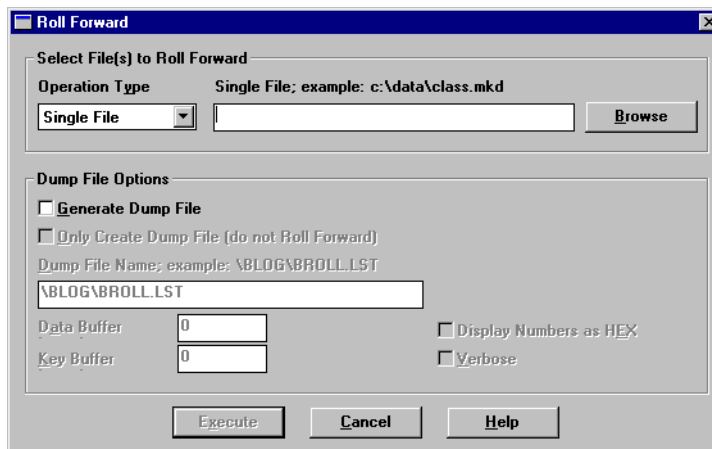
3. Select the specific operation type: single file, list of files, volume name, or drive letter. When you select either volume name or drive letter, you must insert a backslash (\) or slash (/) at the end (for example, \\server\vol1\ or D:\).

4. You can generate a log file, called a dump file, of all the Btrieve operations required to perform the roll forward tasks.

    By default, this file is not created. Select the **Generate Dump File** check box to generate a file. You can also specify the following options.

|	
Only Create Dump File	Indicates that only the dump file is to be created, and the roll forward operation is not to be performed.
Dump File Name	Contains the name of the dump file, which must begin with a slash and not contain a drive letter or server/volume name.
Data Buffer Length	Indicates the number of data buffer bytes to write to the dump file for each Btrieve operation.
Key Buffer Length	Indicates the number of key buffer bytes to write to the dump file for each Btrieve operation.
Display Numbers as HEX	If you select this option, the numbers in the dump file output are formatted as hexadecimal. If you do not select this check box, the numbers are displayed in decimal format.
Verbose	Includes additional information like user name, network address, and time stamp in the dump file.

5. Click **Execute** to generate the dump file and/or perform the roll forward operation. If the data is valid, the **Roll Forward Status** dialog box appears.
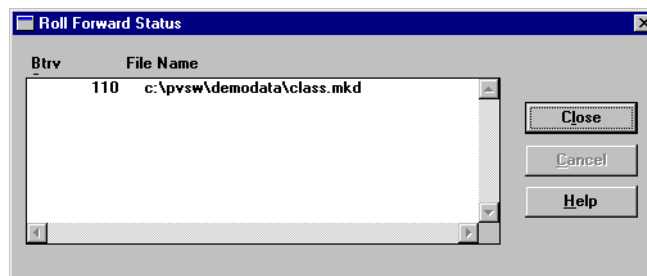
As files are processed, they are added to the scrolling list box which displays the file name and the Zen status code returned from the roll forward operation.

If an error occurs during processing, the **Roll Forward Continue on Error** dialog box appears. This dialog box allows you to continue without being prompted again, to continue and be prompted again, or to stop processing files.
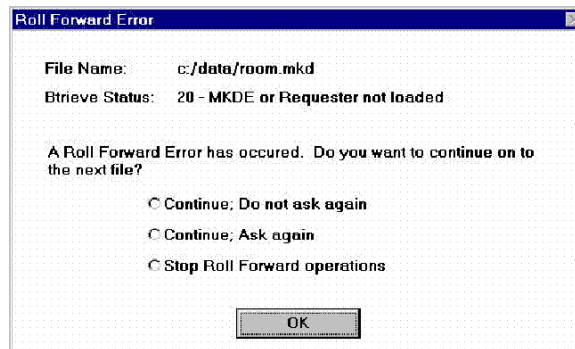
Using the Command Line

This topic explains the syntax for the command line usage of Roll Forward.

```
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	The fully qualified name of a data file for which to roll forward changes. For Windows platforms, you do not need to specify the name of the path if the data file is in the current directory.
drive	A drive letter for which to roll forward changes. End the volume name with a backslash (\) or slash (/), as in F:\ or F:/.
@listFile	The fully qualified name of a text file containing the paths of files, volumes, or drives for which to roll forward changes. Separate these paths with a carriage return/line feed. If the Maintenance tool encounters an error, the tool stops rolling forward the current file, but does not roll back the changes already made. If you specify the **/A** option, the tool continues rolling forward with the next file.
/LdumpFile	Produces an output file, but does not roll forward.
/WdumpFile	Rolls forward and produces an output file.
dumpFile	The file name of the output file to which the Maintenance tool writes a list of logged operations. The default is \blog\broll.lst, relative to the root of the physical drive. The file name cannot contain a drive letter or volume name and must start with a slash (/) or backslash (\). The Maintenance tool places the file on the same volume as the blog.cfg file.

/T*dataLength*	Specifies the length of the operation's data buffer to write to the output file. If you do not specify this option, the tool does not include data buffer contents in the output file.
/E*keyLength*	Specifies the length of the operation's key buffer to write to the output file. If you do not specify this option, the tool does not include key buffer contents in the output file.
/H	Instructs the tool to show numbers in the output file in hexadecimal notation. If you do not specify this option, numbers in the output file are in ASCII format. This option affects the format of the **Entry Count**, **Op Code**, **Key Number**, and **Data Length** fields.
/V	Instructs the tool to include additional information (such as the user name, network address, and time stamp) in the output file.
/O	Specifies the owner name of the data file, if required. An owner name is required if you request an output file of logged operations and the backup copy of the data file has an owner name for read-only access. Use of the /PROMPT option generates an interactive prompt for an owner name upon execution. If more than one file has an owner name, the respective owner names must be separated by commas.
/A	Specifies that if you are rolling back more than one file and the Maintenance tool encounters an error, the tool continues rolling forward with the next file. When you do not specify this option, the tool stops rolling forward if it encounters an error. The tool does not roll back the changes already made. **Note:** When you use the **/A** option, you might want to redirect output to a file, as described in Redirecting Error Messages and Command Files.
/UID*<name>* /UID*uname*	Specifies the name of the user authorized to access a database with security enabled.
/PWD*<word>* /PWD*pword*	Specifies the password for the user who is identified by *uname*. *Pword* must be supplied if *uname* is specified.
/DB*<name>* /DB*dbname*	Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Note: If the key buffer or the data buffer is not an input parameter for a Btrieve operation, nothing is written to the dump file.

Examples

Example A The following example recovers changes to the class.mkd file from the default archival log and log location.

