Short Firebird Migration Guide to Firebird 4.0

Table of Contents

Preface

This article is written for those who plan to migrate to Firebird 4.0 from Firebird 2.5 or Firebird 3.0. Why 2.5? Many people still use 2.5, but plan to migrate to 4.0 due to the native replication, so we will cover migration process 2.5→4.0, as well as 3.0→4.0.

The sponsor of "Short Firebird Migration Guide to 4.0" is IBSurgeon (https://www.ib-aid.com, https://www.ibase.ru): vendor of HQbird, advanced Firebird distribution, and technical support, optimization and recovery services provider.

1. Manual installation of Firebird 4.0 on Windows

We will describe how to install Firebird 4.0 from the archive, as the most fastest and flexible installation way. One of the advantages of the installation from the archive is ability to easily install several Firebird versions simultaneously on the same computer.

Download the archive with installation of Firebird (also called zipkit) from www.firebirdsql.org. Please note that there are archives for 32-bit and 64-bit versions of Firebird available, so choose the one you prefer.

Unpack the zipkit into the folder where you want to have Firebird 4.0 installed, for example, c:\Firebird\4.0.

Then, for simplicity, we recommend creating user SYSDBA. In 2.5 this user is mandatory, and exists in every installation with the default password *masterke*.

We recommend creating of SYSDBA in case of the migration from 2.5, to facilitate the migration process.

1.1. Create SYSDBA

Starting from version 3.0, user SYSDBA does not exist (in the default user manager plugin Srp), so it is necessary to explicitly create user and specify its password. It can be done using the standard interactive query tool *isql.exe* (recommended), or with security database management tool *gsec.exe*.

Note

Depending on the Firebird folder location, these tools can request to start themselves with "Run as Administrator".

1.1.1. Create SYSDBA with ISQL

Run interactive query tool *isql.exe* from root folder of Firebird. Make sure that Firebird 4.0 is not running neither as a service nor as an application. It is necessary to connect with security database in the embedded mode and specify user SYSDBA without a password (see the complete example below). Though user SYSDBA does not exist, in the embedded mode user and password are not checked, and Firebird trusts any specified user name. The following SQL query to create SYSDBA user:

```
CREATE USER SYSDBA PASSWORD '<password>';
```

After that, close isql.exe with command 'exit';

Example 1. Create SYSDBA with ISQL

```
c:\Firebird\4.0>isql

Use CONNECT or CREATE DATABASE to specify a database

SQL> connect security.db user SYSDBA; Database: security.db, User: SYSDBA
SQL> CREATE USER SYSDBA PASSWORD 'm8ku234pp';
SQL> exit;

c:\Firebird\4.0>
```

1.1.2. Create SYSDBA with GSEC

Run *gsec.exe* with user SYSDBA and security database `security.db`, and then run the following command:

```
add SYSDBA -pw <password>
```

Example 2. Create SYSDBA with GSEC

```
c:\Firebird\4.0>gsec -user SYSDBA -database security.db

* gsec is deprecated, will be removed soon *

GSEC> add SYSDBA -pw m8ku234pp
GSEC> quit

c:\Firebird\4.0>
```

Warning

Tool *gsec.exe* is deprecated, many security features available through SQL, are not implemented in gsec.

1.2. Configuration

Before you start Firebird as a service, it is necessary to choose the server architecture (also called server mode): SuperServer, SuperClassic or Classic.

1.2.1. Server architecture

By default, Firebird runs as in SuperServer mode (recommended). To change the architecture/mode, it is necessary to change parameter *ServerMode* в `firebird.conf`. If you have used Classic or SuperClassic in 2.5 or 3.0, and you have specific reasons for it, you can continue to use it.

Uncomment it (remove symbol #) and set one of the following: Super, SuperClassic или Classic.

```
ServerMode = Classic
```

1.2.2. Authorization with Firebird 2.5 client library (fbclient.dll)

Firebird 4.0 by default uses secure password authorization Srp. Firebird 2.5 (and earlier) client libraries use traditional, less secure legacy authentication (LegacyAuth), which is by default disabled in Firebird 4.0.

