ACTIAN™

Distributed Tuning Objects Guide

Zen v15

Activate Your Data™

Contents

About This Document

This documentation covers the developing of applications using the Distributed Tuning Objects.

Who Should Read This Manual

This document is designed for any user who is familiar with Zen and wants to develop administrative applications using the Distributed Tuning Objects.

We would appreciate your comments and suggestions about this document. Your feedback can determine what we write about the use of our products and how we deliver information to you. Please post your feedback in the community forum on the Zen website.

Distributed Tuning Objects Introduction

The following topics introduce the functions that comprise the Zen Distributed Tuning Objects:

- What is DTO?
- DTO Objects Model and Objects Relationship
- Getting Started with DTO
- DTO Object Summary
- Working with DTO Collections
- Where to Find DTO Samples

You can also go directly to more detailed information on using DTO in Zen:

- Establishing a DTO Session
- Configuring Zen Servers with DTO
- Monitoring Zen Servers with DTO
- Creating and Maintaining Catalogs and Dictionaries with DTO
- Distributed Tuning Objects Enumerations

What is DTO?

Distributed Tuning Objects (DTO throughout the rest of this document) are the COM wrapper for the Zen Distributed Tuning Interface (DTI throughout the rest of this document). DTO is a collection of objects encapsulating DTI. DTO also contains some features that go beyond the capabilities of DTI such as the ability to start and stop the database engines.

DTO enables developers to develop a range of useful, customized server administration tools and interfaces quickly and easily. The power and flexibility of DTO can be applied to the full range of database management and database definition tasks such as production, performance tuning, and metadata administration.

DTO is implemented as a dual interface, in-process server. The developer can use any OLE Automation controller, or create a COM client using many programming languages such as:

- Microsoft Visual Basic
- Microsoft Active Server Pages (ASP)
- Microsoft Visual C++

- Embarcadero Delphi
- Embarcadero C++ Builder

DTO Objects Model and Objects Relationship

For the purpose of this manual, DTO classes are arranged in the following functional categories:

Connection

In order to be able to configure and monitor the behavior of the database engine, users musts first connect to it. This category provides functionality necessary to connect to and disconnect from database engines.

The DtoSession object manages the connection to a database engine.

Monitoring and Diagnostic

This category provides functionality to monitor Zen servers and clients and provides diagnostic information.

The DtoMonitor object and its descendents provide server monitoring and diagnostic information. You can also obtain engine information directly from DtoSession with the DtoEngineInformation object.

Configuration

This category allows the user to configure the Zen engines and clients. The DtoCategories collection and its descendents provide this functionality.

You can also add and remove product licenses using the DtoLicenseMgr object.

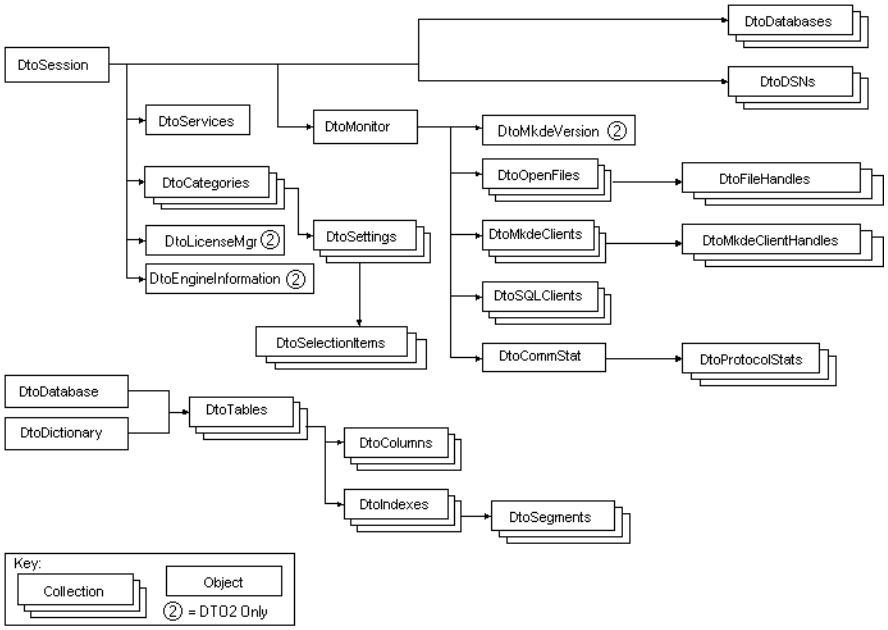
Catalog and Dictionary

Functionality grouped in this category allows users to create new databases, new data dictionaries and also define and drop tables, columns, and indexes.

The DtoDictionary class and its descendents provide catalog functionality.

DTO Object Tree

Many DTO objects are exposed as properties of other DTO objects. The relationship provides developers with a logical structure that simplifies programming with automation controllers. You can reference objects using the familiar dot notation to access properties and methods.



The DTO object tree contains three main branches, logically grouping objects for Configuration, Monitor and Catalog.

DTO Versions

The following table shows usage information regarding the two versions of DTO:

Item	DTO2		DTO1
	x86	x64	x86
DLL name	DTO2.DLL	w64DTO2.DLL	DTO.DLL
Library	DTOLib2		DTOLib
ProgID of DtoSession	DTO.DtoSession.2		DTO.DtoSession.1
ProgID of DtoDatabase	DTO.DtoDatabase.2		DTO.DtoDatabase.1
ProgID of DtoDictionary	DTO.DtoDictionary.2		DTO.DtoDictionary.1

DTO2

A new DLL was released with Zen V8 SDK that added objects and new properties to existing objects. In order to maintain backward compatibility, a new DLL was created instead of adding this to the previous DTO.DLL. Both are installed and registered with Zen SDK and you can use either one to develop DTO applications. Using the previous DTO.DLL may be important to you if you cannot recompile your application to use the new DLL. However, if you continue to use the previous DTO.DLL, you cannot use the new objects or some new properties that were added to existing objects. DTO2.DLL supports 32-bit development environments, including the .NET framework.

W64DTO2

The Zen v11 SP1 SDK added 64-bit support for 64-bit environments, including the .NET framework.

In order to utilize DTO for 64-bit applications, you will need to install the 64-bit server or client with your 64-bit application. Installing the 64-bit server or client installs the 64-bit version (W64DTO2.DLL), whereas installing the 32-bit server or client installs the 32-bit version (DTO2.DLL or DTO.DLL), as described in the previous table.

Understanding How Applications and DLLs Interact

To better understand how applications and DLLs interact, we will examine the following scenarios.

Assuming we have the following three DLLs:

- DTO.DLL
- DTO2.DLL
- W64DTO2.DLL

And the following application executables:

- ANYCPU.EXE
- X86.EXE
- X64.EXE

The following tables illustrate the results when you try and run the application executables and DLLs together, on 32- and 64-bit machines.

32-bit Machine Process

Application Executable	Runs as	DTO.DLL	DTO2.DLL	W64DTO2.DLL
ANYCPU.EXE	32-bit process	Loads	Loads	BadImageFormat Exception
X86.EXE	32-bit process	Loads	Loads	BadImageFormat Exception
X64.EXE	BadImageFormat Exception			

64-bit Machine Process

Application Executable	Runs as	DTO.DLL	DTO2.DLL	W64DTO2.DLL
ANYCPU.EXE	64-bit process	BadImageFormat Exception	BadImageFormat Exception	Loads
X86.EXE	32-bit process	Loads	Loads	BadImageFormat Exception
X64.EXE	64-bit process	BadImageFormat Exception	BadImageFormat Exception	Loads

Getting Started with DTO

This topic describes how to set up DTO for use in Visual Basic, Active Server Pages (ASP), and Delphi.

- Visual Basic
- Active Server Pages
- Delphi

Visual Basic

Since DTO is a library of dual interface COM objects, there are two ways of working with these objects in Visual Basic. If you are using Active Server Pages, you must use the second method. The first and preferred method is to add the type library to the project. This method allows VB to do type checking and provides the developer with useful drop down options for object creation and function parameters (Intellisense).

The other method is to use the CreateObject function. This creates objects at run time and therefore lacks type checking and the Intellisense feature.

Adding a Reference to DTO to a Project

To add the Distributed Tuning Library to a Visual Basic project:

1. Select **References** from the Project menu.
2. Scroll through the available entries and select the **Pervasive Distributed Tuning Library 1.0** or **Pervasive Distributed Tuning Library 2.0** check boxes. For more information on the difference, see DTO2.
3. Click **OK.**

VB is now aware of all the objects included in DTO. All the objects are now browsable. To view the available objects:

1. Select **Object Browser** from the **View** menu. Alternatively, you can press **F2**.
2. Select **DTOLib** or **DTOLib2** from the list of available libraries depending if you want DTO Version 1 or 2, respectively. For more information on the difference, see DTO2.

Using the CreateObject Function

You need to use this method when instantiating an object with ASP. The CreateObject syntax looks like this:

```
Dim my_session as Object
' For DTO Version 2
Set my_session = CreateObject("DTO.DtoSession.2")
' or use DTO Version 1 for compatibility with
' previous DTO applications
Set my_session = CreateObject("DTO.DtoSession.1")
```

Most of the DTO objects can later be obtained from the session object.

Active Server Pages

There is no special initialization required to use DTO with ASP. However, note the following:

ASP by default does not save state information between calls. You will need to use the Microsoft IIS built-in Session object to preserve object references and variable state.

For example, to initialize a DtoSession object with DTO Version 2:

```
Set Session("my_session") = Server.CreateObject("DTO.DtoSession.2")
```

Delphi

COM objects in Delphi are used in two ways. As with Visual Basic, the first and preferred setup method is to import the type library into the Delphi project. The other method allows COM interfaces to be called directly by using CreateOleObject function. This function will instantiate a single instance of an Automation object.

Importing the DTO Type Library into a Delphi project

In order to import the type library and generate the necessary Pascal declarations:

1. Select **Project** | **Import Type Library**.
2. The **Import Type Library** dialog box displays the type libraries registered on the system. Choose **Pervasive Distributed Tuning Library 1.0** or **Pervasive Distributed Tuning Library 2.0**
3. Enter a location of pascal unit in Unit directory name and press **Create Unit**. The file DTOLib_TLB.pas or DTOLib2_TLB.pasfile will be created and included in the project, depending on whether you use DTO version 1 or 2.
4. Then include the generated pascal unit to the main file by including the following line:

```
   uses DTOLib2_TLB; // Dto Version 2
   uses DTOLib_TLB;  // Dto Version 1
```

Example of Using Pascal Declarations

```
var
        Result:DTOResult;
        Session:DTOSession;
        MySettings:DTOSettings;
        MyCategories:DTOCategories;
        MyCategory:DTOCategory;
        i:integer;
    begin
        Session:=CoDTOSession.Create;
        Result:=Session.Connect
                ('ServerName','UserName','Password');
        MyCategories:=Session.Categories;
        for i:=1 to MyCategories.Count do
        MyCategory:=MyCategories.Item[i];
    end;
```

Example of Using Direct COM Calls

```
Example:

var
  Session, Categories, Category: Variant;
  I: Integer;
```

```
begin
  Session := CreateOleObject('DTO.DtoSession');
  Session.Connect('ServerName','UserName','Password');
  Categories := Session.Categories;
  for I := 1 to Categories.Count do
  Category := Categories.Item[I];
end;
```

DTO Object Summary

The Distributed Tuning Objects Reference is divided into three chapters by functional grouping. The objects in each chapter are listed in this section.

Connection Object

DtoSession

The **DtoSession** object is the core of DTO. It is through the **DtoSession** object that an application connects to Zen servers. A DTO application creates a **DtoSession** object and uses the **Connect** method when a session is required on a specific database server.

Configuration Objects

DtoCategory

Object **DtoCategory** and collection **DtoCategories** group database engine settings and allow the user further access to the **DtoSetting** objects.

DtoSetting

Object **DtoSetting** and collection **DtoSettings** expose the specific settings for the database engines, the communication managers, and the local requester components and allow the user to change these settings. Each category typically exposes a collection of settings.

DtoSelectionItem

Object **DtoSelectionItem** and collection **DtoSelectionItems** contain the full range of items in a setting where selection is allowed. **DtoSetting**.**AllPossibleSelections** returns a collection of all possible values for a given setting.

DtoServices

Object **DtoServices** allows the user to start and stop Zen database services, and to query the current status of a service.

DtoLicenseMgr

Object DtoLicenseMgr (DTO2) allows for adding and removing licenses, as well as viewing product information .

Monitoring Objects

DtoMonitor

Object **DtoMonitor** allows the user to retrieve real-time status information of the database engine and other related services.

Also exposed in DtoMonitor object is information about resource usage such as current, peak, and maximum settings for file handles, open files, and licenses. A peak value is the maximum value since the last engine restart.

DtoOpenFile

Object **DtoOpenFile** and collection **DtoOpenFiles** contain information about active files. This allows user to monitor file access by determining how many files are open, what users have them open, and other related information.

DtoFileHandle

Object **DtoFileHandle** and collection **DtoFileHandles** expose user name/agent ID, connection, task, site, network address, open mode, record lock type, wait state, and transaction state.

DtoMkdeClient

Object **DtoMkdeClient** and collection **DtoMkdeClients** expose the information about active clients. For a particular client, you can query to see if there is an active session of that client and if so, get data about that session and optionally terminate the client.

DtoMkdeClientHandle

Object **DtoMkdeClientHandle** and collection **DtoMkdeClientHandles** expose handle information, including the name and associated information for each file.

DtoMkdeVersion

Object DtoMkdeVersion (DTO2) exposes the major and minor version, build number, and target operating system of the Zen engine.

