Using InterBase/Firebird Connect

Anthony Boris
(anthonyvb@yandex.ru)

This document refers to
InterBase/Firebird Connect version 7.2
13th October 2003

Contents

Chapter 1

What is InterBase/Firebird Connect ?

InterBase/Firebird Connect available under the ParcPlace Public License [1] provides access to Borland InterBase [2] (version 6.x) and Firebird[3] (version 0.9.x and higher) databases.

InterBase is an open source relational database that runs on Linux, Windows, and a variety of Unix platforms. I hope InterBase/Firebird Connect will appear useful to VisualWorks community.

InterBase/Firebird Connect package includes:

- IBEXDI — EXDI layer support
- StORE for InterBase — supports the use of InterBase or Firebird as a repository for Store.
- Lens for InterBase — ObjectLens layer support.

1.1 Supported versions of InterBase and Firebird

This feature works with version 6.x of the InterBase and 0.9.x (or higher) of the Firebird on Windows (NT,2000) and Linux platforms.

IBEXDI, Lens & StORE has been tested using Firebird 1.02 and 1.5RC6 for Windows on Windows NT 4.0 (SP6) and Firebird 0.9.4 for Linux (SuperServer) on Red Hat Linux 6.1

1.2 Supported versions of VisualWorks

This feature has been tested with VW 7.2 (oct03.1).

1.3 Features at a glance

- two-phase commit coordination spanning multiple connections;
- multiple transactions per database connection (attach);
- full control of parameters of database connection (includes user validation,system management,etc.)
- flexible control of transactions(parameters, retain mode, emulation of auto-commit behavior) ;
- support BLOBs (text & binary);
- support of InterBase ARRAY field-type (currently limited to one-dimensional arrays).

Chapter 2

IBEXDI

Code snippets covered some of IBEXDI specific features:

- Two-phase commit coordination spanning multiple connections
- Multiply transaction per database connection (attach)
- Control of database connection parameteres
- Control of transactions
- Work with BLOBs
- Work with ARRAYs

2.1 Two-phase commit coordination spanning multiple connections

```
conn1 transactionCoordinatorFor: conn2.
conn1 connect.
conn2 connect.
conn1 begin.
sess1:=conn1 getSession prepare: 'select name from table1'.
sess2:=conn2 getSession prepare: 'insert into table2 (?)'.
ans:=sess1 execute ; answer.
[ans atEnd] whileFalse:
   [sess2 bindInput: sess1 next.
   sess2 execute ; answer].
conn1 commit.
conn1 disconnect.
conn2 disconnect.
```

2.2 Multiply transactions per database connection (attach)

```
   conn1 := InterBaseConnection new
       username: ’sysdba’;
       password: ’masterkey’;
       environment: ’d:\sampleBlob.gdb’;
       "also, you can define ’role’ and ’charset’ parameters "
       role: ’ADMIN’;
       charSet: ’WIN1252’.
   conn2 := c cloneConnection. "answer new connection with
    independent transaction scope and shared with conn1
    physical database connection"
   conn1 begin.
   sess1 := c prepare: ’select * from table1’.
   (sess1 execute ; answer) next; next.
   conn2 begin.
   conn1 rollback.
   conn1 disconnect. "disconnecting of conn1
    does not influence on conn2"
   sess2 := conn2 prepare: ’select * from table1’.
   (sess2 execute ; answer) upToEnd.
   conn2 commit.
   conn2 disconnect "now physical database connection
    will be detached"
```

2.3 Control of database connection parameteres

```
dp := IdentityDictionary new.
  dp at: #forceWrite put: 1; "0-disable, 1-enable"
     at: #numBuffers put: 500;
     at: #sweep put: nil. "no arguments for ’sweep’ "
  conn databaseParameters: dp.
  conn connect: ’password’.
  conn disconnect.
```