In order to enable LegacyAuth (and allow old firebird client libraries to connect to Firebird 4.0), it is necessary to change the following parameters in firebird.conf: *AuthServer*, *UserManager* и *WireCrypt*.

Example 3. Enable legacy authorization, to allow old Firebird client libraries to connect to Firebird 4.0

```
AuthServer = Srp256, Srp, Legacy_Auth
UserManager = Srp, Legacy_UserManager
WireCrypt = Enabled
```

Important!

After adding the parameters above, there will be 2 active user managers in Firebird (Srp and Legacy_UserManager in our example), and the first in the UserManager list will be used as a default (Srp in our example).

Please note, that users with the same names in the different user managers are different! They can have different password.

If you do not plan to use secure password authentication (Srp), and plan to use only the legacy authentication, you can disable Srp: remove from `AuthServer` plugins Srp256 and Srp; remove Srp from `UserManager`, and `WireCrypt` can be changed to Disabled.

In the example above, we have created user SYSDBA the user manager SRP. In Legacy_UserManager user SYSDBA already exists, with the standard password *masterkey*, which is better to be changed. Let's change the standard password for SYSDBA in Legacy_UserManager with isql.

For this, run `ALTER USER` with specified Legacy_UserManager:

Example 4. Change password for user SYSDBA in Legacy_UserManager

```
c:\Firebird\4.0>isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect security.db user sysdba;
Database: security.db, User: SYSDBA
SQL> ALTER USER SYSDBA SET PASSWORD 'er34gfde' USING PLUGIN Legacy_UserManager;
SQL> exit;

c:\Firebird\4.0>
```

1.2.3. Set connection default timezone

In Firebird 4.0 the new date-time data types with time zones support are introduced.

Even if you don't plan to use data types with time zone support, it should be borne in mind that expressions CURRENT_TIMESTAMP and CURRENT_TIME now return data types with time zones. To facilitate the transition of the legacy code, it is necessary to enable compatibility mode, which allows transparent conversion of types with time zones to the types without timezones. However, this conversion will work incorrectly if the connection default timezone is set incorrectly.

Usually the timezone of the connection is set on the client side. If timezone is not set on the client side, the timezone of the operating system will be used. Or, you can specify the default timezone with the firebird.conf parameter `DefaultTimeZone`:

```
DefaultTimeZone = Europe/Moscow
```

1.2.4. Run several instances of different Firebird versions

It is possible to run several instances of different Firebird versions, installed in the separate folders. In order to run several Firebirds, it is necessary to configure them to use different network ports (for the default configuration of providers, when server listens for TCP/IP). For this it is necessary to change `firebird.conf` parameter *RemoteServicePort*. If you already have Firebird running on the default port 3050, it is necessary to set another free port, for example, 3051.

```
RemoteServicePort = 3051
```

In order to connect from the client application to the Firebird instance with non-standard port, specify port number in the connection string, for example:

```
isql.exe -user SYSDBA -pass mypassword localhost/3051:c:\mydatabase\mydb.dfb
```

Also it is recommended to alter parameters *IpcName* и *RemotePipeName*. If they will remain the same, the second Firebird instance will log an error in firebird.log. It is not a critical error if you don't use XNET or WNET protocols.

However, if you wish to use XNET or WNET protocols to connect to the second instance, it will be necessary to change parameters in `firebird.conf`, and set them on the client side in DPB (database parameters block).

1.3. Run Firebird as a Windows service

instsvc.exe tool insert, change or remove information about Firebird service in the services database of the operating system. Also it allows to start and stop Firebird service.

If you run it without parameters, it will show help about all commands and parameters.

```
instsvc
Usage:
  instsvc i[nstall]
                  [ -a[uto]* | -d[emand] ]
                  [ -g[uardian] ]
                  [ -l[ogin] username [password] ]
                  [ -n[ame] instance ]
                  [ -i[nteractive] ]

          sta[rt]   [ -b[oostpriority] ]
                    [ -n[ame] instance ]
          sto[p]    [ -n[ame] instance ]
          q[uery]
          r[emove]  [ -n[ame] instance ]


  '*' denotes the default values
  '-z' can be used with any other option, prints version
  'username' refers by default to a local account on this machine.
  Use the format 'domain\username' or 'server\username' if appropriate.
```

Important

Instsvc tool must be run in the console with administrative privileges.