```
butil -rollfwd file_path\Zen\Demodata\class.mkd
```

For default file locations, see Where are the files installed? in *Getting Started with Zen*.

Example B This example recovers changes and outputs them to all files on the d:\ volume with the following options:

- Use default dump file.

- Dump 32 bytes of the data buffer.

- Dump 4 bytes of the key buffer.

- Dump in hex mode.

```
butil -rollfwd d:\ /W /H /T32 /E4
```

Example C The following example does not perform roll forward but only outputs the changes to the files listed in files.txt with the following dump options:

- Use d:\temp\files.lst as the dump file.

- Use verbose mode.

- Data files have owner names: own123 and own321.

- Do not dump data or key buffer.

```
butil -rollfwd d:\temp\files.txt /L\temp\files.lst /V /Oown123,own321
```

Converting Data Files

Zen includes tools that convert Btrieve files to later versions to change file settings or take advantage of features in newer Zen engines, such as system keys. The following topics cover the concepts and tools for doing these conversions:

- Rebuild Tool Concepts
- Rebuild Tool GUI Reference
- Rebuild Tool CLI Reference
- Using the Rebuild Tool

The Zen v16 release includes a standalone command line version of the existing Rebuild tool, called **rbldcli_offline**. It is meant for use in converting Zen data files on systems where Zen is not installed. Rebuilding a large number of files, or very large files, can slow Zen database engine performance for other users. Rebuilding the files on another system avoids this problem. In addition, Zen v16 cannot read files in the 5.x format, so they must be converted to a later version. Only **rbldcli_offline** can read and convert 5.x files.

Note: The Rebuild tool is not intended for use with corrupted files. To repair corrupted files by cloning or by exporting, importing, or copying data, see Manipulating Btrieve Data Files with the Maintenance Tool. To compress files to improve engine performance, see Monitoring Data File Fragmentation.

Rebuild Tool Concepts

The Rebuild tool allows you to perform the following operations on MicroKernel data files and dictionary files:

- Convert older Zen file formats to a newer one.

- Convert newer file formats to a previous one, provided the previous one is not pre-6.x.

- Rebuild a file using the same file format, provided it is not pre-6.x.

- Rebuild files to retain, add, or drop page or record compression.

- Change the page size in the file.

- Rebuild files to retain, add, or drop system data and system keys.

- Specify a location and name of the log file used by Rebuild.

If your database uses dictionary files (DDFs), you must rebuild them as well as the data files.

Read the following topics to understand the concepts behind rebuilding data files.

- Platforms Supported
- File Formats
- Command Syntax
- Temporary Files
- Optimizing the Rebuild Process
- Log File

For information on using the Rebuild tools, see one of these topics:

- Rebuild Tool GUI Reference
- Using the Rebuild Tool
- CLI Rebuild Tasks

Platforms Supported

Rebuild comes in two forms: a 32-bit GUI version for Windows, and command line versions for Linux, Raspbian, and Windows. See Rebuild Tool GUI Reference and CLI Rebuild Tasks.

Linux and Raspbian CLI Rebuild

Rebuild runs as a program, rbldcli, on Linux and Raspbian. By default, the program is located in /usr/local/actianzen/bin.

Windows CLI Rebuild

Rebuild runs as a program, rbldcli.exe, on Windows. By default, the program is installed in the Program Files (x86) directory.

File Formats

The current database engines remain compatible with some older data and dictionary file formats, but you may want to convert files to take advantage of newer features. The following table lists common reasons for converting to a higher format.

Converting to Version	Adds Support for File Features
16.0	Maximum key lengths widened to 1024 bytes to enable indexing of longer data, such as VARCHAR, without having to use PARTIAL INDEX
13.0	• Maximum file sizes in the terabytes • System data v2 for tracking and querying insert and last update time stamps
9.5	• More than 119 segment keys and files sizes up to 256 GB • Long owner names with up to 20 characters instead of 8 • AES-256 file encryption of files with long owner names • 16K page size • Page compression
9.0	Disabling 2GB file segmentation to create one continuous file
8.0	Insert, update, and delete performance improvements offered by Turbo Write Accelerator

The file format that results from using the command line Rebuild depends on the **-f** parameter. If you omit the **-f** parameter, then Rebuild uses the value for the Create File Version setting in the MicroKernel Engine. For example, if the Create File Version value is 9.5, then running the Rebuild tool on version 9.0 files converts them to 9.5 format.

It is suggested that you back up all the data files you plan to convert before running Rebuild. This is particularly true if you are rebuilding files to the same location as the source files (in which case

the rebuilt files replace the source files). Having backup copies allows you to restore the original files if you so desire. To ensure that the backup is successful, you may perform one or more of the following operations:

- Close all data files before running a backup.

- Use continuous operations (only during the backup).

Note: You cannot run Rebuild on a file that is in continuous operation mode.

Temporary Files

On Windows, Rebuild creates temporary files in the directory specified by the TMP system environment variable. By default on Linux and Raspbian, Rebuild creates temporary files in the output directory (or in the source directory if the **-b** option is not used). Therefore, you need enough disk space in the temporary file directory (while Rebuild is running) to potentially accommodate both the original file and the new file. You can specify a different directory for storing these files by using the Output Directory option in the Rebuild GUI version or by using the **-b** option with the CLI versions.

