Contents

For many years InterBase has had functionality that other databases still lack, or are only recently implementing. One of these features is Event Alerters. The Interbase Event Alerter mechanism provides a means whereby applications can respond to actions and database changes made by other, concurrently running applications, without having to resort to polling the database on a regular basis, or communicating directly with the other applications.

Unfortunately, the ability to harness the power of Event Alerters via the Borland Database Engine (BDE) does not exist. IBCtrls provides two InterBase specific components which provide the missing functionality.

TIBDatabase

The TIBDatabase component allows you to create and drop InterBase databases, manipulate the database metadata, and additionally provides a direct connection to an InterBase server for the TIBEventAlerter component.

TIBEventAlerter

The TIBEventAlerter component allows a Delphi application to register interest in, and asynchronously handle, events posted by an InterBase server.

TIBDatabase Component

properties   methods

Unit
IBCtrls.pas

Description:
The TIBDatabase component provides a direct connection to an InterBase database, bypassing the Borland Database Engine (BDE).   The TIBDatabase component provides functionality to create and drop InterBase databases, as well as create, drop and modify the database metadata such as stored procedures and triggers which cannot be manipulated via the BDE.

InterBase databases can be created at runtime by calling the CreateDatabase method, or at design time with the IBDatabase Editor.   Similarly InterBase, databases can be dropped with a call to DropDatabase or again with the IBDatabase Editor at design time.   The IBDatabase Editor can be invoked by double clicking on the IBDatabase component or by choosing IBDatabase Editor from the context sensitive menu available by right-clicking on the component.

InterBase metadata can be manipulated at runtime using the SQL property and the ExecSQL method. At design time the SQL Editor enables ad hoc DDL statements to be executed.   The SQL Editor can be invoked by right-clicking on the IBDatabase component and selecting SQL Editor from the context sensitive menu.

TIBEventAlerter Component

properties methods events

Unit
IBCtrls.pas

Description
TIBEventAlerter component allows a Delphi application to register interest in, and asynchronously handle, events posted by an InterBase server. The Interbase event mechanism provides a means whereby applications can respond to actions and database changes made by other, concurrently running applications, without having to resort to polling the database on a regular basis, or communicating directly with the other applications.

In essence, the TIBEventAlerter allows an application to say 'I want to be informed when events X, Y and Z occur'. The application then continues executing without polling the database checking for X, Y and Z. When any of the requested events does occur, the InterBase server will notify the application and OnEventAlert will be called to allow the event to be processed.

The procedure for utilising InterBase events is as follows:

1. Create a trigger or stored procedure on your InterBase server which will post an event.
2. Add an IBDatabase and an IBEventAlerter to your form.
3. Register the events you wish to be notified about.
4. Write an OnEventAlert event handler to handle incoming event notifications.

It is important to remember that InterBase posts events within the context of transactions - you must commit any transaction that posts an event for the client to be notified. In addition, InterBase consolidates events before posting them. For example, if an InterBase trigger posts 20 x STOCK_LOW events within a transaction, when the transaction is committed these will be consolidated into a single STOCK_LOW event, and the client will only receive one event notification.

16 bit local InterBase (InterBase16) users take heed.
While the IBDatabase component allows you to create and connect to InterBase16, you cannot use event alerters with InterBase16, as the server does not support them. If you attempt to register events with InterBase16 an exception will be raised indicating that the required functionality is not supported. The 32 bit version of local InterBase that ships with Delphi 2.0 does support event alerters.

AliasName property

Applies to

TIBDatabase component

Declaration:

```
property AliasName: string
```

Description:

AliasName is a convenience property.   Setting the AliasName property to a BDE alias will cause the IBDatabase component to extract the DatabaseName and default UserName from the Alias definition. This provides a mechanism to easily synchronise the TIBDatabase component with the standard Delphi TDatabase component. For example:

```
  IBDatabase1.AliasName := Database1.AliasName
```

If either the DatabaseName or UserName properties are modified after setting AliasName then AliasName is cleared, as the properties for the component no longer reflect the original   BDE alias.

properties

AliasName	Params
Connected	Password
DatabaseName	SQL
Handle	UserName

Connected property

Applies to
TIBDatabase

Declaration
```
property Connected: boolean
```

Description
The Connected property indicates whether or not a connection to an InterBase database has been established. In addition, setting Connected to true has the same effect as calling the Open method and will attempt to establish a connection with the specified InterBase database.