To install Firebird as a service enter command

```
instsvc install
```

In this case Firebird will be installed as a service named "Firebird Server – DefaultInstance". This service by default will automatically start on operating system boot, under LocalSystem account.

If you want to run several Firebird instances as a service, you must specify their names (different) using -n <name> option

```
instsvc install -n fb40
```

To run service use command

```
instsvc start
```

If Firebird service was installed with non-default name, you must use -n option

```
instsvc start -n fb40
```

To stop Firebird service run command

```
instsvc stop
```

If Firebird service was installed with non-default name, you must use -n option

```
instsvc stop -n fb40
```

To remove Firebird service use command

```
instsvc remove
```

If Firebird service was installed with non-default name, you must use -n option

```
instsvc remove -n fb40
```

You may query all names of Firebird installed services

```
instsvc query
```

```
Firebird Server - fb30 IS installed.
  Status  : running
  Path    : C:\Firebird\3.0\firebird.exe -s fb30
  Startup : automatic
  Run as  : LocalSystem

Firebird Server - fb40 IS installed.
  Status  : running
  Path    : C:\Firebird\4.0\firebird.exe -s fb40
  Startup : automatic
  Run as  : LocalSystem
```

1.3.1. Using install_service.bat and uninstall_service.bat

Firebird installation contain two bat-files that helps installing and removing Firebird service: `install_service.bat` and `uninstall_service.bat`.

Using this files installation will be much simplier

```
install_service.bat
```

To remove service run the following command

```
uninstall_service.bat
```

If you want to set up name for the Firebird service, specify it as an argument

```
install_service.bat fb40
```

The same is to remove specific Firebird service

```
uninstall_service.bat fb40
```

1.4. Installing client

If we speak about installing Firebird client only, you need to have `fbclient.dll` file. Firebird 4.0 client requires Microsoft Runtime C++ 2017 with the same bitness as fbclient.dll. If Microsoft Runtime is not installed, you may just copy it's two files, `msvcp140.dll` and `vcruntime140.dll` that are included in ZIP for Windows.

Also, it is welcome to have `firebird.msg` at the same place where `fbclient.dll` is installed or copied. There are lot of error messages included in `fbclient.dll`, but at least for Firebird console tools it is good to have`firebird.msg` nearby.

Unlike Firebird 2.5 and 3.0 Firebird 4.0 client library may need ICU files (`icudt63.dll`, `icuin63.dll`, `icuuc63.dll` и `icudt63l.dat`). Previous versions required these files only for the server, not client. Now these files may be needed by client library, especially if you want to work with datatypes like `TIMESTAMP WITH TIME ZONE` and `TIME WITH TIME ZONE`. ICU is also used by Firebird client for `UtilInterface::decodeTimeTz()` and `UtilInterface::decodeTimestampTz()` functions.

Note

Windows 10 already have appropriate ICU library.

If you require TCP/IP wire compression, you also need `zlib1.dll` library.

For connection encryption you may need file `plugins/chacha.dll` if you want to use ChaCha encryption plugin.

As usual, application is able to load `fbclient.dll` if it have same bitness as application and resides near the application or in `PATH` or system folders (`system32` for 64bit and `SysWOW64` for 32bit).

Important

Using `PATH` as a place of the client library may conflict with the other applications, that needs another version of the client library. So, if application must work independently of the other applications, client files must be placed at the application folder, and this folder must not be in `PATH`.

To install Firebird client library to the Windows system folder use the command

```
instclient install fbclient
```

Important

Instclient does not copy any other files than `fbclient.dll` to the system folder.