Normally, Rebuild deletes temporary files when the conversion is complete. However, if a power failure or other serious interruption occurs, Rebuild may not delete the temporary files. If this occurs, you should manually delete the temporary files as described in the following table.

Platform	Temporary File Names
Linux and Raspbian	_rbldxxxxxx, where xxxxxx is 6 random characters. **Caution**: Be sure that you do not delete the Rebuild executable, **rbldcli**.
Windows	_rbldx, where x is a number.

Optimizing the Rebuild Process

Rebuild makes Btrieve calls to the database engine. Therefore, the database engine configuration settings and the amount of processing memory affect the performance of the rebuild process, particularly with respect to the amount of time required to rebuild large data files.

In general, building indexes requires much more time than building data pages. If you have a data file with many indexes, it takes more time to rebuild than the same file with fewer indexes.

The following items can affect the rebuild processing time:

- CPU Speed and Disk Speed

- Amount of Memory

- Sort Buffer Size

- Max MicroKernel Memory Usage

- Cache Allocation Size

- Index Page Size

CPU Speed and Disk Speed

The speed of the central processing unit (CPU) and access speed of the physical storage disk can affect processing time during a rebuild. In general, the faster the speed for both of these, the faster the rebuild process. Disk speed is more critical for rebuilding files that are too large to fit entirely in memory.

Tip... Large files in the gigabyte range may take several hours to convert. If you have more than one database engine available, you may wish to share the rebuild processing among a number of machine CPUs. For example, you could copy some of your files to each machine that has a database engine installed, then after the rebuild process is finished, copy the files back to their original locations.

Amount of Memory

Rebuild is capable of rebuilding a file using two different methods, a default method and an alternative method. See **-m<0 | 2>** parameter. For the default method (**-m2**), Rebuild takes the following steps. The more memory available, the faster the execution.

1. Create a new, empty data file with the same record structure and indexes as the source file.

2. Drop all the indexes from the new file.

3. Copy all the data into the new file, without indexes.

4. Add the indexes, using the following process:

   a. For a particular key in the source file, read as many key values as possible into a memory buffer using the Extended Step operation.

   b. Sort the values in the memory buffer and write the sorted values to a temporary file.

   c. Repeat steps a and b, processing the key value from every record.

   The temporary file now contains several key value sets, each individually sorted.

5. Merge the sets into index pages, filling each page to capacity. Each index page is added to the data file at the end, extending the file length.

6. Repeat steps 4 and 5 for each remaining key.

If any failure occurs during this process, such as a failure to open or write the temporary file, Rebuild starts over and uses the alternative method to build the file.

Rebuild uses an alternative method (**-m0**) when memory is insufficient to use the default method or if the default method encounters processing errors:

1. Create a new, empty data file with the same record structure and indexes as the source file.

2. Copy all the data into the new file.

The alternative method is typically much slower than the default method. If you have large data files with many indexes, the difference between the two methods can amount to many hours or even days. The only way to ensure that Rebuild uses the default method is to have enough available memory. Several configuration settings can affect the amount of available memory.

Sort Buffer Size

This setting specifies the maximum amount of memory that the MicroKernel dynamically allocates and deallocates for sorting purposes during run-time creation of indexes.

If the setting is zero (the default), Rebuild calculates a value for optimal memory bytes needed to rebuild indexes in memory.

If the setting is a nonzero value, and the value is larger than the calculated minimum memory that Rebuild needs, then Rebuild uses the smaller value.

Finally, Rebuild compares the amount of memory that it should allocate with 60% of the amount that is actually available. It then attempts to allocate the smaller of the two. If the memory allocation fails, Rebuild keeps attempting to allocate 80% of the last attempted amount. If the memory allocation fails completely (which means the amount of memory is less than the minimum memory bytes), Rebuild uses the alternative method to rebuild the indexes.

Max MicroKernel Memory Usage

This setting specifies the maximum proportion of total physical memory that the MicroKernel is allowed to consume. L1, L2, and all miscellaneous memory usage by the MicroKernel are included. Usage by the Relational Engine is not included.

If you have large files to rebuild, temporarily set Max MicroKernel Memory Usage to a lower percentage than its default setting. Reset it to your preferred percentage after you complete your rebuilding.

Cache Allocation Size

This setting specifies the size of the Level 1 cache that the MicroKernel allocates. The MicroKernel uses this cache when accessing any data files.

This setting determines how much memory is available to the database engine for accessing data files, not for use when indexes are built.

Increasing Cache Allocation to a high value does not help indexes build faster. In fact, it may slow the process by taking up crucial memory that is now unavailable to Rebuild. When rebuilding large files, decrease the cache value to a low value, such as 20% of your current value but not less than 5 MB. This leaves as much memory as possible available for index rebuilding.

Index Page Size

The page size in your file also affects the speed of index building. If Rebuild uses the alternative method, smaller key pages dramatically increase the time required to build indexes. Key page size has a lesser effect on building indexes if Rebuild uses the default method.