DatabaseName property

Applies to
TIBDatabase

Declaration
```
property DatabaseName: string
```

Description
The DatabaseName property determines two things: firstly, the name of the InterBase database that will be connected to; and secondly, what transport mechanism will be used to establish the connection (IPX/SPX, TCP/IP, Named Pipes/NetBEUI, Local InterBase). The DatabaseName property corresponds to the Server Name parameter in a BDE alias. DatabaseName would normally only be set directly in order to create or drop an InterBase database - usually you would set the AliasName property instead.

The rules for establishing connections are as follows.

TCP/IP
TCP/IP is the protocol of choice as it provides the fastest network performance. TCP/IP connections can be established with InterBase on Netware, Windows NT and Unix. To establish a connection via TCP/IP specify the name of the server as defined in the clients HOSTS file, followed by a colon (:), followed by the full path to the InterBase database file. For Windows NT you need to specify the drive name (as seen by the server), and on Netware you need to specify the Netware volume name as part of the file path. For Unix systems the file path is case sensitive. For example

```
winnt:c:\interbas\examples\employee.gdb
netware:sys:\interbas\examples\employee.gdb
unix:/interbas/examples/employee.gdb
```

IPX/SPX
IPX/SPX connections can only be established with InterBase on Netware. To establish a connection via IPX/SPX specify the name of the server, followed by an at symbol (@), followed by the full path to the InterBase database file. For example

```
netware@sys:\interbas\examples\employee.gdb
```

NetBEUI
NetBEUI connections can only be established with Windows NT servers. It is recommended that NetBEUI be only used as a last resort as NetBEUI performance is woeful compared with TCP/IP and IPX/SPX. To establish a connection via NetBEUI specify a double backslash (\\), followed by the server name, followed by a single backslash (\), followed by the full path to the InterBase database file. For example

```
\\winnt\c:\interbas\examples\employee.gdb
```

Local InterBase
To establish a connection with a local InterBase server simply specify the full path name to the InterBase database file. For example

```
c:\interbas\employee.gdb
```

It goes without saying, that you cannot only attempt to a connection with a given transport if you have the appropriate software installed on both client and server.

Handle property

Applies to
TIBDatabase

Declaration
```
property Handle: isc_db_handle
```

Description
Handle provides the database connection handle required by the TIBEventAlerter component when registering interest in InterBase events.   Handle is read only, and set when a connection is established with a server.

Params property

Applies to
TIBDatabase

Declaration
```
property Params: TStrings
```

Description
The Params string list is only used in the creation of InterBase databases. The Params are combined with the UserName and Password properties to allow the user to specify InterBase creation options such as page size, secondary files, default collation order and so on. For a detailed description of the InterBase creation parameters refer to the InterBase manuals. For example, Params could contain:

```
  PAGE_SIZE 2048
  DEFAULT CHARACTER SET "ISO8859_1"
  FILE "employee.gd1" STARTING AT PAGE 10001 LENGTH 10000 PAGES
```

Password property

Applies to
TIBDatabase

Declaration
```
property Password: string
```

Description
Password is the password that will be used when connecting to, creating or dropping an InterBase database.   A password must be supplied, and unlike DatabaseName and UserName it cannot be determined from a BDE alias.

UserName property

Applies to

TIBDatabase

Declaration

```
property UserName: string
```

Description

UserName is the user name that will be used when connecting to, creating or dropping an InterBase database.   UserName will be set for you if you specify a BDE alias in the AliasName property.

methods

Close

CreateDatabase

DropDatabase

ExecSQL

Open

Close method

Applies to
TIBDatabase

Declaration
```
procedure Close
```

Description
The Close method will close the connection between the InterBase database and the TIBDatabase component.   Connected will be set to false in the process.

CreateDatabase method

Applies to
TIBDatabase

Declaration
```
procedure CreateDatabase
```

Description
The CreateDatabase method will attempt to create an InterBase database as specified by the DatabaseName property.   You must provide a UserName and Password, and the IBDatabase component must be closed before calling CreateDatabase.

If for some reason the database creation fails, an exception will be raised.

DropDatabase method

Applies to
TIBDatabase

Declaration
```
procedure DropDatabase
```

Description
DropDatabase will attempt to drop an InterBase database.   In order to drop a database, a connection to the database must first be established, and DropDatabase will attempt to Open a connection to the specified database.   It is therefore a prerequisite to supply a UserName and Password in addition to DatabaseName before calling DropDatabase.

If for some reason (such as the database being in use) DropDatabase fails, an exception will be raised.

Open method

Applies to
TIBDatabase

Declaration
```
procedure Open
```

Description
The Open method will attempt to establish a connection with the specified InterBase database.   In order to succeed you must specify a DatabaseName, UserName and Password before calling Open.   If Open fails an exception will be raised with an appropriate error message.