1.5. Installing embedded

Embedded now requires more files than previous versions. Minimum set of files and folders for Firebird 4.0 embedded is the following:

- intl
  - fbintl.conf
  - fbintl.dll
- plugins
  - engine13.dll
- firebird.conf
- icudt63l.dat
- fbclient.dll

- ib_util.dll
- icudt63.dll
- icuin63.dll
- icuuc63.dll
- msvcp140.dll
- vcruntime140.dll
- firebird.msg

Also you may copy tools `fbsvcmgr.exe`, `fbtracemgr.exe`, `gbak.exe`, `gfix.exe`, `gstat.exe`, `isql.exe`, `nbackup.exe` to that folder, if you need them.

Note

When migrating from Firebird 2.5 two moments should be considered:

- Several files is needed instead of one `fbembed.dll`, and you must not rename `fbclient.dll`. Use `fbclient.dll` name as a library name in your driver or components.
- If you want to connect to the one database from the different applications on the same computer, change `firebird.conf` file - set parameter `ServerMode` to `SuperClassic` to `Classic`. (as the default behavior of Firebird 2.5 embedded). SuperServer mode does not allow to connect to the database more than one application with the Firebird embedded.

2. Converting the database to a new format

Databases of the Firebird 4.0 now have On-Disk Structure (ODS) 13.0. You may convert your database to the Firebird 4.0 format using *gbak* - make backup first on the previous Firebird version, than restore on Firebird 4.0. But, before that you should check and fix any incompatibilities.

2.1. SQL incompatibilities list

Incompatibilities at the SQL level may happen with the database objects (PSQL procedures and functions) and with DSQL queries.

Here are some common problems at the SQL level that you may fix before moving to the new ODS. Complete list of incompatibilities you will find in Firebird 3.0 Release Notes (for those who moves from the Firebird 2.5) and Firebird 4.0 Release Notes 4.0, "Compatibility Issues" chapter.

2.1.1. New reserved words

Check your database for the new keywords that are used for the identifiers, column names and variables. With the dialect 1 these keywords cannot be used in SQL. In dialect 3 - may be used, but must be surrounded by double quotes. Please read Firebird 3.0 and 4.0 Release Notes, "Reserved Words and Changes" chapter. Keywords may be used as identifiers, but this is not recommended.

2.1.2. Column names in PSQL cursors

This item is relevant for those who moves from Firebird 2.5. All output columns of PSQL cursors declared as `DECLARE CURSOR` must have explicit name or alias. The same goes for PSQL cursors `FOR SELECT … AS CURSOR <cursor name> DO …`.

Example 5. The problem with unnamed columns in cursors

```
create procedure sp_test
returns (n int)
as
  declare c cursor for (select 1 /* as a */ from rdb$database);
begin
  open c;
  fetch c into n;
  close c;
  suspend;
end
```

```
Statement failed, SQLSTATE = 42000
unsuccessful metadata update
-ALTER PROCEDURE SP_TEST failed
-Dynamic SQL Error
-SQL error code = -104
-Invalid command
-no column name specified for column number 1 in derived table C
```

2.1.3. New data types

Firebird 4.0 introduces new data types:

- `TIMESTAMP WITH TIME ZONE`
- `TIME WITH TIME ZONE`
- `INT128`
- `NUMERIC(38, x)` и `DECIMAL(38, x)`
- `DECFLOAT(16)` и `DECFLOAT(34)`

  Last two types does not give problems, because you could not use them before, and they were not returned by expressions.

Some expressions now may return result as `NUMERIC(38, x)`, `DECIMAL(38, x)` or `INT128`. This problem will be described later, because it does not appear at the ODS change.

`CURRENT_TIMESTAMP` and `CURRENT_TIME` now returns `TIMESTAMP WITH TIME ZONE` и `TIME WITH TIME ZONE`. This may be serious problem. You may set compatibility mode for the old client libraries and applications, but this will not help for the code of stored procedures, functions and triggers. You should use `LOCALTIMESTAMP` and `LOCALTIME` instead of `CURRENT_TIMESTAMP` and `CURRENT_TIME` where you do not want to work with the timezone datatypes. These expressions were introduced in Firebird 2.5.9 and Firebird 3.0.4 to allow you prepare your databases for the migration to the Firebird 4.0.