DtoEngineInformation

Object DtoEngineInformation (DTO2) exposes the major and minor version, DTI API version, and other server and client information.

DtoSqlClient

Object **DtoSqlClient** and collection **DtoSqlClients** expose the information about active SQL users such as count and list of active SQL users, and detail information about each client.

DtoCommStat

Object **DtoCommStat** exposes the communication statistics. You can query the current, peak, and maximum values where appropriate.

DtoProtocolStat

Object **DtoProtocolStat** and collection **DtoProtocolStats** expose information regarding each network protocol running on a server.

Database and Dictionary Objects

DtoDatabase

Object **DtoDatabase** and collection **DtoDatabases** are responsible for managing the database catalog information, such as database name, database flags, security, and table definitions.

DtoDSN

Object **DtoDSN** and collection **DtoDSNs** represent the Zen DSNs on your server. They can be used to create new DSNs and to manage existing Zen ODBC DSNs.

DtoDictionary (deprecated)

Object **DtoDictionary** is the root object for all of the operations affecting dictionary files. Use this object to open a dictionary, create a dictionary, get table information add a table or drop a table.

The preferred method of accessing the Tables collection is using the **DtoDatabase** object.

DtoTable

Object **DtoTable** and collection **DtoTables** are responsible for managing the table information, such as name, columns, and indexes.

DtoColumn

Object **DtoColumn** and collection **DtoColumns** are responsible for managing the information about the columns.

DtoIndex

Object **DtoIndex** and collection **DtoIndexes** expose the indexes defined for a table.

DtoSegment

Object **DtoSegment** and collection **DtoSegments** contain information on the segments of a given index for a table.

Working with DTO Collections

Collections are objects that contain other objects.

Instantiating a Collection

Visual Basic

Use the Set keyword to set a variable to the collection object.

```
Dim result as DtoResult
Dim my_session as New DtoSession
Dim my_databases as DtoDatabases
result = my_session.Connect("myserver", "username", "pw")

if not(result=0) Then
' error handling for Connect method
end if

' Use Set when an object or collection
' is the return type

Set my_databases = my_session.Databases
```

ASP

When using ASP, you normally need to instantiate only two objects directly: DtoSession and DtoDictionary. Instantiating an object with Active Server Pages uses the Create Object syntax on the built-in Server object of IIS.

```
Set my_session = Server.CreateObject("DTO.DtoSession.2")
```

Note: If you want the object to exist between HTTP calls, you must use the IIS built-in object Session to maintain the object state, as shown in the following example.

```
Set Session("my_session") = Server.CreateObject("DTO.DtoSession.2")
```

However, if you create new objects, such as databases, DSNs, tables, columns, indexes, or segments, you use this same syntax. The progid for each object follows the convention shown in the preceding example.

For other collections, you can use the Set syntax documented previously for Visual Basic.

Looping through a Collection

Visual Basic

There are two ways to loop through collections in Visual Basic: using a For loop and using a counter.

The following shows the Visual Basic syntax to loop through a collection using a For/Next statement.

```
' obtain categories collection
Dim my_categories as DtoCategories
Dim category as DtoCategory
Set my_categories = my_session.Categories

' loop through collection
For Each category In my_categories
   settings = category.Settings
Next
```

The following shows the Visual Basic syntax to loop through a collection using a counter.

```
Dim column as DtoColumn
Dim table as DtoTable
Set table = dictionary.Tables("Billing")
Dim i as long
for i=1 to table.Columns.count
  set column=table.Columns(i)
  'perform operations with this column
next i
```

ASP

Here is sample ASP code that displays a list of categories for Zen configuration:

```
<%
Set Session("my_session") = Server.CreateObject("DTO.DtoSession.2")
result = Session("my_session").Connect("myserver", "username", "pw")
' Error handling for Connect method not shown

Set my_categories = Session("my_session").Categories
' Now loop through and print categories in a unordered HTML list (<UL></UL>)
%>

<ul>
<% For each category in my_categories %>

<% ' The = (equal) sign displays the variable %>
<% ' in the HTML stream and is a short cut  %>
<% ' for the Response.Write() built-in  %>
<% ' VBScript method. %>

<li><%=cat.CategoryId> - <%=cat.Name%></li>

<% Next %>
</ul>
```

Obtaining Number of Members

Visual Basic

Use the Count property to determine the number of objects in the collection.

```
Dim num_items as Integer
Dim my_session as New DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "pw")
```

```
' Error handling for Connect method not shown
Set my_databases = my_session.Databases
num_items = my_databases.Count
```

ASP

Use the Count property to determine the number of objects in the collection.

```
<%
Set Session("my_session") = Server.CreateObject("DTO.DtoSession.2")
result = Session("my_session").Connect("myserver", "username", "pw")
' Error handling for Connect method not shown
Set my_databases = Session("my_session").Databases
num_items = my_databases.Count
' Now write output to HTML stream
Response.Write("<p>Number of databases = " & num_items)
%>
```

Obtaining a Specific Member

Visual Basic and ASP

Use the Item property to obtain a specific member object of a collection.

Elements within collections can be accessed using their ordinals. Some collections support accessing elements using the element's unique property, such as name. All ordinals are 1-based.

Since Item is a default property of the collection object, the following two statements are identical:

```
Collection.Item(index)
Collection(index)
```

Where to Find DTO Samples

The Zen SDK includes a complete DTO sample written in Visual Basic. This sample is included in your SDK installation of samples and header files at the following location.

The SDK components, code snippets, and samples are available on the Actian website.

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

Also look at the Actian website for other developer information and resources pertaining to the DTO access method.

Establishing a DTO Session

This topic provides information about establishing a session using the Distributed Tuning Objects:

- DtoSession Object

DtoSession Object

Object **DtoSession** is the root object for most DTO operations. It represents and manages a connection to a Zen database engine.

Properties

Connected	Returns a Boolean indicating whether the Session object is connected to the Zen engine. True = connected False = not connected
Error	Returns error of the last method call. Pass the result of the method call and a dtoResult string will be returned that explains the error. See DtoResult for a listing of these error codes.
ServerName	Gets or sets the server name of a DtoSession object.
UserName	Sets the user name for an object.
Password	Sets the password for a session.

Collections

DtoCategories Collection

DtoDatabases Collection

DtoDSNs Collection

Objects

DtoMonitor Object

DtoServices Object

DtoLicenseMgr Object

DtoEngineInformation Object

Methods

Connect method

Disconnect method

GetSetting method

Remarks

A DtoSession object is the starting point for all operations except dictionaries. You use DtoSession to make connection to servers, obtain configuration information such as categories and settings, explore databases and DSNs, and to monitor Zen usage information.

To use DtoSession, first instantiate your object and use the Connect method to specify a server for the session object.

The user name and password you use for the session connect is for the machine only. This does not authenticate you to any Zen database.

Note: If instantiating this object using ASP or if you use the CreateObject method in Visual Basic, the progid of DtoSession is "DTO.DtoSession.2" (DTO version 2) or “DTO.DtoSession.1” (DTO version 1). See DTO2 for information on the differences between the two versions.

Example

```
' instantiate session object
Dim my_session as New DtoSession

' connect to a server
result = my_session.Connect("myserver", "username", "password")

' check for good connection using the Error property
if Not (result = Dto_Success)
Then Msgbox"Could not connect to the server. Error was "+ my_session.Error(result)

' now use your session object to obtain
' category and database collections

Dim my_categories as DtoCategories
Dim my_databases as DtoDatabases
Set my_categories = my_session.Categories
Set my_databases = my_session.Databases
```

See Also

DtoCategories Collection

DtoServices Object

DtoMonitor Object

DtoDatabases Collection

DtoDSNs Collection

Methods Detail

Connect method

Open a connection to a server.

Syntax

```
Dim result as DtoResult
result = Object.Connect([server], [username], [password])
```

Arguments

Object	DtoSession object
server	(optional) Name of the server to which you want to connect. If omitted, an attempt to connect to the local server is made. You can also first set the ServerName property and call the method without specifying this parameter.
username	(optional) User name for the server. You can also first set the UserName property and call the method without specifying this parameter.
password	(optional) Password for the user. You can also first set the Password property and call the method without specifying this parameter.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property to obtain a description for the result.

Remarks

Since connections to servers can fail for a variety of reasons, be sure to check the return value for this method and take appropriate action programmatically.

If user name and password are not specified, an attempt will be made to login as a guest. If such an attempt is successful, some functionality will not be available.

Check the isConnected property in order to see whether the session is currently connected.

Examples

```
Dim result as DtoResult
Dim my_session as New DtoSession
result = my_session.Connect("myserver", "smook", "1234")

Dim result as DtoResult
Dim my_session as New DtoSession
my_session.UserName="smook"
my_session.Password="1234"
my_session.ServerName="myserver"
result = my_session.Connect
```

Disconnect method

End a connection to a server.

Syntax

```
result = Object.Disconnect
```

Arguments

Object	DtoSession object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property to obtain a description for the result.

Remarks

This method should be called for every server to which you are connected prior to using the session object to connect to another server, or exiting the application.

Example

```
Dim result as DtoResult
```

```
Dim my_session as New DtoSession
result = my_session.Connect("myserver", "username", "pw")
'
' perform operations here
'
result = my_session.Disconnect
```

GetSetting method

Obtains a DtoSetting object using a setting id.

Syntax

```
Set my_setting = Object.GetSetting(setting_id)
```

Arguments

Object	DtoSession object
setting_id	A valid setting id. Each DtoSetting object has a property SettingID that uniquely identifies it. If you are trying to obtain all settings for a particular category, use the Settings property of the DtoCategory Object. This method is useful if you already know a particular setting you wish to obtain and you have previously stored its setting_id.

Return Values

my_setting	DtoSetting object. If the given setting cannot be found, NULL is returned.

Remarks

This method is useful for obtaining a specific setting without having to first obtain a category and then searching through a DtoSettings collection.

Example

```
Dim oSession As New DtoSession
Dim Result As dtoResult
Result = oSession.Connect("localhost", "", "")

Dim oSetting As DtoSetting
Dim settingFileversion As Integer
settingFileversion = 97
Set oSetting = oSession.GetSetting(settingFileversion)

If oSetting Is Nothing Then
MsgBox "Invalid setting"
Else
Dim new_selections As New DtoSelectionItems
'0 = 9.5 '1 = 9.0 '2 = 8.x '3 = 7.x '4 = 6.x '5 = 13.0
'These are file format values
```

```
new_selections.Add oSetting.AllPossibleSelections.GetByID(0)
oSetting.Value = new_selections
End If
```

Configuring Zen Servers with DTO

The following topics provide information about the objects that comprise the configuration group of the Distributed Tuning Objects:

- DtoCategories Collection
- DtoCategory Object
- DtoLicenseMgr Object
- DtoSettings Collection
- DtoSetting Object
- DtoSelectionItems Collection
- DtoSelectionItem Object
- DtoServices Object

DtoCategories Collection

This object is a collection of **DtoCategory** objects representing all the setting categories available for a particular **DtoSession** object.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

None

Remarks

This collection allows retrieving individual items by passing a 1-based ordinal.

Use the Count property to find the number of members in the collection.

```
num_categories = my_categories.Count
```

Use the Item property to retrieve the one-based index of a collection.

```
Set first_item = my_categories(1)
```

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' now obtain categories collection
Dim my_categories as DtoCategories
Set my_categories = my_session.Categories
```

See Also

DtoCategory Object

DtoSession Object

DtoCategory Object

This object allows you to perform operations on a particular category from a **DtoCategories** collection.

Properties

CategoryID	Returns unique category ID for a DtoCategory.
Name	Returns name of the category
Session	Returns session of the category

Collections

DtoSettings Collection

Methods

None

Remarks

To get a list of settings for a category, use the Settings property to return a DtoSettings collection. You can then use the DtoSetting objects contained therein to obtain information relating to a particular setting.

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
Dim category as DtoCategory
Dim my_categories as DtoCategories
Dim settings as DtoSettings
result = my_session.Connect("myserver", "username", "password")

' now obtain categories collection
Set my_categories = my_session.Categories

' loop through collection
For Each category In my_categories
  Set settings = category.Settings
Next
```

See Also

DtoCategories Collection

DtoSession Object

DtoLicenseMgr Object

DTO2 only: This object allows you to authorize and deauthorize product licenses, initiate a license validation action, and retrieve an XML string of product information.

Properties

None

Collections

None

Methods

AddLicense

DeleteLicense

GetProductInfo

ValidateLicenses

Remarks

Obtain from Session object

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
Dim my_licmgr as DtoLicenseMgr

result = my_session.Connect("myserver", "username", "password")

' now obtain License Manager object
Set my_licmgr = my_session.LicenseMgr

' Add a license

result = my_licmgr.AddLicense(“ERXVD3U4ZS9KR94QPDHV5BN2”)
```

See Also

DtoSession Object

Methods Detail

AddLicense

Authorizes a license.

Syntax

```
result = LicenseManager.AddLicense(License)
```

Arguments

LicenseManager	DtoLicenseMgr object
License	A valid license key to authorize on the engine you are currently connected to with a DtoSession object.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
Dim my_licmgr as DtoLicenseMgr

result = my_session.Connect("myserver", "username", "password")

' now obtain License Manager object
Set my_licmgr = my_session.LicenseMgr

' Add a license

result = my_licmgr.AddLicense(“ERXVD3U4ZS9KR94QPDHV5BN2”)
```

DeleteLicense

Deauthorizes a license.