Rebuild can optimize page size for application performance or for disk storage.

To optimize page size for performance in terms of speed of data access, Rebuild uses a default page size of 4096 bytes. This results in larger page sizes in physical storage and slower rebuilding times.

For a discussion of optimizing page size for disk storage, see Choosing a Page Size in *Zen Programmer's Guide*.

Log File

Rebuild execution information is appended to a text log file. Its default location varies:

- For **Tools > Rebuild** in ZenCC, it is C:\ProgramData\Actian\Zen\logs\rebuild.log

- For **rbldcli**, it is <current working directory>\rbldcli.log.

- For **rbldcli_offline**, it is <current working directory>\rbldcli_offline.log.

For command line execution, you can set another location and name for the log file instead of using the defaults by using the -lfile parameter.

The information logged includes the following:

- Start time of the rebuild process
- Parameters specified on the command line
- Status code and error description (if an error occurs)
- File being processed
- Information about the processing (such as page size changes)
- Total records processed
- Total indexes rebuilt (if the -m2 processing method is used)
- End time of the rebuild process
- Status of the process (for example, if the file rebuilt successfully)

Rebuild Tool GUI Reference

This topic describes the panels of the Rebuild tool.

File Options

In ZenCC, when you select **Tools > Rebuild**, the Rebuild wizard displays a panel where you can add data or dictionary files to a list.

After creating the list of files to be rebuilt, click **Next** to continue to the Rebuild Options panel.

Rebuild Options

Click an area of the image to learn more about configuring how files can be rebuilt. After you select options, click **Next** to finish rebuilding the selected files.
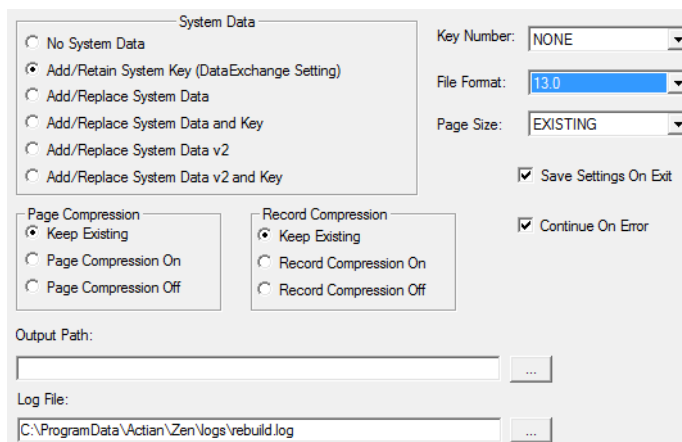
GUI	Description	Related Information
System Data	Specifies whether Rebuild is to retain, add, or drop system data and keys. System data is used for transaction durability logging when no user-defined unique data key exists. System data v2 can be used to track record updates.	To rebuild a file or files

GUI	Description	Related Information
Page Compression	Specifies page compression for the file. The choices are Keep Existing, On, and Off. Keep Existing retains whatever page compression the file contains, if any.	Available only for file format 9.5 or newer. Record and Page Compression
Record Compression	Specifies record compression for the file. The choices are Keep Existing, On, and Off. Keep Existing retains whatever record compression the file contains, if any.	Record and Page Compression.
Continue on Error	Determines whether Rebuild continues if it encounters an error during the rebuild process. If you select Yes, the tool continues with the next file even if an error occurs. The tool alerts you to non-MicroKernel data files or other errors but continues rebuilding data files. If you select No, the tool halts the rebuild if it encounters an error. This option is useful if you use wildcard characters for the rebuilt files.	To rebuild a file or files
Save Settings on Exit	Saves the current values in this dialog box for use in subsequent Rebuild sessions.	To rebuild a file or files
Key Number	Specifies the key by which the tool reads when rebuilding a file. If you select NONE, then Rebuild reads the source file in physical order and creates the rebuilt file in physical order with rebuilt indexes. Because NONE is faster and creates smaller files, use it whenever possible.	• To rebuild a file or files • Key Attributes in *Zen Programmer's Guide*.
File Format	You can select a version other than the default set in the Create File Version of the engine Compatibility property.	To rebuild a file or files
Page Size	Specifies the page size (in bytes) of the new files. Choose either EXISTING, Optimal (disk space), Optimal (data access), or a size in bytes. If you select EXISTING, the tool uses the existing page size. The tool changes the page size if the original size is no longer available or would not work for the other options selected. For example, assume you have a v5.*x* file with a page size of 1024 and 24 keys. Because Btrieve 6.0 and later supports only 23 keys for a page size of 1024, the tool automatically selects a new page size for the file and writes an informative message to the status file.	• To rebuild a file or files • For optimizing for data access, see Optimizing the Rebuild Process • For optimizing for disk space, see Choosing a Page Size in *Zen Programmer's Guide*.

GUI	Description	Related Information
Output Path	Specifies an existing directory for rebuilt files, which can be other than the default of the current directory. This option lets you rebuild large files on a different server. The MicroKernel and its communications components must be loaded on the server that contains the rebuilt files. Do *not* use wildcard characters in the path. If the output directory is not the original file location, the original file is not deleted during the rebuild. If the output directory is the same as the original file, the original file is deleted and replaced with the rebuilt file. When DefaultDB is set to use database security, you can rebuild files only in locations listed in the Directories property for DefaultDB.	To rebuild a file or files
Log File	Specifies a rebuild log file directory, which can be other than the default location of the current working directory. Do *not* use wildcard characters in the path.	• To rebuild a file or files • Log File

Rebuild Tool CLI Reference

Use **rbldcli** if you prefer a command line interface to the Rebuild GUI. Its executable program is **rbldcli.exe** on Windows and **rbldcli** on Linux-based systems. You can manually execute **rbldcli** commands at a command prompt or as an executable batch script. Before you run **rbldcli**, we recommend you understand the concepts and settings covered under Rebuild Tool Concepts.

The rest of this section covers the following topics:

* Command Syntax
* Parameters

Command Syntax

The syntax for a **rbldcli** command is as follows:

```
rbldcli [-parameter ...] file [file] | @cmdFile
```

`parameter`	A *parameter* specifies a setting used with the tool. You may use parameters in any order. Precede each parameter with a hyphen (-). Do not place a space after the hyphen or after the single-letter parameter and the parameter value.
	Note: On Linux and Raspbian, the parameters are case-sensitive.
`file`	The data and dictionary files to convert. If the source files are not in the current working directory, you must include the location for each file, either as a fully qualified or a relative path. You may use the asterisk (*) wildcard character in the file name to select multiple files in a location.
	Note: If the original file contains an owner name, Rebuild applies the same owner name and level to the rebuilt file.
`@commandFile`	Fully qualified file name of a command file for Rebuild to execute. A command file can have multiple entries, with each entry containing command line parameters (if any) and the set of files to convert, followed by <end> or [end].
	When specifying files to rebuild, use full directory path names. You may use the asterisk (*) wildcard character in the file names.
	The following is an example of an **rbldcli** command file:

	`-c d:\mydir\*.* <end>`
	`-c -p1024 e:\dir\*.* <end>`
	`-m0 -k0 d:\db\*.* <end>`

Parameters

The following table lists **rbldcli** parameters. Their behavior is the same as their counterparts described under Rebuild Tool GUI Reference, which may provide additional inforamation.

-c		Continues with the next data or dictionary file if an error occurs. The tool notifies you of non-MicroKernel data files or errors with MicroKernel files, but continues rebuilding data files. The error is written to the log file. See Log File.
		Tip... This parameter is helpful when you use wildcard characters (*.*) for a mixed set of files, which can have a combination of MicroKernel files and non-MicroKernel files. Rebuild reports an error for each non-MicroKernel file, along with errors on MicroKernel files, but continues processing.
-d		Converts pre-6.0 supplemental indexes, which allow duplicates, to 6.x, 7.x, or 8.x indexes with linked-duplicate keys.
		Without -d, Rebuild preserves the indexes as repeating-duplicate keys.
		If you access your data files only through the MicroKernel Engine and your files have a relatively large number of duplicate keys, you can use -d to enhance the performance of the Get Next and Get Previous operations.
-m<0 | 2>		The "m" stands for "method." By default, Rebuild selects -m2 as the processing method. If you select -m0, then Rebuild uses the alternative method -m0 if memory is insufficient.
		See Amount of Memory for how it affects the method chosen.
	0	Rebuilds the data or dictionary file without dropping and replacing indexes. The -m0 method is typically slower than -m2.
		A file created with -m0 has key pages approximately 55% to 65% full, making it more optimized for writing than for reading. If you can afford the extra rebuild time, which can be considerable depending on the situation, you may choose this option based on the needs of your application.
		See also Optimizing the Rebuild Process.
	2	Rebuilds the data or dictionary file by dropping the indexes, copying records to the new file, and rebuilding the indexes. The -m2 method is typically faster and creates smaller files than the -m0 method. It creates a file in which the physical order of records may differ from the original file. A file built using -m2 has key pages 100% full and optimized for reading.

-p<D | P | *bytes*>	Optimizes page size for disk storage or processing, or specifies a specific page size to use for the rebuilt file.
	If -p is omitted, Rebuild uses the page size from the source file. If the source page size is not possible for the new file version, Rebuild changes the page size and displays an informative message explaining the change. For example, older file formats, such as 5.x, supported a page size of 1024 with 24 keys. File format 8.x supports only 23 keys for a page size of 1024, so Rebuild would select a new page size to create an 8.x file.
	The database engine may ignore the page size specified and automatically upgrade the page size to a multiple of 512 bytes, not exceeding the size allowed for the selected file version. For example, for the 9.5 file format, odd page sizes such as 1536 and 3072 are not supported. The database engine automatically upgrades to the next valid page size because that page size is more efficient. For older file formats, the database engine may upgrade the page size based on additional conditions.
	See also Index Page Size.
D	Optimizes page size for disk space efficiency.
	See Choosing a Page Size in *Zen Programmer's Guide*.
P	Optimizes data access for application performance. For -pP, Rebuild uses a default page size of 4096 bytes.
	See Optimizing the Rebuild Process
bytes	Specifies the page size (in bytes) for the new file. For file versions before 9.0, valid values are 512, 1024, 1536, 2048, 2560, 3072, 3584, and 4096. For file version 9.0, the values are the same, with the addition of 8192. For file version 9.5, valid values are 1024, 2048, 4096, 8192, and 16384. For file versions 13.0 and 16.0, valid values are 4096, 8192, and 16384.

-b*directoryname*		The default output location is the current directory of the source files to be rebuilt. Use -b to select another location for the rebuilt file. You may use either a fully qualified or a relative path. Do not use wildcard characters in *directoryname*. *Directoryname* must already exist. Rebuild does not create it.
		Without -b, the rebuilt file replaces the original data file.
		If you use -b to select an alternate directory, the rebuilt file is placed there and the original file is retained. However, if the location already contains data files with the same names, then Rebuild fails and returns an error message to check the log file.
		You must have create file permission for the *directoryname* location, as well as for the source file location if you omit -b.
		In order to work, rbldcli requires that the local database engine be running. If *directoryname* is a location on a remote server, then a database engine must be running there as well.
		Note: The standalone tool **rebldcli_offline.exe** does not require a database engine. Other than this, its capabilities are identical to the original **rbldcli**, except that it cannot output rebuilt files to a location on a remote server.
-k*number*		Selects the key number by which Rebuild reads when rebuilding the source file and sorting the rebuilt file.