2.1.4. Time and date literals

Firebird 4.0 now have stricter control of date and time literals syntax.

Literals 'NOW', 'TODAY', 'TOMORROW', 'YESTERDAY' with the implicit typecast (prefixed with TIMESTAMP, DATE, TIME) now rejected. Value of these literals were evaluated during prepare of DSQL or PSQL, and produced unexpected results.

If you have something like TIMESTAMP 'NOW' in DSQL or PSQL, there will be compatibility issue.

Example 6. The following code will not be compiled

```
..
DECLARE VARIABLE moment TIMESTAMP;
..
SELECT TIMESTAMP 'NOW' FROM RDB$DATABASE INTO :moment;

/* variable 'moment' here will be "frozen" as the time of the last compliation of
the procedure or function  */
..
```

You should clear out literals like that from your procedures, triggers and functions (for example, change to explicit `CAST('NOW' AS TIMESTAMP)`) before moving to the new ODS.

Also you should check other date and time literals with the explicit date/time. Previously, such literals allowed non-standard delimiters. Now these delimiters rejected. More details about correct date and time literals you may read in "Firebird 4.0 Language Reference" at the chapter "Date and time literals".

2.2. External functions (UDFs) now considered as deprecated

Support for the external function (UDF) feature is deprecated in Firebird 4. Its immediate effect is that UDFs cannot be used with the default configuration, where the parameter `UdfAccess` in `firebird.conf` is set to None and the UDF libraries ib_udf and fbudf are withdrawn from the distribution.

Most of the functions in those libraries were already deprecated in previous Firebird versions and replaced with built-in analogues. Safe replacements for a few of the remaining functions are now available, either in a new library of user-defined routines (UDRs) named [lib]udf_compat.[dll/so/dylib], or as scripted conversions to PSQL stored functions.

The Firebird 4 distribution contains a script to migrate all (or any) of those UDF declarations. You can edit and extract from it to suit, if you wish, but you must keep the respective re-declarations and conversions intact as scripted.

If you still want to use UDFs, you must change `firebird.conf` parameter

```
UdfAccess = Restrict UDF
```

You can find more information in Firebird 4.0 Release Notes.

2.3. Upgrade your database to the new ODS

After preparation you may try to uprgade your database using *gbak* tool.

This example assumes that there is one computer with Firebird 3 and Firebird 4. Firebird 3 is using TCP port 3053, and Firebird 4 - -3054.

First you should make backup copy of your database using current Firebird version

```
gbak -b -g -V -user <username> -pas <password> -se <service> <database> <backup_file>
-Y <log_file>
```

Example 7. Create backup using current Firebird version

```
gbak -b -g -V -user SYSDBA -pas 8kej712 -se server/3053:service_mgr my_db
d:\fb30_backup\my_db.fbk -Y d:\fb30_backup\backup.log
```

Next you need to restore this backup using Firebird 4

```
gbak -c -v -user <username> -pas <password> -se <service> <backup_file>
<database_file> -Y <log_file>
```

Example 8. Restore of the backup using Firebird 4.0

```
gbak -c -v -user SYSDBA -pas 8kej712 -se server/3054:service_mgr
d:\fb30_backup\my_db.fbk d:\fb40_data\my_db.fdb -Y d:\fb40_data\restore.log
```

Important

Note that -V and -Y options must be used so that you can see in the log file what went wrong during the restore process.

After restore carefully examine the `restore.log` for errors. Note that this log will not indicate incompatibilities at the SQL level, because metadata objects are note recompiled (altered) during restore. If any procedure, trigger or view contain incompatibilities, then only ALTER of this object Firebird will raise an error. To clean database from these errors you need to extract database script using "isql -x database >script.sql ..." with the previous Firebird version, and try to create empty database from this script using Firebird 4, fixing SQL errors one by one.