Syntax

```
result = LicenseManager.DeleteLicense(License)
```

Arguments

LicenseManager	DtoLicenseMgr object
License	A valid license key to deauthorize from the engine you are currently connected to with a DtoSession object.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

None

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
Dim my_licmgr as DtoLicenseMgr

result = my_session.Connect("myserver", "username", "password")

' now obtain License Manager object
Set my_licmgr = my_session.LicenseMgr

' Delete a license

result = my_licmgr.DeleteLicense(“ERXVD3U4ZS9KR94QPDHV5BN2”)
```

GetProductInfo

Retrieves a list of all Zen products found by the License Manager.

Syntax

```
result = LicenseManager.GetProductInfo
```

Arguments

LicenseManager	DtoLicenseMgr object

Return Values

result	Returns a list of products, in an XML formatted string.

Remarks

For information about the XML formatted string, see Remarks for PvGetProductsInfo() in *Distributed Tuning Interface Guide*.

Example

```
' instantiate session object and connect to server
Dim session As New DtoSession
Set session = New DtoSession
Dim result As dtoResult
result = session.Connect("server", "user", "password")

If result <> Dto_Success Then
MsgBox "Error on connect." & CStr(result)
Exit Sub
End If
Dim xmlstring As String
xmlstring = session.LicenseMgr.GetProductInfo
RichTextBox1.TextRTF = xmlstring
```

ValidateLicenses

Initiates a check of the validity of all keys on the computer specified by the session connection.

Syntax

```
result = LicenseManager.ValidateLicenses
```

Arguments

LicenseManager	DtoLicenseMgr object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

ValidateLicenses returns only the result from *requesting* a validation check. It does *not* return any information about the state of the keys. You must separately call GetProductInfo to get the XML string of product information that includes information about the state of the keys.

Example

```
' instantiate session object and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
Dim my_licmgr as DtoLicenseMgr

result = my_session.Connect("myserver", "username", "password")

' now obtain License Manager object
Set my_licmgr = my_session.LicenseMgr

' initiate a validation check of all keys
result = my_licmgr.ValidateLicenses
```

DtoSettings Collection

This collection contains DtoSetting objects which represent all the settings for a particular DtoCategory object.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

This collection allows retrieving individual items by passing a variant containing either the 1-based ordinal or the setting name.

Note: Individual settings can also be obtained using the GetSetting method of DtoSession Object, thus saving a loop through the categories and settings.

Example

```
Dim my_categories as DtoCategories
Set my_categories = my_session.Categories

Dim my_settings as DtoSettings
Dim first_setting as DtoSetting
Set my_settings = my_categories.Settings
Set first_setting = my_settings(1)
```

See Also

DtoCategories Collection

DtoCategory Object

DtoSetting Object

This object represents a configuration setting.

Properties

AllPossibleSelections	Returns DtoSelectionItems Collection or DtoSelectionItem Object representing all the possible items in a single or multiple type setting. This property is only valid for a setting with property Type of 'dtoSingleSelection' or 'dtoMultiSelection', which corresponds to property TypeNames of 'Single Selection' and 'Multiple Selection'.
Category	Returns DtoCategory object associated with this setting.
DefaultValue	Returns default value for the setting. The returned value is a variant based on the Type of the setting. A Type of Single selection returns a DtoSelectionItem object. A Type of Multiple selection returns a DtoSelectionItems collection.
Factor	Returns factor value of the setting. For example, many settings are stored by Zen in bytes, but for the purpose of changing the setting the user might input a value in kilobytes. If a setting returned 16384 for the Value property and the Factor property returned 1024, your program should divide 16384 by 1024 and return 16 to the user. Then query the FactorString property to get the new units. In this case, it would be kilobytes. Before setting the Value property , you should multiply the user-supplied value by Factor.
FactorString	Returns units for the Value property adjusted for the Factor property. For example, if the UnitString property returns "bytes," the FactorString property will return "kbytes," and Factor property will return "1024".
FalseString	Returns false value for a Boolean type setting. This property is only valid for a Boolean type setting. Use the Type property to determine if the setting is Boolean.
Help	Returns help text associated with a setting.
IsClient	Returns a Boolean of whether the setting is applicable to Zen Client or Enterprise Server. True = Client False = Server

Max	Returns maximum value of a Long type setting. This property is only valid for a Long type setting. Use the Type property to determine if the setting is Long. If this property returns a negative number, interpret it as follows: /* Maximum valid memory or disk size */P_MAX_MEM_DISK_SIZE -129 /* Maximum size limited by available disk space */ P_MAX_LIMITED_BY_DISK -2 /* Maximum size limited by available memory */ P_MAX_LIMITED_BY_MEMORY -1
Min	Returns minimum value of a Long type setting. This property is only valid for a Long type setting. Use the Type property to determine if the setting is Long. This property returns -1 if the property is not applicable.
Name	Returns name of the setting.
Rank	Returns rank of a setting. The rank groups settings according to whether they apply to advanced users only. 0 = Normal 1 = Advanced
Session	Returns session associated with this object.
SettingID	Returns unique identifier for a setting.
TrueString	Returns true value for a Boolean type setting. This property is only valid for a Boolean type setting. Use the Type property to determine if the setting is Boolean.
Type	Returns setting type (Setting Type enumeration) 0 = Boolean 1 = Long 2 = String 3 = Single selection 4 = Multiple selection
TypeName	Returns setting type in string form Boolean Long String Single selection Multiple selection

UnitString	Returns measure of a long type setting. For example: **seconds, bytes** To adjust the Value for a more user-friendly range of values, use the Factor and FactorString properties.
Value	Gets or sets the value of a setting. The returned value is a variant based on the Type of the setting. A Type of Single selection returns a DtoSelectionItem object. A Type of Multiple selection returns a DtoSelectionItems collection. When setting this property for a Long type setting, check to see that the value is within the limits for the particular setting by querying the Min and Max properties.

Collections

DtoSelectionItems Collection

Methods

None

Remarks

Use the Type property to find the type of the setting. Note that, depending on the type:

- The properties `TrueString` and `FalseString` apply to Boolean type settings (0) only.
- The properties `Factor`, `FactorString`, `Max`, `Min`, and `UnitString` apply to Long type settings (1) only.

Example

```
Set my_settings = my_category.Settings
Set first_setting = my_settings(1)
```

See Also

DtoCategories Collection

DtoCategory Object

DtoSelectionItems Collection

A collection of DtoSelectionItem objects representing the possible values of a selection type setting.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

Add method

GetById method

Remove method

Remarks

The property `AllPossibleSelections` of object DtoSetting Object returns the **DtoSelectionItems** collection.

Use the Count property to find the number of members in the collection.

Example

```
Set first_setting = my_settings(1)

type = first_setting.Type

' only call this for selection type settings
' see Setting Type enumeration
' for a list of setting types

if (type = dtoSingleSel) OR (type = dtoMultiSel)
Set all_the_selections = first_setting.AllPossibleSelections
```

See Also

DtoCategories Collection

DtoSetting Object

Methods Detail

Add method

Add an item to a DtoSelectionItems collection.

Syntax

```
result = Collection.Add(Object)
```

Arguments

Collection	DtoSelectionItems Collection to which to add object.
Object	A DtoSelectionItem object that you want to add.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type DtoSelectionItem. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

Example

```
Dim Result As dtoResult
Dim Session As New DtoSession
Result = Session.Connect("nik-ntws", "", "")
Dim my_setting As DtoSetting
Dim SetID As Long
SetID = 26
Set my_setting = Session.GetSetting(SetID)
   If my_setting Is Nothing Then
        MsgBox " Setting is wrong"
        Exit Sub
   End If

' Start to assign new values:
' Add Item with ItemID 1
new_selections.Add my_setting.AllPossibleSelections.Item(1)
' TCP

my_setting.Value = new_selections
```

GetById method

Returns a DtoSelectionItem object from a DtoSelectionItems collection given an id.

Syntax

```
my_selection_item = Collection.GetById(id)
```

Arguments

Collection	DtoSelectionItems collection
id	The id of the item you wish to retrieve from the collection. The id for a particular selection item can be obtained with the **ItemId** property of the DtoSelectionItem object.

Return Values

my_selection_item	DtoSelectionItem Object corresponding to *id.*

Example

```
Dim Result As DtoResult
Dim Session As New DtoSession
Result = Session.Connect("nik-ntws", "", "")
Dim my_setting As DtoSetting
Dim SetID As Long
SetID = 26
Set my_setting = Session.GetSetting(SetID)
   If my_setting Is Nothing Then
        MsgBox " Setting is wrong"
        Exit Sub
   End If

Dim new_selections As New DtoSelectionItems
new_selections.Add my_setting.AllPossibleSelections.GetByID(3)  'Microsoft TCP/IP

my_setting.Value = new_selections
```

Remove method

Removes an item from a DtoSelectionItems collection

Syntax

```
result = Collection.Remove(item)
```

Arguments

Collection	DtoSelectionItems collection
item	Variant that can contain the index (starting with 1) of the item you wish to remove from the collection or the name of the selection item.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type DtoSelectionItem

Example

```
Dim Result As DtoResult
Dim Session As New DtoSession
Result = Session.Connect("nik-ntws", "", "")
Dim my_setting As DtoSetting
Dim SetID As Long
SetID = 26
Set my_setting = Session.GetSetting(SetID)
   If my_setting Is Nothing Then
        MsgBox " Setting is wrong"
        Exit Sub
   End If

Dim new_selections As New DtoSelectionItems
new_selections.Add my_setting.AllPossibleSelections.GetByID(3)  ''Microsoft TCP/IP
new_selections.Remove(1)

my_setting.Value = new_selections
```

DtoSelectionItem Object

An object representing a possible value of a selection type setting.

Properties

ItemID	Returns unique ID of a selection item.
Setting	Returns setting to which this selection item applies.
String	Returns value of a selection item.

Methods

None

Remarks

The **AllPossibleSelections** property of the DtoSetting Object returns the DtoSelectionItems collection.

Example

```
Set first_setting = my_settings.Item(1)

Dim type as dtoSettingType
type = first_setting.Type

' only call this for selection type settings
' see Setting Type enumeration
' for a list of setting types

if (type = dtoSingleSel) OR (type = dtoMultiSel) then
Dim all_the_selections as DtoSelectionItems
Dim selection as DtoSelectionitem
Set all_the_selections = first_setting.AllPossibleSelections

Dim String_text as String
For each selection in all_the_selections
  String_text = selection.String
Next
```

See Also

DtoSelectionItems Collection
DtoSetting Object

DtoServices Object

This object is a collection of DtoService objects, representing the Zen services running on the server.

Properties

Status	Returns status of a service. You must pass the service of which you want the status: dtoServiceTransactional dtoServiceRelational dtoServiceIDS
StatusString	Returns string representation of the current status.

Methods

RestartAllServices method

StartRelational method

StartTransactional method

StopRelational method

StopTransactional method

StartDXAgent

StartDXReplication

StopDXAgent

StopDXReplication

Remarks

The methods of DtoServices control the Zen engine services running on the machine you connected to with the DtoSession object. All these methods return the enumeration DtoResult.

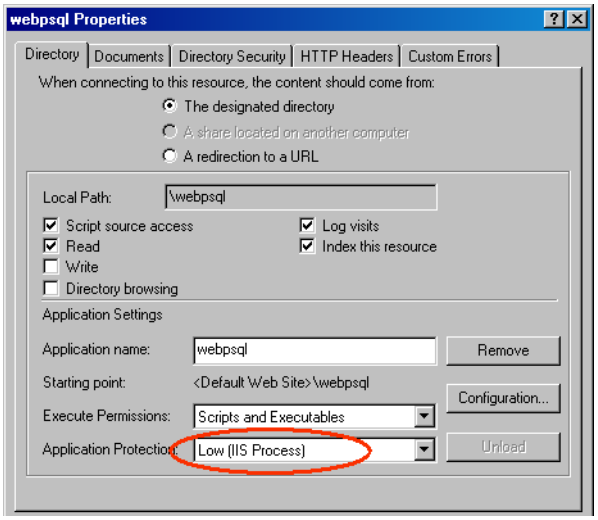
This object lets you start and stop the Zen engine services running on Windows platforms. Also, you can query the current status of Zen services using the **Status** or **StatusString** properties.

Security Information Regarding DtoServices Object

- This object uses the same user name and password as the DtoSession object to connect to a Windows server.
- When using the methods of this object from a web application hosted by Microsoft Internet Information Service (IIS), you must set properties on the directory where the DTO web application resides so that IIS allows DTO to run in the same process as the IIS service. Otherwise, you can obtain the current state of the services, but using the start or stop methods returns DTO error 431 (access denied). To set the IIS folder properties needed for the methods of the DTO Services object, perform the following procedure on folders where DTO web applications are located. See the Microsoft IIS documentation for more information on IIS configuration.

To configure IIS to allow starting and stopping of services from a DTO web application

1. Click **Start**, select **Settings** and point to **Control Panel**.
2. Double-click **Administrative Tools**.
3. Double-click **Internet Service Manager**.
4. Browse to the folder containing your DTO ASP application.
5. Right-click the folder in the left pane and select **Properties**.
6. Click the **Directory** tab.
7. Specify **Low (IIS Process)** in the **Application Protection** field in the dialog for IIS directory properties for DTO Services Methods.



Examples

```
' This example connects to a server and restarts
' Zen services.

Dim my_session as new DtoSession
Dim my_services as DtoServices
Dim result as DtoResult

result = my_session.Connect("myserver", "username", "password")
Set my_services = my_session.Services
result = my_services.RestartAllServices

' This example connects to a server and starts the
' DataExchange (DX) Agent service and the DX replication
' service.

Dim my_session as new DtoSession
Dim my_services as DtoServices
Dim result1 as DtoResult
Dim result2 as DtoResult

result = my_session.Connect("myserver", "username", "password")
Set my_services = my_session.Services
result1 = my_services.StartDXReplication
result2 = my_services.StartDXAgent
```

See Also

DtoSession Object

DtoSetting Object

Methods Detail

RestartAllServices method

Stops and then restarts the services for transactional, relational, DataExchange (DX) agent, and DX replication.

Syntax

```
result = Services.RestartAllServices
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StartRelational method

Starts the Relational service. As of Zen v14, this is the same as the StartTransactional method.