Events property

Applies to
TIBEventAlerter

Declaration
`property Events: TStrings`

Description
The Events property contains the list of events that an IBEventAlerter component will respond to. A single IBEventAlerter can register interest in up to 15 events. If you need to respond to more that 15 events use more that one IBEventAlerter component. An exception will be raised if you attempt to add too many events at runtime. At design time the Events property editor will only allow a maximum 15 events to entered.

To add an event to the Events list use the following code

```
  IBEventAlerter.Events.Add( ‘STOCK_LOW’)
```

Note: event names are case-sensitive.

properties

Events

IBDatabase

Registered

IBDatabase property

Applies to

TIBEventAlerter

Declaration

```
property IBDatabase: TIBDatabase
```

Description

IBDatabase is a reference to the TIBDatabase component that will provide the InterBase connection required to register Events.

Registered property

Applies to
TIBEventAlerter

Declaration
```
property Registered: boolean
```

Description
Registered indicates whether any events are currently registered or not. Setting Registered to true will call RegisterEvents and register the events in the Events list. At design time, no event notifications will be received even if Registered is true.

OnEventAlert event

Applies to

TIBEventAlerter

Declaration

```
procedure OnEventAlert: TEventAlert
TEventAlert = procedure( Sender: TObject; EventName: string;
          EventCount: longint; var CancelAlerts: Boolean)
```

OnEventAlert is called every time an InterBase event is received by an IBEventAlerter component. The EventName variable contains the name of the event that has just been received. The EventCount variable contains the number of EventName events that have been received since OnEventAlert was last called.

To cancel interest in any further events, set CancelAlert = true. If you later decide that you want to receive events again, call the QueueEvents method.

events
OnEventAlert

methods

CancelEvents	RegisterEvents
QueueEvents	UnregisterEvents

CancelEvents method

Applies to
TIBEventAlerter

Declaration
```
procedure CancelEvents
```

Description
CancelEvents cancels interest in any pending InterBase events.   CancelEvents does not unregister the events and to restore interest in the events again simply call QueueEvents.

RegisterEvents method

Applies to
TIBEventAlerter

Declaration
`procedure RegisterEvents`

Description
RegisterEvents registers interest in the events listed in the Events property. RegisterEvents will call the QueueEvents method to start receiving event notifications.

RegisterEvent will raise an exception if IBDatabase is nil or if the IBDatabase component is not open. An exception will always be raised when running your application if, at design time, you place an IBEventAlerter on the form before the IBDatabase. This is due to the fact that when the form loads the IBEventAlerter will be created first and attempt to RegisterEvents before the IBDatabase has had a chance to connect to the InterBase server. To avoid this problem ensure that IBDatabase comes before any IBEventAlters in the form's creation order.

QueueEvents method

Applies to
TIBEventAlerter

Declaration
`procedure QueueEvents`

Description
QueueEvents is called to start receiving event notifications.

You must call RegisterEvents to specify which events you wish to receive before calling QueueEvents. If RegisterEvents has not been called an exception will be raised.

UnregisterEvents method

Applies to
TIBEventAlerter

Declaration
```
procedure UnregisterEvents
```

Description
UnregisterEvents calls CancelEvents to cancel any pending event notifications, and then unregisters interest in the events in the Events list.   When the IBEventAlerter component is destroyed UnregisterEvents will be called automatically.

SQL property

Applies to
TIBDatabase component

Declaration:
property SQL: TStrings

Description
The SQL property provides the ability to create ad hoc DDL statements which can then be executed by calling the ExecSQL method.   The combination of the SQL property and ExecSQL method enable the creation and manipulation of any InterBase metadata object including tables, indexes, stored procedures, triggers, domains, constraints etc. For example, to create a stored procedure that posts events:

```
with IBDatabase, SQL do
begin
  Clear;
  Add( 'create procedure Broadcast( event varchar(40)) as');
  Add( 'begin');
  Add( '  post_event :event;');
  Add( 'end');
  ExecSQL;
end;
```

Data Definition Language
SQL statements which define database metadata such as tables, indexes etc. For example, `create table Employee ...`

ExecSQL method

Applies to
TIBDatabase component

Declaration:
`procedure ExecSQL;`

Description
The ExecSQL method executes the DDL statement contained in the SQL property. If there is an error (for example, if you try to create the same table twice), an exception will be thrown. ExecSQL is specifically designed to execute SQL statements that do not return any results (ie. metadata manipulation statements). Executing statements that do return data, such as `select` statements, will not cause an error, however, there is no way to access the returned dataset. For `select` statements you should use the standard Delphi TQuery component.

ExecSQL explicitely commits each SQL statement that is executed.

see SQL for an example of ExecSQL.