2.3.1. UDF Warnings

After restore in the `restore.log` you may see the following warnings

```
gbak: WARNING:function UDF_FRAC is not defined
gbak: WARNING: module name or entrypoint could not be found
```

It means you have a UDF that is declared in the database but whose library is missing — which, of course, we know is true. There is a description above what to do in this case. Mostly this is related to your custom UDFs. If you used only ib_udf and fbudf libraries, you may replace them to the built-in functions or their safe UDR counterparts from `udf_compat.dll`. This can be done by the script `misc/upgrade/v4.0/udf_replace.sql`. Use the following command

```
isql -user sysdba -pas masterkey -i udf_replace.sql {your-database}
```

Example 9. Warning

This script does not change anything for the third-party UDF libraries!

3. Migrating database aliases

This section is relevant for Firebird 2.5 users. The contents of `aliases.conf` now moved to `databases.conf` (since Firebird 3.0). Now you are able to setup not only aliases in the `databases.conf`, but lot of database specific parameters.

Database-level parameters now marked in `firebird.conf` as 'Per-database configurable'.

4. Migrating user list

User list migration is different for Firebird 2.5 and Firebird 3.0. Migrating user list from Firebird 3.0 is much simpler, here is an example.

4.1. Migrating user list from Firebird 3.0

The simpliest way to migrate Firebird 3 security database to Firebird 4 is to make backup of `security3.fdb` and to restore it with Firebird 4.0. But, in this case you will loose some new features. Let explain a bit complex way:

1. Make backup using Firebird 3.0

```
c:\Firebird\3.0>gbak -b -g -user SYSDBA security.db d:\fb30_backup\security.fbk
```

2. Restore this backup on Firebird 4.0 with the new name

```
c:\Firebird\4.0>gbak -c -user SYSDBA -pas 8kej712 -se localhost/3054:service_mgr
d:\fb30_backup\security.fbk d:\fb40_data\security_30.fdb
```

3. Save the following script to the file `copy_user.sql`

```
set term ^;

EXECUTE BLOCK
AS
  -- change constants to your drive, folder and file names
  DECLARE SRC_SEC_DB     VARCHAR(255) = 'd:\fb40_data\security_30.fdb';
  DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
  ---------------------------------------------------
  DECLARE PLG$USER_NAME  SEC$USER_NAME;
  DECLARE PLG$VERIFIER   VARCHAR(128) CHARACTER SET OCTETS;
  DECLARE PLG$SALT       VARCHAR(32) CHARACTER SET OCTETS;
  DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$FIRST      SEC$NAME_PART;
  DECLARE PLG$MIDDLE     SEC$NAME_PART;
  DECLARE PLG$LAST       SEC$NAME_PART;
  DECLARE PLG$ATTRIBUTES BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$ACTIVE     BOOLEAN;
  DECLARE PLG$GROUP_NAME SEC$USER_NAME;
  DECLARE PLG$UID        PLG$ID;
  DECLARE PLG$GID        PLG$ID;
  DECLARE PLG$PASSWD     PLG$PASSWD;
BEGIN
  -- transferring SRP plugin users
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$VERIFIER,
          PLG$SALT,
          PLG$COMMENT,
          PLG$FIRST,
          PLG$MIDDLE,
          PLG$LAST,
          PLG$ATTRIBUTES,
          PLG$ACTIVE
      FROM PLG$SRP
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
```

```
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
          INTO :PLG$USER_NAME,
               :PLG$VERIFIER,
               :PLG$SALT,
               :PLG$COMMENT,
               :PLG$FIRST,
               :PLG$MIDDLE,
               :PLG$LAST,
               :PLG$ATTRIBUTES,
               :PLG$ACTIVE
  DO
  BEGIN
    INSERT INTO PLG$SRP (
        PLG$USER_NAME,
        PLG$VERIFIER,
        PLG$SALT,
        PLG$COMMENT,
        PLG$FIRST,
        PLG$MIDDLE,
        PLG$LAST,
        PLG$ATTRIBUTES,
        PLG$ACTIVE)
    VALUES (
        :PLG$USER_NAME,
        :PLG$VERIFIER,
        :PLG$SALT,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST,
        :PLG$ATTRIBUTES,
        :PLG$ACTIVE);
  END
  -- transferring Legacy_UserManager plugin users
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$GROUP_NAME,
          PLG$UID,
          PLG$GID,
          PLG$PASSWD,
          PLG$COMMENT,
          PLG$FIRST_NAME,
          PLG$MIDDLE_NAME,
          PLG$LAST_NAME
      FROM PLG$USERS
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
```

```
            INTO :PLG$USER_NAME,
                 :PLG$GROUP_NAME,
                 :PLG$UID,
                 :PLG$GID,
                 :PLG$PASSWD,
                 :PLG$COMMENT,
                 :PLG$FIRST,
                 :PLG$MIDDLE,
                 :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
        :PLG$USER_NAME,
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^

set term ;^

commit;

exit;
```