Syntax

```
result = Services.StartRelational
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StartTransactional method

Starts the Btrieve transactional engine service. As of Zen v14, this is the same as the StartRelational method.

Syntax

```
result = Services.StartTransactional
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StopRelational method

Stops the Relational engine service. As of Zen v14, this is the same as the StopTransactional method.

Syntax

```
result = Services.StopRelational
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StopTransactional method

Stops the Btrieve transactional engine service. As of Zen v14, this is the same as the StopRelational method.

Syntax

```
result = Services.StopTransactional
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StartDXAgent

Starts the DataExchange (DX) agent service. The DX agent is a component that detects critical replication failures and notifies administrators by e-mail. See DataExchange documentation for more information.

You can start the DX agent service before the DX replication service, but the agent returns a message informing you that replication is stopped. This is expected behavior because the replication service is not yet running.

Syntax

```
result = Services.StartDXAgent
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StartDXReplication

Starts the DataExchange (DX) replication service (the Replication Engine). The Replication Engine captures and shares changes from one Zen database to other databases in a DataExchange replication network. See DataExchange documentation for more information.

Starting the replication service also starts the transactional and the relational services.

Syntax

```
result = Services.StartDXReplication
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StopDXAgent

Stops the DataExchange (DX) agent service. The DX agent is a component that detects critical replication failures and notifies administrators by email. See DataExchange documentation for more information.

Syntax

```
result = Services.StopDXAgent
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

StopDXReplication

Stops the DataExchange (DX) Replication Engine. The Replication Engine captures and shares changes from one Zen database to other databases in a DataExchange replication network. See DataExchange documentation for more information.

Syntax

```
result = Services.StopDXReplication
```

Arguments

Services	DtoServices object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Monitoring Zen Servers with DTO

The following topics provide information about the objects that comprise the monitoring group of the Distributed Tuning Objects:

- DtoMonitor Object
- DtoOpenFiles Collection
- DtoOpenFile Object
- DtoFileHandles Collection
- DtoFileHandle Object
- DtoMkdeClients Collection
- DtoMkdeClient Object
- DtoMkdeClientHandles Collection
- DtoMkdeClientHandle Object
- DtoCommStat Object
- DtoProtocolStats Collection
- DtoProtocolStat Object
- DtoSqlClients Collection
- DtoSqlClient Object
- DtoMkdeVersion Object
- DtoEngineInformation Object

DtoMonitor Object

This object provides usage information about a Zen server. It is the root object of all other monitoring operations.

Properties

CurClients	Returns current number of clients for a session.
CurFilesInUse	Returns current number of files in use for a session.
CurHandlesInUse	Returns current number of handles in use for a session.
CurLicensesInUse	Returns current number of licenses in use for a session.
CurLicDataInUseMB	Returns current value in megabytes (MB) of data in use. DTO2 only.
CurLocksInUse	Returns current number of locks in use for a session.
CurSessionCountInUse	Returns current number of sessions in use (current session count). DTO2 only.
CurThreads	Returns number of threads for a session.
CurTransInUse	Returns current number of open transactions for a session.
EngineUpTimeSecs	Returns how long in seconds the database engine has been running. DTO2 only.
MaxClients	Returns maximum number of clients for a session.
MaxFiles	Returns maximum number of files for a session.
MaxHandles	Returns maximum number of handles for a session.
MaxLicenses	Returns user license count for a session.
MaxLicDataMB	Returns maximum allowed size in megabytes (MB) for data in use (data in use limit). The hex value 0xFFFFFFFF means unlimited. DTO2 only.
MaxSessionCount	Returns maximum number of sessions allowed by a license (session count limit). The hex value 0xFFFFFFFF means unlimited. DTO2 only.
MaxThreads	Returns maximum number of threads for a session.
MaxTrans	Returns maximum number of transactions for a session.
PeakClients	Returns highest number of clients for a session
PeakFilesInUse	Returns highest number of files in use for a session
PeakHandlesInUse	Returns highest number of handles in use for a session

PeakLicensesInUse	Returns highest number of licenses in use for a session
PeakLicDataInUseMB	Returns peak value in megabytes (MB) of concurrent data in use. DTO2 only.
PeakLocksInUse	Returns highest number of locks in use for a session
PeakSessionCountInUse	Returns peak number of concurrent sessions in use. DTO2 only.
PeakThreads	Returns highest number of threads for a session.
PeakTransInUse	Returns highest number of transactions in use for a session.

Collections

DtoMkdeClients Collection

DtoOpenFiles Collection

DtoSqlClients Collection

Objects

DtoCommStat Object

DtoSession Object

DtoMkdeVersion Object

Methods

None

Example

```
' Instantiate session and connect to server
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Now get DtoMonitor object from DtoSession
Dim session_monitor as DtoMonitor
Set session_monitor = my_session.Monitor

' Now get current files in use
Dim current_files as long
current_files = session_monitor.CurFilesInUse
```

See Also

DtoOpenFiles Collection

DtoMkdeClients Collection

DtoMonitor Object

DtoOpenFiles Collection

A collection of DtoOpenFile objects representing currently open files.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoOpenFiles collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

You can obtain a DtoOpenFiles collection through the properties of the DtoMonitor Object object.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get open files from monitor
Dim my_openfiles as DtoOpenFiles
Set my_openfiles = my_monitor.OpenFiles
```

See Also

DtoMonitor Object

DtoOpenFile Object

An object representing an open file.

Properties

ActiveCursors	Returns number of active cursors for an open file.
AFLIndex	Returns AFL Index for an open file.
ContinuousOps	Returns whether an open file is using continuous operations.
FileName	Returns file name associated with an open file.
IsLocked	Returns whether an open file is locked. 0 = Not locked 1 = Locked
IsReadOnly	Returns whether an open file is read-only. 0 = Not read-only 1 = Read-only
IsTrans	Returns whether an open file is in a transaction state . 0 = No 1 = Yes
Monitor	Returns DtoMonitor object associated with the open file.
OpenMode	Returns open mode of the open file.
OpenModeName	Returns a text version of OpenMode.
PageSize	Returns page size of an open file.
PhysFileSizeKB	Returns physical size of the file in kilobytes (KB). DTO2 only.
ReferentialIntegrity	Returns whether referential integrity is set for an open file. 0 = No 1 = Yes
TimeStamp	Returns time stamp of an open file.
TTSFlag	Reserved for future use.

Methods

None

Collections

DtoFileHandles Collection

Remarks

This object represent a currently opened file. For a collection of all open files, use the DtoOpenFiles Collection.

Example

```
Dim my_session as new DtoSession
Dim is_read_only as Boolean
Dim my_monitor as DtoMonitor
Dim my_openfiles as DtoOpenFiles
Dim first_file as DtoOpenFile
Dim result as DtoResult

result = my_session.Connect("myserver", "username", "password")
Set my_monitor = my_session.Monitor
Set my_openfiles = my_monitor.OpenFiles
Set first_file = my_openfiles(1)
is_read_only = first_file.IsReadOnly
```

See Also

DtoOpenFiles Collection

DtoMonitor Object

DtoFileHandles Collection

A collection of DtoFileHandle objects representing all the file handles for an open file.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoFileHandles collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

Example

```
Dim my_session as new DtoSession
Dim my_monitor as DtoMonitor
Dim my_openfiles as DtoOpenFiles
Dim first_file as DtoOpenFile
Dim my_handles as DtoFileHandles
Dim result as DtoResult

result = my_session.Connect("myserver", "username", "password")
Set my_monitor = my_session.Monitor
Set my_openfiles = my_monitor.OpenFiles
Set first_file = my_openfiles.Item(1)
Set my_handles = first_file.FileHandles
```

See Also

DtoFileHandle Object

DtoOpenFile Object

DtoMonitor Object

DtoFileHandle Object

An object representing a file handle for an open file.

Properties

ClientIndex	Returns index for a file handle.
IsLocked	Returns whether a file handle is locked.
IsWaiting	Returns waiting status for a file handle.
OpenMode	Returns open mode for the file handle.
OpenModeName	Returns a text version of OpenMode.
TransState	Returns transaction state.
UserName	Returns user name associated with the file handle.

Methods

None

Remarks

Use the DtoFileHandles Collection to obtain a list of all file handles for an open file.

Example

```
Dim my_session as new DtoSession
Dim my_openfiles as DtoOpenFiles
Dim first_file as DtoOpenFile
Dim my_handles as DtoFileHandles
Dim first_handle as DtoFileHandle
Dim locked_state as Boolean
Dim result as DtoResult

result = my_session.Connect("myserver", "username", "password")
Set my_monitor = my_session.Monitor
Set my_openfiles = my_monitor.OpenFiles
Set first_file = my_openfiles.Item(1)
Set my_handles = first_file.FileHandles
Set first_handle = my_handles.Item(1)
locked_state = first_handle.IsLocked
```

See Also

DtoFileHandles Collection

DtoOpenFile Object

DtoMonitor Object

DtoMkdeClients Collection

A collection of MicroKernel Engine client objects.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoMkdeClients collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult

result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get MicroKernel Engine Clients from monitor
Dim my_mkdeclients as DtoMkdeClients
Set my_mkdeclients = my_monitor.MkdeClients
```

See Also

DtoMkdeClient Object

DtoMonitor Object

DtoMkdeClient Object

An object representing an active MicroKernel Engine client.

Properties

BtrvID	Returns Btrieve ID of a MicroKernel Engine client.
CacheAccesses	Returns number of cache accesses for a MicroKernel Engine client.
ClientPlatform	Returns client platform enumeration for a MicroKernel Engine client. See Client Platform for possible values.
ClientPlatformName	Returns a text version of ClientPlatform.
ClientSite	Returns client site for a MicroKernel Engine client.
ConnectionNumber	Returns connection number for a MicroKernel Engine client.
CurrentLocks	Returns current number of locks for a MicroKernel client.
DiskAccesses	Returns number of disk accesses for a MicroKernel Engine client.
NetAddress	Returns address of a MicroKernel Engine client.
NumCursors	Returns number of cursors for a MicroKernel Engine client.
RecordsDeleted	Returns number of deleted records for a MicroKernel Engine client.
RecordsInserted	Returns number of inserted records for a MicroKernel Engine client.
RecordsRead	Returns number of read records for a MicroKernel Engine client.
RecordsUpdated	Returns number of updated records for a MicroKernel Engine client.
ServiceAgentID	Returns service agent identification of a MicroKernel Engine client
TaskNumber	Returns task number of a MicroKernel Engine client.
TransLevel	Returns transaction level of a MicroKernel Engine client.
TransState	Returns transaction state enumeration. See Transaction State for possible values.
UserName	Returns MicroKernel Engine client user name.

Collections

DtoMkdeClientHandles Collection
DtoMonitor Object

Methods

Disconnect method

Remarks

To obtain all the MicroKernel Engine clients, use the DtoMkdeClients Collection.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")
' Get monitor object from session
Dim my_monitor as DtoMonitor
my_monitor = my_session.Monitor

' Now get MicroKernel Engine Clients from monitor
Dim my_mkdeclients as DtoMkdeClients
Set my_mkdeclients = my_monitor.MkdeClients

' retrieve first client and query a property
Dim first_client as DtoMkdeClient
Dim num_locks as long
Set first_client = my_mkdeclients.Item(1)
num_locks = first_client.CurrentLocks
```

See Also

DtoMkdeClientHandles Collection

DtoMkdeClients Collection

Methods Detail

Disconnect method

Disconnects a specific MicroKernel Engine client.

Syntax

```
result = Object.Disconnect
```

Arguments

In	Object	DtoMkdeClient object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Call this method for every MicroKernel Engine client that you wish to disconnect.

Example

```
Dim result as DtoResult
result = my_mkdeclient.Disconnect
```

DtoMkdeClientHandles Collection

A collection of DtoMkdeClientHandle objects.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoMkdeClientHandles collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get MicroKernel Engine Clients from monitor
Dim my_mkdeclients as DtoMkdeClients
Set my_mkdeclients = my_monitor.MkdeClients

' retrieve first client and get its client handles
Dim first_client as DtoMkdeClient
Dim my_clienthandles as DtoMkdeClientHandles
Set first_client = my_mkdeclients(1)
' to get all handles, use ClientHandles collection
Set my_clienthandles = first_client.ClientHandles

' determine number of members in the collection
Dim num_clienthandles as Long
num_clienthandles = my_clienthandles.Count
```

See Also

DtoMkdeClientHandle Object
DtoMonitor Object

DtoMkdeClientHandle Object

An object representing a MicroKernel client handle.

Properties

FileName	Returns file name associated with a MicroKernel client handle
LockType	Returns locks status enumeration of a MicroKernel client handle. See Lock Type for possible values.
OpenMode	Returns open mode enumeration of the MicroKernel client handle. See Open Mode for possible values.
OpenModeName	Returns a text version of OpenMode.
TransState	Returns transaction state.
WaitState	Returns wait status enumeration for a MicroKernel client handle. See Wait State for possible values.

Methods

None

Remarks

To obtain all the MicroKernel client handles for a specific client, use the DtoMkdeClientHandles Collection.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get MicroKernel Engine Clients from monitor
Dim my_mkdeclients as DtoMkdeClients
Set my_mkdeclients = my_monitor.MkdeClients

' retrieve first client and get its client handles
Dim first_client as DtoMkdeClient
Dim my_clienthandles as DtoMkdeClientHandles
```

```
Set first_client = my_mkdeclients(1)
Set my_clienthandles = first_client.MkdeClientHandles

' determine number of members in the collection
Dim num_clienthandles as long
num_clienthandles = my_clienthandles.Count

' get first client handle and query file name
Dim first_clienthandle as DtoMkdeClientHandle
Dim fileName as string
Set first_clienthandle = my_clienthandles(1)
fileName = first_clienthandle.FileName
```

See Also

DtoMkdeClientHandles Collection

DtoMkdeClient Object

DtoMonitor Object

DtoCommStat Object

An object that represents usage information for a server.