2.4 Control of transactions

```
 "array of parameters (see InterBase manuals for details)"
  conn transactionParameters:
    #(write concurrency noWait
      protected lockRead ’TABLE1’
      protected lockWrite ’TABLE2’).
  conn begin.
  sess := (conn getSession prepare: ’select * from table1’).
  sess execute; answer.
  conn commitRetain. "commit with cursor hold"
  conn disconnect
```

2.5 Work with BLOBs

```
  session prepare: ’INSERT INTO MYTABLE (id, myField)
   VALUES(?, ?)’.
  entry:=Array
     with: 1
     with: (ReadStream on: #(1 2 3 4 5)). "ByteArray must be
      wrapped into a ReadStream "
  session bindInput: entry;
     execute;
     answer.
```

2.6 Work with ARRAYs

```
session prepare: ’create table myArray
(id integer, myField integer[4])’;
  execute;
  answer.
connection begin.
session prepare: ’INSERT INTO MYARRAY (id, myField)
  VALUES(?, ?)’.
entry:=Array
   with: 99
   "wrap array with an instance of InterBaseArray"
   with: (InterBaseArray
             forArray: #(2 3 4 5)
             column: ’MYFIELD’
             table: ’MYARRAY’).
session bindInput: entry;
   execute;
   answer.
connection commit.
"now update array slice"
connection begin.
    session prepare:
     ’select id, myField from myArray for update’;
```

```
        cursorName: ’S’; "set name of cursor to ’S’ "
        execute.
    answer := session answer.
    session2 := connection getSession prepare:
     ’update myArray set myField=? where current of S’.
    [answer atEnd]
        whileFalse:
            ["get an instance of InterBaseArray"
            entry := answer next last.
            "new array slice"
            entry array: #(555 999) ;
"replace array slice from 2 to 3 with 555 & 999 "
                dimensions: #(#(2 3)).
            session2 bindInput: (Array with: entry).
            session2 execute; answer].
    connection commit.
   "fetch"
   session prepare: ’select * from myArray ’;
       execute.
   session answer upToEnd.
```

Chapter 3

Store for InterBase

`StoreForInterBase` allow use InterBase or Firebird as back-end for Store.

3.1 StoreForInterBase Installation

1. Installing InterBase(Firebird)

   Get InterBase or Firebird (my preference) for your platform (e.g. from `http://www.ibphoenix.com`) and install it.

2. Setting Up StORE

   Load parcel StoreForInterBase into clean image. After load will be open StORE For InterBase/Firebird ”wizard” window (later you can always do-it `InterBaseBroker createDatabase` to re-open), which will allow you to execute all operations on installation.

   Follow the instructions in the overview panel for:

   - creating new database;
   - adding new user accounts;
   - test of the connection with database;
   - running of the StORE installation process.

Creating database and the addition of users, may need to be performed by a database administrator. Below example (Linux)[1]:

```
cd /opt/interbase/bin
```
(your InterBase home directory)

```
isql
```

in interactive mode enter (don’t forget semicolons !):

[1]Certainly, you should replace ”masterkey” — SYSDBA password by default — with real password.

```
create database ’/opt/visualworks/mystore.gdb’
   user ’sysdba’ password ’masterkey’ page_size 4096;
exit;
```

Addition of users:

```
cd /opt/interbase
/bin/gsec -user sysdba -password masterkey
```

or, for create accounts on remote host:

```
    gsec -database host:/opt/interbase/isc4.gdb
```

(where `/opt/interbase/isc4.gdb` path to security database)

in interactive mode enter:

```
add newuser -pw password
quit
```

Chapter 4

Lens for InterBase/Firebird

The Lens provides high-level facilities that simplify the task of database access from VisualWorks. Used in concert with the Lens-Runtime and IBEXDI parcels, parcel `IBLens` provides the facilities to use the Lens on InterBase(6.x) and Firebird(0.9.x and higher) database servers.

Links

[1] ParcPlace Public License: `http://www.parcplace.com/support/opensource/PPL-1.0.html`

[2] Borland InterBase: `http://www.borland.com/interbase/`

[3] Firebird: `http://www.firebirdsql.org/`

[4] IBPhoenix site: `http://www.ibphoenix.com`