		If you omit this parameter, Rebuild reads the source file in physical order and creates the rebuilt file in physical order with rebuilt indexes. Because omitting the parameter is faster and creates smaller files, use it whenever possible.
		See also Optimizing the Rebuild Process.
-s[D | K | 2D | 2K]		Rebuilds the file retaining existing system data and keys from the source file, or adding them if not present. If you omit this parameter, Rebuild does not retain any system data or key in the rebuilt file.
	D	Rebuilds the file with new system data. System data will not be indexed.
	K	Rebuilds the file with new system data. System data will be indexed.
	2D	Rebuilds the file with new system data v2. System data will not be indexed. For 13.0 and 16.0 format files only.
	2K	Rebuilds the file with new system data v2. System data will be indexed. For 13.0 and 16.0 format files only.

-l*file*	Specifies a log file name, and optionally a path location, for the Rebuild log file. The default file name is rbldcli.log on Windows, Linux and Raspbian. The default location is the current working directory.
	The following conditions apply:
	• The path location must already exist. Rebuild does not create it.
	• If you specify a path location without a file name, Rebuild ignores this parameter and uses the default file name and location.
	• If you specify a file name without a path location, Rebuild uses the default location.
	• You must have read and write file permission for the location you specify. Rebuild uses the default location if it cannot create the log file because of file permissions.
	See also Log File.
-pagecompresson	Turns on page compression for the file provided the following conditions are true:
	• The version of the database engine is 9.5 or later.
	• The setting for Create File Version is 0950 (9.5) or later.
-pagecompressoff	Turns off page compression for the file. This parameter has no effect if the file does not use page compression.
-recordcompresson	Turns on record compression for the file.
-recordcompressoff	Turns off record compression for the file. This parameter has no effect the file does not use record compression.

-f<6 | 7 | 8 | 9 | 95 | 13 | 16>	Specifies a file format for the rebuilt data or dictionary file. File formats supported are versions 6.x, 7.x, 8.x, 9.x, 9.5, 13.0, and 16.0. The following example rebuilds a file in 9.0 format:
	
	`rbldcli -f9 file_path\class.mkd`
	
	The following example rebuilds a file in 13.0 format:
	
	`rbldcli -f13 file_path\class.mkd`
	
	If omitted, Rebuild defaults to the value selected in ZenCC for the MicroKernel Create File Version setting in the Compatibility properties for the engine.
	
	Note: If you specify a file format newer than the version supported by the database engine, Rebuild uses the highest supported file format for that engine. It reports no error or message that this has been done.
	
	Note: Rebuild does not convert data types in indexes. If you rebuild a file to an older file format for use with an older database engine, confirm what data types are supported by the database engine and the application.
	
	Example: Your data file contains index fields that use the WZSTRING data type. If you rebuild the data file to a 6.x file format, the WZSTRING data type is not converted. You would be unable to use the data file with a Btrieve 6.15 engine. That engine does not support the WZSTRING data type.
	
	Example: Your data file contains true NULLs. You rebuild the data file to a 7.x file format. The true NULLs are preserved. You would be unable to use the data file with the Zen 7 engine because it does not support true NULLs.

-uid*uname*	Sets the name of the user authorized to access a database with security enabled.
-pwd*pword*	Sets the password for the user identified by *uname*. *Pword* must be supplied if *uname* is specified.
-db*dbname*	Selects the name of the secured database.

Using the Rebuild Tool

The following topics cover the GUI and command line versions of the Rebuild tool:

*   GUI Rebuild Tasks

*   CLI Rebuild Tasks

GUI Rebuild Tasks

*   To start the GUI Rebuild wizard

*   To rebuild a file or files

To start the GUI Rebuild wizard

Select **Tools** > **Rebuild** in the Zen Control Center menu or access **Rebuild** from the operating system **Start** menu or **Apps** screen.

To rebuild a file or files

1. After you click **Next** at the Rebuild welcome screen, the **Select Files** screen appears.

2. Click **Add** and in the Selected Files dialog box, select the data or dictionary file you want to rebuild. You can select more than one file to rebuild at a time.

    Rebuild deletes the original file after rebuilding it if the file is being rebuilt in the same directory. If the new file is in a different directory, the original file is not deleted.

3. Click **Next** after you have added the desired file or files.

4. Specify the rebuild options. See Rebuild Options.

5. Click **Next** to begin the rebuild process.

    Rebuild reports the processing information. When the rebuild process finishes, its success or failure is displayed and **View Log File** is enabled.

6. To display the results, click **View Log File**. The contents of the log file display in the default text editor for the operating system.

    Rebuild writes to the log file for every file it attempts to convert. If you disabled the **Continue on Error** setting, the log file contains the information up to the point of the error. If the rebuild was not successful, the status file contains error messages explaining why the rebuild failed.

7. Click **Finish** when you have finished rebuilding files and viewing the log file.

CLI Rebuild Tasks

The Rebuild command line tool is named **rbldcli.exe** on Windows and **rbldcli** on Linux and Raspbian. The following topics cover command line syntax and typical Rebuild tasks:

- To run Rebuild on Windows
- To run Rebuild on Linux and Raspbian
- To see your progress while rebuilding files

To run Rebuild on Windows

1. Open a command prompt on the system where Zen is installed.

2. Optionally, change to the \bin directory where you installed the program files. This is not required if the location is in the Path system variable.

3. Enter one of the following commands at the prompt:

   ```
   rbldcli [-parameter ...] file
   ```

   or

   ```
   rbldcli @command_file
   ```

Parameter, *file*, and *@command_file* are defined in Command Syntax.

Example Usage

The following example continues on error, sets a page size of 4096 bytes, and places the rebuilt files in a different directory on the server.