Properties

CurActiveThreads	Returns current number of active threads for a session.
CurQueuedRequests	Returns number of queued requests for a session.
CurRemoteSessions	Returns number of queued requests for a session or protocol.
MaxActiveThreads	Returns maximum number of active threads for a session.
MaxQueuedRequests	Returns maximum number of queued requests for a session.
MaxRemoteSessions	Returns maximum number of remote sessions for a session.
PeakActiveThreads	Returns highest number of active threads for a session
PeakQueuedRequests	Returns highest number of queued requests for a session.
PeakRemoteSessions	Returns highest number of remote sessions for a session
RequestsProcessed	Returns total number of requests processed for a session
TotalTimeOuts	DTO2 only: Returns total number of communication time outs.
TotalRecoveries	DTO2 only: Returns total number of reconnects using the Zen auto reconnect feature. See *Advanced Operations Guide* for more information.

Collections

DtoProtocolStats Collection

Objects

DtoMonitor Object

Methods

None

Remarks

All the properties for this object return values of type Long integer.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Get comm stat object
Dim my_commstat as DtoCommStat
Set my_commstat = my_monitor.MkdeCommStat

' Get total number of requests processed
Dim requests as long
requests = my_commstat.RequestsProcessed
```

See Also

DtoMonitor Object

DtoProtocolStats Collection

DtoProtocolStat Object

DtoProtocolStats Collection

A collection of DtoProtocolStat objects.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

None

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' Instantiate Session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get Monitor object from Session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Get Comm stat object from Monitor
Dim my_commstat as DtoCommStat
Set my_commstat = my_monitor.MkdeCommStat

' Now get Protocol stats collection from Comm stat
Dim my_protocols as DtoProtocolStats
Set my_protocols = my_commstat.ProtocolStats
```

See Also

DtoProtocolStat Object

DtoMonitor Object

DtoProtocolStat Object

Provides information about a communications protocol.

Properties

CurRemoteSessions	Returns number of queued requests for a session or protocol.
PeakRemoteSessions	Returns highest number of remote sessions for a session or protocol.
ProtocolID	Returns ID of a protocol. Only the following return code is currently supported: 4 – WINSOCK TPC/IP
RequestsProcessed	Returns total number of requests processed for a session or protocol.

Methods

None

Remarks

To access a particular protocol using this object, you must first obtain a DtoProtocolStats Collection using the DtoMonitor Object and the DtoCommStat Object

All the properties for this object return values of type Long integer.

Examples

To obtain the number of requests processed using this protocol:

```
num_requests = Object.RequestsProcessed
```

To obtain the current number of remote sessions:

```
RemoteSess_count = Object.CurRemoteSessions
```

See Also

DtoProtocolStats Collection

DtoCommStat Object

DtoMonitor Object

DtoSqlClients Collection

A collection of DtoSqlClient objects, representing all the SQL clients on a server.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

None

Remarks

Note: This collection cannot be obtained by a Workstation engine.

Use the Count property to find the number of members in the collection.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get SQL Clients from monitor
Dim my_sqlclients as DtoSqlClients
Set my_sqlclients = my_monitor.SqlClients
```

See Also

DtoSqlClient Object

DtoMonitor Object

DtoSqlClient Object

Allows you to query information about and disconnect a SQL client.

Properties

AppDesc	Returns a description of the application that created a SQL client.
ConnectTime	Returns connection time for the SQL client.
CurStatusTime	Returns time since the last status.
DSN	Returns DSN associated with a SQL Client.
HostName	Returns host name for a SQL client.
IP	Returns IP for a SQL Client.
ProcessID	Returns process ID for a SQL client.
Status	Returns status of a SQL client.
ThreadId	Returns thread identification of a SQL client
UserName	Returns user name associated with the SQL client.

Methods

Disconnect method

Remarks

Use the DtoSqlClients Collection to obtain all the current SQL clients.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor object from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get SQL Clients from monitor
Dim my_sqlclients as DtoSqlClients
Set my_sqlclients = my_monitor.SqlClients
```

```
' Pick first client from collection and find
' the DSN associated with it
Dim first_sqlclient as DtoSqlClient
Dim DSNname as string
Set first_sqlclient = my_sqlclients(1)
DSNname = first_sqlclient.DSN
```

See Also

DtoSqlClients Collection

DtoMonitor Object

Methods Detail

Disconnect method

Disconnects a specific SQL Client.

Syntax

```
result = Object.Disconnect
```

Arguments

Object	DtoSqlClient object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Call this method for every SQL client you wish to disconnect.

Example

```
result = my_sqlclient.Disconnect
```

DtoMkdeVersion Object

DTO2 only: An object representing the version of the MicroKernel Engine.

Properties

MajorVersion	Returns major version of the engine.
MinorVersion	Returns minor version of the engine.
BuildNumber	Build number of the MicroKernel Engine release
OSTarget	Target operating system returns NTSV for Windows and UXSV for Unix systems.

Methods

None

Remarks

You can obtain a DtoMkdeVersion object through the properties of the DtoMonitor Object.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get monitor from session
Dim my_monitor as DtoMonitor
Set my_monitor = my_session.Monitor

' Now get the MkdeVersion object from monitor
Dim my_mkdeversion as DtoMkdeVersion
MajorVer = my_mkdeversion.MajorVersion
```

See Also

DtoMonitor Object

DtoEngineInformation Object

DTO2 only: An object representing information about the database engine.

Properties

MajorVersion	Returns major version of the engine.
MinorVersion	Returns minor version of the engine.
dbuApiVersion	Version of the DTI/DTO interface
IsServerEngine	Returns true if target is a server engine (as opposed to a workgroup engine)
ServerClientType	Returns one of the following: 0 = UNKNOWN_ENGINE_CLIENT 1 = NT_SERVER 3 = WIN32_CLIENT 4 = UNIX_SERVER 5 = CACHE_ENGINE_CLIENT 6 = VXWIN_SERVER 7 = VXLINUX_SERVER 9 = REPORT_ENGINE

Methods

None

Remarks

You can obtain a DtoEngineInformation object through the properties of the DtoSession Object.

Example

```
' Instantiate session and connect
Dim my_session as new DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' Get engine information from session
Dim my_engineInfo as DtoEngineInformation
Set my_engineInfo = my_session.EngineInformation

' Now get the client type from the engine info object
clientType = my_engineInfo.ServerClientType
```

See Also

DtoSession Object

Creating and Maintaining Catalogs and Dictionaries with DTO

The following topics provide information about the objects that comprise the Catalog group of the Distributed Tuning Objects:

- DtoDatabases Collection
- DtoDatabase Object
- DtoDSNs Collection
- DtoDSN Object
- DtoDictionary Object
- DtoTables Collection
- DtoTable Object
- DtoColumns Collection
- DtoColumn Object
- DtoIndexes Collection
- DtoIndex Object
- DtoSegments Collection
- DtoSegment Object

DtoDatabases Collection

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoDatabases collection.

Methods

Add method

Remove method

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' instantiate session object and connect to a server
Dim my_session as New DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' now use your session object to obtain db collection
Dim my_databases as DtoDatabases
Set my_databases = my_session.Databases
```

See Also

DtoDatabase Object

DtoSession Object

Methods Detail

Add method

Add an item to a DtoDatabases collection.

Syntax

```
result = Collection.Add(Object,[dsnName])
```

Arguments

Collection	DtoDatabases collection to which to add object.
Object	A new DtoDatabase object
dsnName	Optional. Will create a standard server DSN for the new database

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type object. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

This method will add the given database to the collection and to the underlying DBNAMES.CFG file on the server.

Example

```
Dim result As dtoResult
Dim database As DtoDatabase

Set database = New DtoDatabase
' Set properties for new database
database.Name = "MyDemodata"
database.DdfPath = "C:\test"
database.DataPath = "C:\test"
database.Flags = dtoDbFlagCreateDDF + dtoDbFlagRI

result = my_session.Databases.Add(database)
If NOT result = Dto_Success Then
   MsgBox "Error"+ Session.Error(result)
End If
```

Remove method

Removes an item from a DtoDatabases collection.

Syntax

```
Collection.Remove(database,[deleteDDF])
```

Arguments

Collection	Collection from which to remove object.
database	A variant that can contain the index (starting with 1) or the database name of the item you wish to remove from the collection
deleteDDF	Set to True to delete dictionary files. Set to False to leave dictionary files intact.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method removes the item from the databases collection and from the underlying DBNAMES.CFG file.

Example

```
Dim result As dtoResult
result = my_session.Databases.Remove("MyDemodata",1)
If NOT result = Dto_Success Then
   MsgBox "Error"+ my_session.Error(result)
End If
```

DtoDatabase Object

Properties

DataPath	Gets or sets the location of the data for a database.
DbCodePage	Gets or sets the database code page. This property is an enumeration. See Database Code Page for a list of values. The value zero specifies the server encoding (the code page of the server running the database engine).
DbFlags	Gets or sets the database flags for a database. This property is an enumeration. See Database Flags for a list of values.
DdfPath	Gets or sets the dictionary path for a database.
Name	Gets or sets the name of the database.
Secured	Returns whether the database has security enabled. (0=unsecure, 1=secure)
Session	Gets or sets the Session object associated with this DtoDatabase object.

Collections

DtoTables Collection

Methods

AddUserToGroup

AlterUserName

AlterUserPassword

Close

Copy

CreateGroup

CreateUser

DropGroup

DropUser

Open

RemoveUserFromGroup

Secure

UnSecure

Remarks

The Secure and UnSecure methods can only be used when the database is closed.

Examples

The following example shows how to instantiate the session object and connect to the server.

```
' instantiate session object and connect to server
Dim my_session as New DtoSession
Dim result as DtoResult
result = my_session.Connect("myserver", "username", "password")

' now use your session object to obtain db collection
Dim my_databases as DtoDatabases
Set my_databases = my_session.Databases

' get first database and query its dictionary path
Dim first_database as DtoDatabase
Dim dictionarypath as string
Set first_database = my_databases(1)
dictionarypath = first_database.DdfPath
```

The following example shows how to retrieve and set a code page using the DBCodePage property with the "Demodata" sample database.

```
Dim m_dtoSession1 As New DtoSession
Dim result As dtoResult
result = m_dtoSession1.Connect("localhost", "", "")
Dim sCodePage As String
sCodePage = m_dtoSession1.Databases("DEMODATA").DBCodePage
MsgBox "Code Page for database (before change): " & CStr(sCodePage)
If result = Dto_Success Then
Rem Set the code page for the database by passing in
Rem the code page number (for example, 0, 932, 1252,
Rem and so forth).
m_dtoSession1.Databases("DEMODATA").DBCodePage = 0
End If
MsgBox "Code Page for database: " & CStr(m_dtoSession1.Databases("DEMODATA").DBCodePage)
m_dtoSession1.Disconnect
```

See Also

DtoDatabases Collection

Methods Detail

AddUserToGroup

Adds an existing user to an existing group in the database.

Syntax

```
result = Object.AddUserToGroup(username, groupname)
```

Arguments

Object	Dtodatabase object
username	User name to add to the group
groupname	Group name to which the user is added

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

This function fails if the specified group or user do not already exist in the database, or if the user is a member of another group.

The following preconditions must be met

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- The user and group already exist in the specified database.
- The user is not a member of another group.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function AddUserToGroup(sUserName As String, sGroupName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
```

```
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's add the user to the group
    res = m_dbn.AddUserToGroup(sUserName, sGroupName)
    If res <> Dto_Success Then
        LogResult ("Error on Add User to Group: " & CStr(res))
    Else
        LogResult ("User " & sUserName & " added to group " & sGroupName & ".")
End If
End If
m_dbn.Close
End Function
```

AlterUserName

Alters an existing user's name in the specified database.

Syntax

```
result = Object.AlterUserName(username, new_username)
```

Arguments

Object	Dtodatabase object
username	Name of existing database user
new_username	New name for the database user. If set to NULL, the function fails.

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- The user name must already exist in the specified database.
- The new user name cannot already exist in the specified database.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function AlterUserName(sUserName As String, sNewUserName As String) As Boolean
Dim res As dtoResult
```

```
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's alter the username
    res = m_dbn.AlterUserName(sUserName, sNewUserName)
    If res <> Dto_Success Then
        LogResult ("Error on Alter Username: " & CStr(res))
     Else
        LogResult ("Alter Username completed. New user name is " & sNewUserName)
    End If
End If
m_dbn.Close
End Function
```

AlterUserPassword

Alters an existing user's password.

Syntax

```
result = Object.AlterUserPassword(username, new_password)
```

Arguments

Object	Dtodatabase object
username	Name of the database user whose password is to be changed
new_password	New password for the user. If set to NULL, the password is cleared.