Important

Do not forget to change `SRC_SEC_DB` variable.

Note

SYSDBA was excluded, because it is expected that you already initialized SYSDBA accout during Firebird 4 installation.

4. Execute this script at Firebird 4.0, connecting to the security.db in embedded mode

```
c:\Firebird\4.0>isql -i "d:\fb40_data\copy_users.sql" -u SYSDBA -ch UTF8
security.db
```

Congratulations! Your user list is transferred with all attributes and passwords.

4.2. Migrating user list from Firebird 2.5

This task is more complex. Firebird 3.0 introduced new authentication protocol - SRP (Secure Remote Password Protocol), which is used by default. Old (legacy) authentication method is available, but it is turned off by default, because considered not safe enough. Firebird 3.0 Release Notes describes transferring Legacy_UserManager users to SRP, but in this case you do not have ability to connect with fbclient version 2.5. Also, transferring of passwords from Legacy_UserManager to SRP is impossible. Example script will transfer user names, but will generate random passwords. If you want to restore current passwords, you should do it manually. Here is alternate script that allows to transfer legacy users from `security2.fdb` to `security3.fdb` using Legacy_UserManager plugin.

4.2.1. Copying user list to SRP plugin

New authentication model (Firebird 3) does not allow to convert Firebird 2.5 security database (security2.fdb) to Firebird 4 security database. But there exists upgrade method, that allows to save user account - user name, etc, except passwords from 2.5.

This method needs to run `security_database.sql` script, that is included to `misc/upgrade` of your Firebird 3 installation. This description assumes that you have copy of this script at the same folder with isql.

Note

- Firebird 4.0 does not have `security_database.sql` in its distribution, so you need to download Firebird 3.0 zip archive.
- Do not forget to change *masterkey* password in the script to your real SYSDBA password.

1. Make backup of `security2.fdb` using Firebird 2.5

```
c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
c:\Firebird\2.5\security2.fdb d:
\fb25_backup\security2.fbk
```

2. Restore this backup on Firebird 4.0

```
c:\Firebird\4.0>gbak -c -user SYSDBA -password masterkey -se
localhost/3054:service_mgr d:\fbdata\4.0\security2.fbk d:\f
bdata\4.0\security2db.fdb -v
```

3. Open the folder of isql Firebird 4.0, and run script:

   ```
   isql -user sysdba -pas masterkey -i security_database.sql
   {host/path}security2db.fdb
   ```

   `security2db.fdb` - this is an example name, you can use any name you choose.

4. Script generates new random passwords and show them on the screen. You need to copy this output and tell users about their new passwords.

4.2.2. Transferring user list to Legacy_UserManager plugin

Unlike the previous option this script will save user passwords. But it is recommended to move these users to Srp plugin.