```
rbldcli -c -p4096 -bc:\dbtemp c:\datafiles\*.mkd
```

To run Rebuild on Linux and Raspbian

1. Ensure that the account under which you are logged in has permission to run Zen utilities.

   By default, you must be logged in as user zen-svc to run utilities. User zen-svc has no password and can be accessed only through the root account by using the su command. To use utilities from accounts other than zen-svc, you must first make modifications to the .bash_profile. See Zen Account Management on Linux and Raspbian in *Getting Started with Zen*.

2. Change directory to `/usr/local/actianzen/bin` directory.

3. Type one of the following commands at the prompt:

```
rbldcli [-parameter ...] file
```

or

```
rbldcli @command_file
```

Parameter, *file*, and *@command_file* are defined in Command Syntax.

Example Usage

The following example continues on error, sets a page size of 4096 bytes, and places the rebuilt files in a different directory on the server.

```
rbldcli -c -p4096 -b/usr/local/actianzen/tmp /usr/local/actianzen/data/Demodata/*.mkd
```

To see your progress while rebuilding files

Rebuild reports on the screen the number of records processed per file, incrementing 50 records at a time. In addition, Rebuild writes information to the log file. See Log File.

Description Files

A *description file* is an ASCII text file that contains descriptions of file and key specifications that the Maintenance tool can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. Description files are not the same as DDFs, or Data Dictionary Files, used by the Relational Engine.

Description files contain one or more elements. An element consists of a keyword, followed by an equal sign (=), followed by a value (with no space). Each element in a description file corresponds to a particular characteristic of a data file or key specification.

Note: Before using description files, you should be familiar with Btrieve fundamentals. For information about these topics, see *Zen Programmer's Guide*.

This appendix discusses the following topics:

- Rules for Description Files
- Description File Examples
- Description File Elements

Rules for Description Files

Use the following rules when creating a description file.

- Enter elements in either uppercase or lowercase.

- Separate elements from each other with a separator (blank space, tab, or carriage return/line feed), as in the following example:

```
record=4000
key=24
```

- Specify the description file elements in the proper order. The table under Description File Elements presents the elements in the appropriate order.

- Address all element dependencies. For example, if you specify `nullkey=allsegs` in your description file, you must also specify a value for the `value=` element.

- Define as many keys as you specify with the **Key Count** element. For example, if you specify `key=12`, you must define 12 keys in the description file.

- For a key that consists of multiple segments, you must define the following elements for each key segment:

  - Key Position

  - Key Length

  - Duplicate Key Values

  - Modifiable Key Values

  - Key Type

  The **Descending Sort Order** element is optional for each segment.

- If any key in the file uses an ACS, you must specify an ACS file name or an ISR table name. You can include this information as either the last element of the key (applies to current key only) or the last element in the description file (applies to entire data file).

  - You can specify only one ACS per key, and you must provide an ACS file name or ISR table name. Different keys in the same file can use different types of collating sequences. For example, Key 0 can use an ACS file name, and Key 1 can use an ISR table name.

  - Different segments of the same key cannot have different collating sequences.

  - If you specify an ACS at the end of a description file, it is used as the default ACS. That is, if you specify `alternate=y` for a given key but do not include an ACS file name or ISR table name for that key, the database engine uses the ACS file name or ISR table name specified at the end of the file.

- If you are creating a new key and you specify `alternate=y` but you omit the ACS file name or ISR table name, the database engine does not create the key.

- If a **Description File** element is optional, you can omit it.

- Make sure the description file contains no text formatting characters. Some word processors embed formatting characters in a text file.

Description File Examples

The sample description files shown in this section describe a data file. This data file has a page size of 512 bytes and 2 keys. The fixed-length portion of the record is 98 bytes long. The file allows variable-length records but does not use blank truncation.

The file uses record compression, allows for Variable-tail Allocation Tables (VATs), and has the free space threshold set to 20 percent. The MicroKernel engine preallocates 100 pages, or 51,200 bytes, when it creates the file. The file has two keys: Key 0 and Key 1. Key 0 is a segmented key with two segments.

In this figure, both keys use the same ACS file name (upper.alt).

```
record=98 variable=y truncate=n compress=y      File
key=2 page=512 allocation=100 replace=n         Specifications
fthreshold=20 vats=y

position=1 length=5 duplicates=y                Key 0
modifiable=n type=string alternate=y            Segment 1
nullkey=allsegs value=20 segment=y

position=6 length=10 duplicates=y               Key 0
modifiable=n type=string alternate=y            Segment 2
nullkey=allsegs value=20 segment=n

position=16 length=2 duplicates=n
modifiable=y type=numeric descending=y          Key 1
nullkey=n segment=n

name=c:\myacsfiles\upper.alt
```

In the next figure, Key 0 and Key 1 use different ACS file names (lower.alt and upper.alt, respectively).

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y
```
File
Specifications

```
position=1 length=5 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y
name=sys:\zen\demodata\lower.alt
```
Key 0
Segment 1

```
position=6 length=10 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n
name=c:\myacsfiles\lower.alt
```
Key 0
Segment 2

```
position=16 length=2 duplicates=n
modifiable=y type=numeric descending=y
nullkey=n segment=n
name=c:\myacsfiles\upper.alt
```
Key 1

In this third figure, the file has no keys except the system-defined key used for logging.