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- The user name must already exist in the specified database.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function AlterUserPassword(sUser As String, sNewPassword As String) As Boolean
Dim res As dtoResult
```

```
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's alter the user's password
    res = m_dbn.AlterUserPassword(sUser, sNewPassword)
    If res <> Dto_Success Then
        LogResult ("Error on Alter User Password: " & CStr(res))
    Else
        LogResult ("Alter User Password successful.")
    End If
End If
m_dbn.Close
End Function
```

Close

Closes a set of data dictionary files that were opened using the *Open* method.

Syntax

```
result = Object.Close
```

Arguments

Object	DtoDatabase object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Call this method after the database has been opened using *Open* method. Error information can be obtained using *Error* property.

Example

```
Dim m_database as new DtoDatabase
Dim result as DtoResult

result = m_database.Open(“dbuser”,”pwd”)

' perform operations here
'
result = m_database.Close
```

Copy

Creates a new database based on the current one.

Syntax

```
result = Object.Copy(username, password, newDBname, newDictionaryPath, newDataPath)
```

Arguments

Object	DtoDatabase object
username	Database user name for the database. If the database does not have security enabled, set to an empty string.
password	Password for database user. If the database does not have security enabled, set to an empty string.
newDBname	Database name for the copied database.
newDictionaryPath	Absolute path to the directory in which the dictionary files are to be created. This directory must already exist.
newDataPath	Data path for the database. Pass an empty string to use the default data path (that is, the same as the dictionary path)

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Referential integrity is preserved in the copied database.

More information on the errors returned by the method can be obtained using the *Error* property.

Example

```
Dim Database As New DtoDatabase
Dim result as DtoResult
Database.Session = my_session ' assume session exists
Database.Name = “DEMODATA”
‘ no user name or password, unsecure database
result = Database.Copy("", "", "DEMODATA2", "D:\DEMODATA2", "D:\DEMODATA2")

If NOT result = Dto_Success Then
  MsgBox "Error"+ Session.Error(result)
End If
```

CreateGroup

Creates a new user group in the existing database.

Syntax

```
result = Object.CreateGroup(groupname)
```

Arguments

Object	Dtodatabase object
groupname	Name of the group that you want to add to the database

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- A group with the same name cannot already exist in the specified database.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function CreateGroup(sGroupName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's create the user.
    res = m_dbn.CreateGroup(sGroupName)
    If res <> Dto_Success Then
        LogResult ("Error on Create Group: " & CStr(res))
    Else
        LogResult ("Group " & sGroupName & " created.")
    End If
End If
m_dbn.Close
End Function
```

CreateUser

Creates a new user in an existing database. Optionally sets a password and assign the new user to an existing group.

Syntax

```
result = Object.CreateUser(username, [password], [groupname])
```

Arguments

Object	Dtodatabase object
username	Name of the user to add to the database
password	User password. If set to NULL, no password is set.
groupname	Database group name to which to assign the user. If set to NULL, user is not assigned to a group.

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

The following preconditions must be met:

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- A user with the same name cannot already exist in the specified database.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function CreateUser(sUserName As String, sPassword As String, sGroupName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's create the user.
    res = m_dbn.CreateUser(sUserName, sPassword, sGroupName)
    If res <> Dto_Success Then
        LogResult ("Error on Create User: " & CStr(res))
```

```
    Else
        LogResult ("User " & sUserName & " created in group " & sGroupName & ".")
    End If
End If
m_dbn.Close
End Function
```

DropGroup

Removes an existing group from the database.

Syntax

```
result = Object.DropGroup(groupname)
```

Arguments

Object	Dtodatabase object
groupname	Name of the group that you want to remove from the database

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- A group with the same name cannot already exist in the specified database.
- The group cannot contain any members.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function DropGroup(sGroupName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's drop the group.
    res = m_dbn.DropGroup(sGroupName)
    If res <> Dto_Success Then
        LogResult ("Error on Drop Group: " & CStr(res))
```

```
    Else
        LogResult ("Group " & sGroupName & " dropped.")
    End If
End If
m_dbn.Close
End Function
```

DropUser

Removes an existing user from the database.

Syntax

```
result = Object.DropUser(username)
```

Arguments

Object	Dtodatabase object
username	Name of the user that you want to remove from the database

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- A user with the same name must already exist in the specified database.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function DropUser(sUserName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's drop the user.
    res = m_dbn.DropUser(sUserName)
    If res <> Dto_Success Then
        LogResult ("Error on Drop User: " & CStr(res))
     Else
        LogResult ("Drop User for " & sUserName & " completed.")
```

```
End If
End If
m_dbn.Close
End Function
```

Open

Opens a connection to the database with the given username and password.

Syntax

```
result = Object.Open(username, password)
```

Arguments

Object	DtoDatabase object
username	User name for the database. If database is not secured, set to an empty string.
password	Password for the database. If database is not secured, set to an empty string

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This operation is used in order to open a set of dictionary files. This set must contain FILE.DDF, INDEX.DDF and FIELD.DDF. It may also contain a number of optional DDF files. Remember to call the Close method to free memory. Once the database is opened, no one else can make changes to it until the Close method is called.

You cannot issue the Secure or UnSecure methods while the database is open.

More information on the errors returned by the method can be obtained using the *Error* property of the DtoSession Object.

Example

```
Dim m_session as new DtoSession
Dim m_database as new DtoDatabase
Dim result as DtoResult
result = m_session.Connect("myserver","user","pwd")
m_database.Session = m_session
m_database.Name = "DEMODATA"
result = m_database.Open("dbuser","pwd")
```

RemoveUserFromGroup

Removes an existing user from an existing group.

Syntax

```
result = Object.RemoveUserFromGroup(groupname, username)
```

Arguments

Object	Dtodatabase object
groupname	Database group name
username	Database user name

Return Values

result	DtoResult long value indicating the result of the method call.

Remarks

- You must first create a session, then Open a database successfully as user Master.
- The associated database has database-level security enabled.
- The user and group already exist in the specified database.
- The user is not a member of another group.

The following post condition must be met:

- Close the database to free the resources.

Example

```
Function RemoveUserFromGroup(sUserName As String, sGroupName As String) As Boolean
Dim res As dtoResult
Dim m_dbn As New DtoDatabase
Dim m_dbn.Session = m_dto
Dim m_dbn.Name = "demodata"
res = m_dbn.Open("Master", "1234")
If res = Dto_Success Then
    'Open worked, let's remove the user from the group
    res = m_dbn.RemoveUserFromGroup(sGroupName, sUserName)
    If res <> Dto_Success Then
        LogResult ("Error on Remove User From Group: " & CStr(res))
     Else
        LogResult ("Remove user " & sUserName & " from group " & sGroupName & " completed.")
    End If
End If
m_dbn.Close
```

```
End Function
```

Secure

Enables security for a database.

Syntax

```
result = Object.Secure(user, password)
```

Arguments

Object	DtoDatabase object
user	User should be set to Master for securing the database.
password	Password for the Master user.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

When you enable database security, you must specify Master as the database user name and choose a password. Security for the database is enforced based on the access rights defined for the database, and should match behavior seen in SQL or ODBC access methods.

Ensure that the database is closed when attempting to enable security.

More information on the errors returned by the method can be obtained using the *Error* property of the DtoSession Object.

Example

```
Dim m_database as new DtoDatabase
Dim result as DtoResult
m_database.Name = "DEMODATA"
m_database.Session = my_session ' assume session exists
result = m_database.Secure("Master", "password")
```

UnSecure

Disables security for a database.

Syntax

```
result = Object.UnSecure(user, password)
```

Arguments

Object	DtoDatabase object
user	User should be set to “Master” to unsecure the database.
password	Password for the Master user.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

When you disable database security, you must specify Master as the database user name and provide the Master user password.

Ensure that the database is closed when attempting to disable security.

More information on the errors returned by the method can be obtained using the *Error* property of the DtoSession Object.

Example

```
Dim m_database as new DtoDatabase
Dim result as DtoResult
m_database.Name = "DEMODATA"
m_database.Session = my_session ' assume session exists
result = m_database.UnSecure("Master", "password")
```

DtoDSNs Collection

A collection of DtoDSN objects.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoDSNs collection.

Methods

Add method

Remove method

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' instantiate session object
Dim my_session as New DtoSession
Dim result as DtoResult

' connect to a server
result = my_session.Connect("myserver", "username", "password")

' now get DSNs collection
Dim my_dsns as DtoDSNs
Set my_dsns = my_session.DSNs
```

See Also

DtoDSN Object

DtoSession Object

Methods Detail

Add method

Add an item to a DtoDSNs collection and creates a DSN on the server.

Syntax

```
result = Collection.Add(Object)
```

Arguments

Collection	DtoDSNs collection to which to add object.
Object	A new DtoDSN object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type object. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

Example

```
Dim result As dtoResult
Dim DSNs As DtoDSNs
Dim dsn As DtoDSN

Set dsn = New DtoDSN

' Set properties for new dsn
dsn.Name = "MyDemodata_DSN"
dsn.Description = "a sample DSN"
dsn.Dbname = "MyDemodata"
dsn.Openmode = dtoNormalDSNOpenMode

result = my_session.DSNs.Add(dsn)
If NOT result = Dto_Success Then
  MsgBox "Error"+ my_session.Error(result)
End If
```

Remove method

Removes a DSN item from a DtoDSNs collection and deletes it from the server.

Syntax

```
result = Collection.Remove(dsn)
```

Arguments

Collection	Collection from which to remove object.
dsn	A variant containing either the index (starting with 1) or the name of the item you wish to remove from the collection

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method does not remove the associated database or database name.

Example

```
Dim result As dtoResult
Dim DSNs As DtoDSNs
result = my_session.DSNs.Remove("MYDSN")
If NOT result = Dto_Success Then
  MsgBox "Error"+ my_session.Error(result)
End If
```

DtoDSN Object

An object representing a Zen DSN.

Properties

DbName	Gets or sets the Dbname associated with the DSN.
Description	Gets or sets the description of the DSN.
Name	Gets or sets the name of the DSN.
OpenMode	Gets or sets the open mode enumeration of the DSN. See DSN Open Mode for possible values.
Session	Gets or sets the DtoSession object associated with this DtoDSN object.
Translate	Gets or set the encoding translation, which specifies how character data is translated between the database engine and a client application. This property is an enumeration. See DSN Translate Option for a list of values.

Methods

None

Remarks

To obtain information about a particular database, use the DtoDatabase Object.

Examples

To Query the DbName Associated with a DSN

```
' instantiate session object
Dim my_session as New DtoSession
' connect to a server
my_session.Connect("myserver", "username", "password")
' now use your session object to obtain db collection
my_dsns = my_session.DSNs
first_dsn = my_dsns.Item(1)
dsn_dbname = first_dsn.DbName
```

To Add a New DSN

```
' instantiate session object
```

```
Dim my_session as New DtoSession
Dim result as dtoResult

' connect to a server
result = my_session.Connect("myserver", "username", "password")
' now use your session object to obtain DSN collection
Dim my_dsns as DtoDSNs
Set my_dsns = my_session.DSNs

' Populate new DtoDSN object
Dim NewDSN as New DtoDSN
NewDSN.DbName = "DEMODATA"
NewDSN.Description = "A DSN for the DEMODATA db"
NewDSN.Name = "Demodata_DSN"

' now add the new DSN to the collection
result = my_dsns.Add(NewDSN)
```

To Get or Set the Encoding Translation

```
Dim m_dtoSession1 As New DtoSession
Dim result As dtoResult
result = m_dtoSession1.Connect("localhost", "", "")
Dim sTranslate As String
Dim iTranslate As Integer
iTranslate = m_dtoSession1.DSNs("DEMODATA").Translate
If iTranslate = 0 Then sTranslate = "None"
If iTranslate = 1 Then sTranslate = "OEM/ANSI Conversion"
If iTranslate = 2 Then sTranslate = "Automatic"

MsgBox "DSN Translate Setting (before change): " & sTranslate
If result = Dto_Success Then
Rem set the encoding translation.
m_dtoSession1.DSNs("DEMODATA").Translate = 1
End If
iTranslate = m_dtoSession1.DSNs("DEMODATA").Translate
If iTranslate = 0 Then sTranslate = "None"
If iTranslate = 1 Then sTranslate = "OEM/ANSI Conversion"
If iTranslate = 2 Then sTranslate = "Automatic"

MsgBox "DSN Translate Setting (after change): " & sTranslate
m_dtoSession1.Disconnect
```

See Also

DtoDSNs Collection

DtoSession Object

DtoDictionary Object

An object representing a Zen dictionary. This object is deprecated in favor of DtoDatabase Object. DtoDictionary can still be used only if you can specify the path to the dictionary on the Open method.

Properties

Path	Returns path of the dictionary object.

Collections

DtoTables Collection

Methods

Open method

Create method

Close method

AddTable method

DropTable method

Reload method

Delete method

Remarks

All of the operations affecting dictionary files have to be done through this object. The user can open a dictionary, create a dictionary, get table information add a table or drop a table using this object.

Note: If instantiating this object using ASP or if you use the CreateObject method in Visual Basic, the progid of DtoDictionary is "DTO.DtoDictionary.2" for DTO2, or “DTO.DtoDictionary.1” for DTO version 1. See DTO2 for more information on the differences between the two versions.

Example

```
Dim result as DtoResult
Dim dictionary as New DtoDictionary
result = dictionary.Open("d:\MyDemodata")
```

See Also

DtoTables Collection

DtoTable Object

Methods Detail

Open method

Opens a set of data dictionary files using either a database name or a dictionary path.

Syntax

```
result = Object.Open(path, [user], [password])
```

Arguments

Object	DtoDictionary object
path	Absolute path to the directory containing dictionary files or name of the named database if local. You cannot use a named database for this argument if you are connected to a remote server.
user	Optional user name for the DDF set
password	Optional password for the DDF set

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Note: The *path* argument may either contain a path to the directory containing the DDF files or it can use a name of the database contained in the local DBNAMES.CFG. See DtoDatabases Collection in order to create and maintain database names.