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y sysdataonrecord=loggable
```

Description File Elements

Description file elements must appear in a particular order. The following table lists the description file elements in the appropriate order. For each element, the table specifies the required format and the range of acceptable values.

- An asterisk (*) indicates that the element is optional.

- A pound sign (#) indicates that it is not applicable in the current MicroKernel version but is retained for backward compatibility with previous MicroKernel versions.

- A percent sign (%) indicates that the element is applicable only to the current MicroKernel version.

Element	Keyword and Format	Range	Comments
File Specification Information			
Comment Block*	/*. . . . . */	5120 bytes	None.
Record Length	record=*nnnn*	4 to page size limits	None.
Variable-Length Records	variable=<y|n>	N/A	Not applicable to key-only files.
Reserved Duplicate Pointer*	dupkey=<*nnn*>	0–119	Applicable only to files for which you plan to add linked-duplicatable keys.
Blank Truncation*	truncate=<y|n>	N/A	Not applicable for files that use record compression.
Record Compression*	compress=<y | n>	N/A	Not applicable to key-only files. See also Record and Page Compression.
Key Count	key=*nnn*	0–119	Specify 0 to create a data-only file. If key count is 0, then Include Data and Use System Data cannot be set to no.

Element	Keyword and Format	Range	Comments
Page Size	page=*nnnn*	• 512–4096 bytes for file versions prior to 9.0 (a multiple of 512 bytes up to 4096) • 512, 1024, 1536, 2048, 2560, 3072, 3584, 4096, or 8192 bytes for file version 9.0. • 1024, 2048, 4096, 8192, or 16384 bytes for file versions 9.5. • 4096, 8192, or 16384 bytes for file versions 13.0 and 16.0.	
Page Preallocation*	allocation=*nnnnn*	1–65535	None.
Replace Existing File*#	replace=<y|n>	N/A	None.
Include Data*	data=<y|n>	N/A	Specify *n* to create a key-only file. Cannot create a key-only file that both allows duplicates and uses a system-defined key.
Free Space Threshold*	fthreshold=<5|10|20|30>	N/A	Applicable only for files that have variable-length records. The default is 5.
Variable-Tail Allocation Tables (VATs)	huge=<y|n> # vats=<y|n>	N/A	Applicable only for files that have variable-length records.
Balanced Index*	balance=<y|n>	N/A	None.
Use Key Number*	usekeynum=<y|n>	N/A	Used with the Key Number element.

Element	Keyword and Format	Range	Comments
[1]Use System Data*%	sysdataonrecord=<n|loggable>	N/A	If no element specified, MicroKernel configuration is used. If creating a key-only file, MicroKernel configuration is used and this element is ignored. Also, you cannot create a key-only file that both allows duplicates and uses a system-defined key.
[1]Use System Data v2*	sysdata2=<n|y>	N/A	Add system data v2 to the file. Not for use with a key-only file. Requires a 13.0 or higher format file.
Page Compression*	pagecompress=<y|n>	N/A	See also Record and Page Compression.
File Version*	version=0x<HH>	N/A	If no element is specified, the system default file version is used. If the element is specified, use hexadecimal digits as in these examples: • 0x90 for version 9.0 • 0x95 for version 9.5 • 0xD0 for version 13.0 • 0xD3 for version 16.0
Key Specification Information			
Key Number*	keynum=*nnn*	0–118	Must be unique to the file, specified in ascending order, and valid for the page size of the file. Applicable only when creating a file.
Key Position	position=*nnnn*	1 to record length	Cannot exceed the record length.

Element	Keyword and Format	Range	Comments
Key Length	length=*nnn*	key type limit	Cannot exceed the limit dictated by the Key Type. For binary keys, the key length must be an even number. The total of the Key Position and Key Length cannot exceed the file's Record Length.
Duplicate Key Values	duplicates=<y|n>	N/A	Cannot create a key-only file that allows duplicates and uses a system-defined key.
Modifiable Key Values	modifiable=<y|n>	N/A	None.
Key Type	type= *validMKDEKeyType*	N/A	Can enter the entire word (as in float) or just the first three letters (as in flo).
Descending Sort Order*	descending=<y|n>	N/A	None.
Alternate Collating Sequence	alternate=<y|n>	N/A	Applicable only for case sensitive STRING, LSTRING, WSTRING, WZSTRING, or ZSTRING keys. When creating an additional index for an existing file, if you want the index to use an ACS other than the first one in the data file, use with `caseinsensitive=y`.
Case-Insensitive Key*	caseinsensitive=<y|n>	N/A	Applicable only for STRING, LSTRING, or ZSTRING keys that do not use an ACS.

Element	Keyword and Format	Range	Comments
Repeating Duplicates*	repeatdup=<y|n>	N/A	If creating a key-only file, use repeating duplicates. If using this element, you must use duplicates=y.
Null Segments*	nullkey=<allsegs | n | anyseg |>	N/A	None.
Null Key Value	value=*nn*	1-byte hex	Used with the Null Segments element.
Segmented Key	segment=<y|n>	N/A	None.
Alternate Collating Sequence	name=*sequenceFile* or *isr=table name (%)*	valid path *or* values valid to operating system or -1	Used with the Alternate Collating Sequence element.

[1]When the database engine adds system data, the resulting records may be too large to fit in the file's existing page size. In such cases, the database engine automatically increases the page size to the next accommodating size.