This operation is used in order to open a set of dictionary files. This set must contain FILE.DDF, INDEX.DDF and FIELD.DDF. It may also contain a number of optional DDF files. Remember to call the Close method to free memory. Once the dictionary set is opened no one else can make changes to it until the Close method is called.

More information on the errors returned by the method can be obtained using the *Error* property of the DtoSession Object.

Example

```
Dim dictionary as new DtoDictionary
Dim result as DtoResult

result = dictionary.Open("d:\MyDemodata")
```

Create method

Creates an empty set of data dictionary files.

Syntax

```
result = Object.Create(path, [user], [password])
```

Arguments

Object	DtoDictionary object
path	The absolute path to the directory in which the dictionary files to be created.
user	Optional user name for the DDF set.
password	Optional password for the DDF set.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

If the directory contained in *path* argument does not exist, an attempt will be made to create it. If operation is successful a set containing file.ddf, field.ddf, index.ddf will be created.

Remember to call *Close* method to free memory. Once the dictionary set is created other clients cannot open it or make changes to it until Close method is called.

More information on the errors returned by the method can be obtained using the *Error* property. Unlike Open method the path parameter can only contain an absolute path.

Example

```
Dim Dictionary As New DtoDictionary
Dim result as DtoResult

result = Dictionary.Create("C:\TEST", "login", "password")
If NOT result = Dto_Success Then
  MsgBox "Error"+ Session.Error(result)
End If
```

Close method

Closes a set of data dictionary files. Opened using *Open* method or created using *Create* method.

Syntax

```
result = Object.Close
```

Arguments

Object	DtoDictionary object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Call this method after the dictionary has been opened using *Open* method or created using *Create* method. Error information can be obtained using *Error* property.

Example

```
Dim dictionary as new DtoDictionary
Dim result as DtoResult
```

```
result = dictionary.Open("d:\MyDemodata")
'
' perform operations here
'
result = dictionary.Close
```

AddTable method

Adds table information to data dictionary files and creates a data file to match the definition.

Note: In the same directory, no two files should share the same file name and differ only in their file name extension. For example, do not create 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.

Syntax

```
result = Object.AddTable(table)
```

Arguments

Object	DtoDictionary object
table	DtoTable object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method adds the table definition to the ddf files and attempts to create the data file specified in Location property of table. If Location property is left blank then this method will attempt to create a data file named *tableName.mkd*. If a table with such name already exists a number will be appended to the name and another attempt will be made.

In order for this operation to complete successfully at least one column must be defined.

Example

The following example shows how to create a dictionary object and then add a table to it.

```
Dim Dictionary As New DtoDictionary
Dim Table As DtoTable
Dim Tables As DtoTables
Dim result As dtoResult
Dim Columns As DtoColumns
Dim Indexes As DtoIndexes
Dim Column As DtoColumn
Dim Index As DtoIndex
Dim Segments As DtoSegments
Dim Segment As DtoSegment

    result = Dictionary.Create("C:\TEST", "login", "password")
    If NOT result = Dto_Success Then
       MsgBox "Error"+ Session.Error(result)
    End If

' ******* Begin AddTable  ******************

    Set Table = New DtoTable

    Set Column = New DtoColumn
    With Column
    .Decimal = 0
    .Flags = dtoColumnNullable
    .ISR = ""
    .Name = "F_Int"
    .Number = 0
    .Size = 4
    .Type = dtoTypeInteger
    End With
    Table.Columns.Add Column

    Set Column = New DtoColumn
    With Column
    .Decimal = 4
    .Flags = dtoColumnNullable + dtoColumnCaseInsensitive
    .ISR = ""
    .Name = "F_Str"
    .Number = 1
    .Size = 55
    .Type = dtoTypeLString
    End With
    Table.Columns.Add Column

Set Column = New DtoColumn
    With Column
    .Decimal = 4
    .Flags = dtoColumnCaseInsensitive
    .ISR = ""
    .Name = "F_Str_Second"
    .Number = 2
    .Size = 100
    .Type = dtoTypeLString
    End With
    Table.Columns.Add Column

    Set Column = New DtoColumn
    With Column
    .Decimal = 10
    .Flags = dtoColumnDefault
    .ISR = ""
    .Name = "F_Float"
    .Number = 3
    .Size = 8
    .Type = dtoTypeBFloat
```

```
    End With
    Table.Columns.Add Column

'Add Indexes
    Set Index = New DtoIndex
    result = Index.AddSegment("F_Int", 0)
    Set Segment = New DtoSegment
    Segment.Number = 0
    Segment.ColumnName = "F_Int"
    Segment.Flags = dtoSegmentAscending
    Index.Segments.Add Segment

    Index.Name = "FintInd"
    Index.Number = 0
    Index.Flags = dtoIndexModifiable
    Table.Indexes.Add Index
'Add second Index:
    Set Index = New DtoIndex
    Set Segment = New DtoSegment
    Segment.Number = 0
    Segment.ColumnName = "F_Str"
    Segment.Flags = dtoSegmentAscending
    Index.Segments.Add Segment

    Set Segment = New DtoSegment
    Segment.Number = 1
    Segment.ColumnName = "F_Str_Second"
    Segment.Flags = dtoSegmentAscending
    Index.Segments.Add Segment

    Index.Name = "FStrTagInd"
    Index.Number = 1
    Index.Flags = dtoIndexModifiable
    Table.Indexes.Add Index

    Table.Overwrite = true
    Table.Flags = dtoTableTrueNullable
    Table.Name = "Table3"

    result = Dictionary.AddTable(Table)

    If NOT result = Dto_Success Then
       MsgBox "Error"+ Session.Error(result)
    End If
```

DropTable method

Remove a table from the current dictionary.

Syntax

```
result = Object.DropTable(tableName, [deleteFile])
```

Arguments

Object	DtoDictionary object.
tableName	Name of the table to be dropped.

deleteFile	A boolean value indicating whether the underlying data file should be deleted.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

Note that the dictionary has to be opened successfully for this operation to succeed.

Example

```
result = Dictionary.DropTable("Table3", true)
   If NOT result = Dto_Success Then
       MsgBox "Error"+ Session.Error(result)
   End If
```

Reload method

Refreshes a dictionary object.

Syntax

```
result = Object.Reload
```

Arguments

Object	DtoDictionary object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Delete method

Deletes dictionary object and the corresponding DDF files.

Syntax

```
result = Object.Delete
```

Arguments

Object	DtoDictionary object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

DtoTables Collection

Returns a collection of DtoTable objects.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection. You can pass an ordinal value or a table name.

Methods

None

Remarks

This collection only includes the user defined tables and does not include system tables. The dictionary has to be successfully opened. In order to add or drop tables from the collection use AddTable and DropTable.

Use the Count property to find the number of members in the DtoTables collection.

Example

Using DtoDatabase

Before obtaining the Tables collection, you must first invoke the Open method on the database object, even if the database is not secured.

```
Dim m_session as new DtoSession
Dim m_database as new DtoDatabase
Dim table as new DtoTable
Dim result as DtoResult

result = m_session.Connect("server","user","password")
m_database.Name = "demodata"
m_database.Session = m_session
' Open database, assuming database not secured
result = m_database.Open("","")

For each table in m_database.Tables
if table.Name = “Billing” then
' found the billing table
End If
next
```

```
m_database.Close ' close the db if you open it
```

Using DtoDictionary

```
Dim dictionary as new DtoDictionary
Dim table as new DtoTable
Dim result as DtoResult
Dim location as string

'find location of Mytable table
result = dictionary.Open("d:\MyDemodata")

For Each table In dictionary.Tables
   If table.Name = "Mytable" Then
      location = table.Location
      exit For
   End If
next
```

See Also

DtoDatabase Object

AddTable method

DropTable method

DtoTable Object

DtoTable Object

An object representing a table in a database.

Properties

Flags	Gets or sets the flags associated with this table. See Table Flags for possible values.
Location	Gets the file name of the table. To determine the path of this file, use the properties of the DtoDatabase Object.
Name	Gets or sets the name of the table.
Overwrite	If True, this table can overwrite a table with the same name during an AddTable method call. True = overwrite table False = do not overwrite table and return an error

Collections

DtoColumns Collection

DtoIndexes Collection

Methods

None

Remarks

DtoTable object contains two collection objects: Columns and Indexes. All operations involving columns and indexes are accomplished using these objects.

To add a new table to a dictionary, use the AddTable method.

To remove a table from a dictionary, use the DropTable method.

Example

For an example of creating a new DtoTable object, see AddTable method.

```
Dim dictionary as new DtoDictionary
```

```
Dim table as new DtoTable
Dim result as DtoResult
Dim location as string

'determine file name of Mytable table
result = dictionary.Open("d:\MyDemodata")

For Each table In dictionary.Tables
   If table.Name = "Mytable" Then
      location = table.Location
   End If
next
```

See Also

DtoTables Collection

DtoColumn Object

DtoIndex Object

DtoColumns Collection

A collection of DtoColumn objects representing all the columns in a table.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoColumns collection.

Methods

Add method

Remove method

Clear method

Remarks

You obtain this collection from a property of the DtoTable Object object.

Use the Count property to find the number of members in the collection.

Example

```
Dim dictionary as new DtoDictionary
dictionary.Open("d:\MyDemodata")
students_table = dictionary.GetTable("STUDENT")
students_cols = students_table.Columns
```

See Also

DtoIndexes Collection

DtoColumn Object

DtoTable Object

Methods Detail

Add method

Add an item to a DtoColumns collection.

Syntax

```
result = Collection.Add(Object)
```

Arguments

Collection	DtoColumns collection to which to add object.
Object	A new DtoColumn object.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type DtoColumn. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

Note: You cannot use this method to add columns to preexisting Zen tables. This method does not alter data or DDF files, and can only be used to add columns in memory before a table is created. See the example for AddTable method as a reference.

Remove method

Removes an item from a DtoColumns collection.

Syntax

```
Collection.Remove(column)
```

Arguments

Collection	DtoColumns collection from which to remove object.

column	A variant containing either the index (starting with 1) or the name of the item you wish to remove from the DtoColumns collection

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

You can pass either a column name or the 1-based ordinal of the item in the DtoColumns collection.

Note: You cannot use this method to remove columns from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove columns in memory before a table is created. See the example for AddTable method as a reference.

Clear method

Removes all items from a DtoColumns collection.

Syntax

```
result = Collection.Clear
```

Arguments

In	*Collection*	DtoColumns or DtoIndexes collection obtained from a DtoTable object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method removes all the columns from a table in memory.

Note: You cannot use this method to remove columns from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove all columns in memory before a table is created. See the example for AddTable method as a reference.

DtoColumn Object

This object represents a table column.

Properties

Decimal	Gets or sets the decimal places for a column
Flags	Gets or sets the flags for a column
ISR	Gets or sets the ISR for a column
Name	Gets or sets the Name for a column
Number	Gets or sets the ordinal of a column
Size	Gets or sets the size of the column
Type	Gets or sets column type. An enumerated value. See Btrieve Types for possible values.
TypeName	Returns a string value containing the type name.

Methods

None

Remarks

This object allows you to display properties of a specific table column.

Example

```
' Instantiate and open dictionary
Dim dictionary as new DtoDictionary
Dim result as DtoResult
result = dictionary.Open("d:\MyDemodata")

' Get STUDENT table from MyDemodata database
Dim students_table as DtoTable
Set students_table = dictionary.Tables("STUDENTS")

' Get columns collection from STUDENT table
Dim students_cols as DtoColumns
Set students_cols = students_table.Columns

' Retrieve first column and get its name
Dim first_col as DtoColumn
Set first_col = students_cols(1)
name = first_col.Name
```

See Also

DtoColumns Collection

DtoTable Object

DtoIndexes Collection

A collection of DtoIndex objects representing the indexes of a table.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a DtoIndexes collection. You can pass either a 1-based ordinal or the name of the Index.

Methods

Add method

Remove method

Clear method

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' Instantiate and open dictionary
Dim dictionary as new DtoDictionary
Dim result as DtoResult
result = dictionary.Open("d:\mydemodata")

' Get STUDENT table from MYDEMODATA database
Dim students_table as DtoTable
Set students_table = dictionary.Tables("STUDENT")

' Retrieve DEMODATA indexes collection
Dim students_idx as DtoIndexes
Set students_idx = students_table.Indexes
```

See Also

DtoIndex Object

DtoTable Object

Methods Detail

Add method

Add an item to a collection.

Syntax

```
result = Collection.Add(Object)
```

Arguments

Collection	Collection to add object to.
Object	A new DtoIndex object to add to the DtoIndexes collection.

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type DtoIndex. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

Note: You cannot use this method to add indexes to preexisting Zen tables. This method does not alter data or DDF files, and can only be used to add indexes in memory before a table is created. See the example for AddTable method as a reference.

Remove method

Removes an item from a collection.

Syntax

```
result = Collection.Remove(index)
```

Arguments

Collection	Collection from which to remove object.
index	A variant containing either the index (starting with 1) or the name of the item you wish to remove from the DtoIndexes collection

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

You can pass either a name or a 1-based ordinal to the Remove method.

Note: You cannot use this method to remove indexes from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove indexes in memory before a table is created. See the example for AddTable method as a reference.

Clear method

Removes all items from a DtoColumns or DtoIndexes collection.

Syntax

```
result = Collection.Clear
```

Arguments

Collection	DtoIndexes collection obtained from a DtoTable object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method removes all the indexes from a table in memory.

Note: You cannot use this method to remove indexes from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove all indexes in memory before a table is created. See the example for AddTable method as a reference.

DtoIndex Object

This object represents an index for a table.

Properties

Flags	Gets or sets the column name for the index segment. This is an enumerated value. See Index Flags for possible values.
Name	Gets or sets the name of the index.
Number	Gets or sets the 0-based number of the index.
Tag	Gets the index tag. It contains the names of all the columns that comprise the index.

Collections

DtoSegments Collection

Methods

None

Remarks

Only 119 segments are allowed per index. Note that the combined size of all the columns in segments of an index cannot be more then 255 bytes.

Only the last column in an index segment can have a partial index flag. Index segments that are not the last segment in the index and that use the partial index flag will have the partial flag ignored.

Example

```
' Instantiate and open dictionary
Dim dictionary as new DtoDictionary
Dim result as DtoResult
result = dictionary.Open("d:\mydemodata")

' Get STUDENT table from MYDEMODATA database
Dim students_table as DtoTable
Set students_table = dictionary.Tables("STUDENT")

' Retrieve DEMODATA indexes collection
```

```
Dim students_idx as DtoIndexes
Set students_idx = students_table.Indexes

' Get first index and determine its name
Dim first_idx as DtoIndex
Set first_idx = students_idx(1)
Dim index_name as String
index_name = first_idx.Name
```

See Also

DtoIndexes Collection

DtoSegments Collection

DtoSegments Collection

A collection of DtoSegment objects representing the segments of an index.

Properties

Count	Returns number of members in a collection.
Item	Returns a specific member of a collection.

Methods

Add method

Remove method

Clear method

Remarks

Use the Count property to find the number of members in the collection.

Example

```
' Open dictionary
Dim dictionary as new DtoDictionary
Dim result as DtoResult
result = dictionary.Open("d:\mydemodata")

' Get Students table
Dim students_table as DtoTable
Set students_table = dictionary.GetTable("Student")

' Obtain indexes collection from students table
Dim students_idx as DtoIndexes
Set students_idx = students_table.Indexes

' Delete all the indexes
Dim first_idx as DtoIndex
Set first_idx = students_idx(1)

' Get DtoSegments collection from first_idx
Dim my_segments as DtoSegments
Set my_segments as first_idx.Segments
```

See Also

DtoSegment Object

DtoTable Object

Methods Detail

Add method

Add an item to a collection.

Syntax

```
result = Collection.Add(Object)
```

Arguments

Collection	DtoSegments collection to which to add object.
Object	A new DtoSegment object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method takes a parameter of type DtoSegment. Therefore, you are responsible for first instantiating the object and setting its properties before adding it to the collection.

Note: You cannot use this method to add segments to preexisting Zen tables. This method does not alter data or DDF files, and can only be used to add segments in memory before a table is created. See the example for AddTable method as a reference.

Remove method

Removes an item from a collection.

Syntax

```
result = Collection.Remove(segment)
```

Arguments

Collection	DtoSegments collection from which to remove object.
segment	A variant containing either the index (starting with 1) or the name of the item you wish to remove from the collection

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

You can pass either the 1-based ordinal or the name of the segment.

Note: You cannot use this method to remove segments from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove segments in memory before a table is created. See the example for AddTable method as a reference.

Clear method

Removes all items from a DtoSegments collection.

Syntax

```
result = Collection.Clear
```

Arguments

Collection	DtoSegments collection obtained from a DtoIndex object

Return Values

result	DtoResult long value indicating the result of the method call. Use the *Error* property of the DtoSession Object to obtain a description for the result.

Remarks

This method removes all the segments from an index in memory.

Note: You cannot use this method to remove segments from preexisting Zen tables. This method does not alter data or DDF files, and can only be used to remove segments in memory before a table is created. See the example for AddTable method as a reference.

DtoSegment Object

This object represents a segment in an index.

Properties

ColumnName	Gets or sets the column name associated with this segment.
Flags	Gets or sets segment flags. An enumerated value. See Segment Flags for possible values.
Number	Gets or sets the 0-based segment number.

Methods

None

Remarks

One or more of these comprise an index. Only 119 segments are allowed per index. Note that the combined size of all the columns in segments of an index cannot be more then 255 bytes.

Example

```
' Open dictionary
Dim dictionary as new DtoDictionary
Dim result as DtoResult
result = dictionary.Open("d:\mydemodata")

' Get Students table
Dim students_table as DtoTable
Set students_table = dictionary.GetTable("Student")

' Obtain indexes collection from students table
Dim students_idx as DtoIndexes
Set students_idx = students_table.Indexes

' Delete all the indexes
Dim first_idx as DtoIndex
Set first_idx = students_idx(1)

' Get DtoSegments collection from first_idx
Dim my_segments as DtoSegments
Set my_segments as first_idx.Segments

' Get first segment and query column name
Dim first_seg as DtoSegment
Set first_seg = my_segments(1)
Dim colname as String
colname = first_seg.ColumnName
```

See Also

DtoSegments Collection

DtoTable Object

DtoIndexes Collection

Distributed Tuning Objects Enumerations

The following topic provides information about the enumerations used in the Distributed Tuning Objects:

Enumerated Types in DTO

DTO supports the following enumerated types:

- Service ID
- Service Status

Btrieve Types

Enumeration	Value
0	dtoTypeString
1	dtoTypeInteger
2	dtoTypeFloat
3	dtoTypeDate
4	dtoTypeTime
5	dtoTypeDecimal
6	dtoTypeMoney
7	dtoTypeLogical
8	dtoTypeNumeric
9	dtoTypeBfloat
10	dtoTypeLString
11	dtoTypeZString
12	dtoTypeNote
13	dtoTypeLvar
14	dtoTypeBinary
15	dtoTypeIdentity
16	dtoTypeBit
17	dtoTypeNumericSTS
18	dtoTypeNumericSA
19	dtoTypeCurrency
20	dtoTypeTimestamp
21	dtoTypeBlob
22	dtoTypeGDecimal

Enumeration	Value
25	dtoTypeWString
26	dtoTypeWZString
27	dtoTypeGUID
30	dtoTypeDateTime

Column Flags

Enumeration	Value
0	dtoColumnDefault
1	dtoColumnCaseSensitive
4	dtoColumnNullable
256	dtoColumnBinary
2048	dtoTypeColumnNText
4096	dtoTypeColumnBinary

Index Flags

Enumeration	Value
0	dtoIndexDefault
1	dtoIndexDuplicatesAllowed
2	dtoIndexModifiable
64	dtoIndexDescending
512	dtoIndexPartial

Segment Flags

Enumeration	Value
0	dtoSegmentAscending
64	dtoSegmentDescending

Table Flags

Enumeration	Value
0	dtoTableLegacy
64	dtoTableTrueNullable

DtoResult

Enumeration	Value
0	Dto_Success
1	Dto_errFailed
2	Dto_errMemoryAllocation
3	Dto_errDictionaryNotFound
4	Dto_errDictionaryAlreadyOpen
5	Dto_errDictionaryNotOpen
6	Dto_errInvalidDictionaryHandle
7	Dto_errTableNotFound
8	Dto_errInvalidTableName
9	Dto_errInvalidColumnName
10	Dto_errInvalidColumnDataType
11	Dto_errDuplicateColumnName
12	Dto_errInvalidDataSize
13	Dto_errInvalidColumnOrder
14	Dto_errInvalidIndexName
15	Dto_errColumnNotFound
16	Dto_errTooManySegments
17	Dto_errStringTooShort
18	Dto_errDictionaryAlreadyExists
19	Dto_errDirectoryError

Enumeration	Value
20	Dto_errSessionSecurityError
21	Dto_errDuplicateTable
22	Dto_errDuplicateIndex
27	Dto_errInvalidNameLength
28	Dto_errInternalProtocolError
29	Dto_errInvalidAccountName
30	Dto_errUserAlreadyExists
31	Dto_errGroupNotEmpty
32	Dto_errGroupAlreadyExists
33	Dto_errUserAlreadyPartOfGroup
34	Dto_errUserNotPartOfGroup
35	Dto_errNotAllowedToDropAdministrator
36	Dto_errDatabaseHasNoSecurity
37	Dto_errInvalidPassword
38	Dto_SuccessWithInfo
87	Dto_errServiceInvalidParameter
123	Dto_errInvalidServiceName
161	Dto_errMaxUserCountReached
423	Dto_errInvalidSession
424	Dto_errInvalidArgument
425	Dto_errNotConnected
426	Dto_errInvalidComputerName
427	Dto_errUnknownError
428	Dto_errTableCouldNotBeDeleted
429	Dto_errItemNotFound
430	Dto_errAPINotImplemented
431	Dto_errAccessDenied

Enumeration	Value
1051	Dto_errServiceDependentServiceRunning
1052	Dto_errServiceInvalidServiceControl
1053	Dto_errServiceRequestTimeout
1055	Dto_errServiceDatabaseLocked
1056	Dto_errServiceAlreadyRunning
1057	Dto_errInvalidServiceAccount
1058	Dto_errServiceDisabled
1059	Dto_errServiceCircularDependency
1060	Dto_errServiceDoesNotExist
1062	Dto_errServiceNotActive
1065	Dto_errServiceDatabaseDoesNotExist
1068	Dto_errServiceDependencyFail
1069	Dto_errServiceLogonFailed
1072	Dto_errServiceMarkedForDelete
1075	Dto_errServiceDependencyDeleted
7001	Dto_errInvalidHandle
7002	Dto_errNullPointer
7003	Dto_errBufferTooSmall
7004	Dto_errDtiFailed
7005	Dto_errInvalidDataType
7006	Dto_errOutOfRange
7007	Dto_errInvalidSelection
7008	Dto_errInvalidSequence
7009	Dto_errDataUnavailable
7010	Dto_errInvalidClient
7011	Dto_errAccessRights
7012	Dto_errDuplicateName

Enumeration	Value
7013	Dto_errDatabaseDoesNotExist
7015	Dto_errFileNotOpen
7016	Dto_errDDFAlreadyExist
7017	Dto_errSharedDDFExist
7018	Dto_errInvalidName
7019	Dto_errDSNAlreadyExist
7020	Dto_errDSNDoesNotExist
7021	Dto_errInvalidOpenMode
The following enumerations are only present in DTO2	
7063	See 161
7064	Dto_errNoLicenseObtained
7065	Dto_errNoProductObtained
7101	Dto_errInvalidLicKeyCharacter
7102	Dto_errIllegalLicType
7108	Dto_errLicKeyTooLong
7109	Dto_errLicNotFound
7110	Dto_errLicExpired
7111	Dto_errLicIsTemporary
7112	Dto_errLicAlreadyInstalled
7113	Dto_errLicInvalid
7115	Dto_errInvalidProductId
7118	Dto_errServerNotRunning
7119	Dto_errLocalServerNotRunning
7120	Dto_errLicNotRemovable
7122	Dto_errNoActiveLicense

Setting Rank

Enumeration	Value
0	dtoNormal
1	dtoAdvanced

Setting Type

Enumeration	Value
0	dtoBooleanType
1	dtoLongType
2	dtoStringType
3	dtoSingleSel
4	dtoMultiSel

Client Site

Enumeration	Value
0	dtoClientSiteLocal
1	dtoClientSiteRemote

Client Platform

Enumeration	Value
0	dtoPlatformNotAvailable
1	dtoPlatformWin
2	dtoPlatformWin95
3	dtoPlatformWinWg
4	dtoPlatformNTW
5	dtoPlatformNTS
6	dtoPlatformNW

Enumeration	Value
7	dtoPlatformOS2W
8	dtoPlatformOS2S
9	dtoPlatformDOS

Transaction State

Enumeration	Value
0	dtoNone
19	dtoExclusive
1019	dtoConcurrent

Open Mode

Enumeration	Value
0	dtoNormalOpenMode
255	dtoAcceleratedOpenMode
254	dtoReadOnlyOpenMode
253	dtoVerifyOpenMode
252	dtoExclusiveOpenMode
248	dtoNormalNonTransOpenMode
247	dtoAcceleratedNonTransOpenMode
246	dtoReadOnlyNonTransOpenMode
245	dtoVerifyNonTransOpenMode
244	dtoExclusiveNonTransOpenMode
240	dtoNormalSharedLockingOpenMode
239	dtoAcceleratedSharedLockingOpenMode
238	dtoReadOnlySharedLockingOpenMode
237	dtoVerifySharedLockingOpenMode

Enumeration	Value
236	dtoExclusiveSharedLockingOpenMode

DSN Open Mode

Enumeration	Value
0	dtoNormalDSNOpenMode
1	dtoAcceleratedDSNOpenMode
2	dtoReadOnlyDSNOpenMode
3	dtoExclusiveDSNOpenMode

DSN Translate Option

Enumeration	Value
0	dtoDSNFlagDefault
1	dtoDSNFlagEomAnsi
2	dtoDSNFlagAuto

Lock Type

Enumeration	Value
0	dtoNotLocked
1	dtoSingleLock
2	dtoMultipleLock

Wait State

Enumeration	Value
0	dtoNotWaiting
1	dtoWaitingForRecordLock
2	dtoWaitingForFileLock

Database Code Page

Enumeration	Value
0	dtoDbZeroCodePage
65001	dtoDBCodePageUTF8

Database Flags

Enumeration	Value
0	dtoDbFlagNotApplicable
1	dtoDbFlagBound
2	dtoDbFlagRI
4	dtoDbFlagCreateDDF
32	dtoDbFlagLONGMETADATA

SQL Connection Status

Enumeration	Value
0	dtoSQLConnectionIdle
1	dtoSQLConnectionActive
2	dtoSQLConnectionDying

Service ID

Enumeration	Value
0	dtoServiceTransactional
1	dtoServiceRelational
2	dtoServiceIDS

Service Status

Enumeration	Value
0	dtoServiceStopped
1	dtoServiceStartPending
2	dtoServiceStopPending
3	dtoServiceRunning
4	dtoServiceContinuePending
5	dtoServicePausePending
6	dtoServicePaused
7	dtoServiceNotFound