1. Make backup of `security2.fdb` using Firebird 2.5

   ```
   c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
   c:\Firebird\2.5\security2.fdb d:
   \fb25_backup\security2.fbk
   ```

2. Restore this backup with Firebird 4.0

   ```
   c:\Firebird\4.0>gbak -c -user SYSDBA -password masterkey -se
   localhost/3054:service_mgr d:\fbdata\4.0\security2.fbk d:\f
   bdata\4.0\security2db.fdb -v
   ```

3. Save the following script to the file `copy_security2.sql`

   ```
   set term ^;

   EXECUTE BLOCK
   AS
     -- change these variables to your parameters
     DECLARE SRC_SEC_DB     VARCHAR(255) = 'd:\fbdata\4.0\security2.fdb';
     DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
     ---------------------------------------------------
     DECLARE PLG$USER_NAME  SEC$USER_NAME;
     DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
     DECLARE PLG$FIRST      SEC$NAME_PART;
     DECLARE PLG$MIDDLE     SEC$NAME_PART;
     DECLARE PLG$LAST       SEC$NAME_PART;
     DECLARE PLG$GROUP_NAME SEC$USER_NAME;
     DECLARE PLG$UID        INT;
     DECLARE PLG$GID        INT;
     DECLARE PLG$PASSWD     VARBINARY(64);
   BEGIN
   ```

```
  FOR EXECUTE STATEMENT q'!
      SELECT
          RDB$USER_NAME,
          RDB$GROUP_NAME,
          RDB$UID,
          RDB$GID,
          RDB$PASSWD,
          RDB$COMMENT,
          RDB$FIRST_NAME,
          RDB$MIDDLE_NAME,
          RDB$LAST_NAME
      FROM RDB$USERS
      WHERE RDB$USER_NAME <> 'SYSDBA'
!'
      ON EXTERNAL :SRC_SEC_DB
      AS USER :SRC_SEC_USER
      INTO
          :PLG$USER_NAME,
          :PLG$GROUP_NAME,
          :PLG$UID,
          :PLG$GID,
          :PLG$PASSWD,
          :PLG$COMMENT,
          :PLG$FIRST,
          :PLG$MIDDLE,
          :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
        :PLG$USER_NAME,
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^
```

```
set term ;^

commit;

exit;
```

Important

Do not forget to change value of the variable `SRC_SEC_DB` to your security database.

Note

This script excludes SYSDBA, because this user was initialized during Firebird 4 install. By default installation creates SYSDBA using SRP plugin, so, there will be no SYSDBA for Legacy_UserManager plugin. But, you may create SYSDBA using Legacy_UserManager, and in this case there will be two SYSDBA - for Srp and for Legacy_UserManager.

4. Run script on Firebird 4.0 connecting to the security database in embedded mode

```
c:\Firebird\4.0>isql -i "d:\fb40_data\copy_security2.sql" -u SYSDBA -ch UTF8
security.db
```

At that point all users transferred with all attributes and passwords.

5. Configuring Trusted Authentication

Configuring Trusted Authentication (if it is needed) in Firebird 4.0 is done the same way as in Firebird 3.0. For those migrating from Firebird 2.5, we will describe this process in more detail.

1. First you must turn on Win_Sspi plugin in the `firebird.conf` or `databases.conf` (by default it is turned off). This plugin will be used together with Srp.

```
AuthServer = Srp256, Win_Sspi
```

2. Next you must turn user mapping from Win_Sspi to CURRENT_USER.

```
CREATE MAPPING TRUSTED_AUTH
USING PLUGIN WIN_SSPI
FROM ANY USER
TO USER;
```

This query creates mapping only for the particular database. This mapping will not be applied to other databases. If you want to create mapping for all databases at the server, add GLOBAL keyword (check "Firebird 4 Language Reference" for the full syntax of CREATE MAPPING).

```
CREATE GLOBAL MAPPING TRUSTED_AUTH
USING PLUGIN WIN_SSPI
FROM ANY USER
TO USER;
```

3. Turn on SYSDBA-like access for Windows administrators (if needed).

   ```
   CREATE MAPPING WIN_ADMINS
   USING PLUGIN WIN_SSPI
   FROM Predefined_Group
   DOMAIN_ANY_RID_ADMINS
   TO ROLE RDB$ADMIN;
   ```

   Instead of SYSDBA-like access you may give administrative privileges to the particular user.

   ```
   create global mapping cto_sysdba
   using plugin win_sspi
   from user "STATION9\DEVELOPER"
   to user SYSDBA;
   ```

6. Application level incompatibilities

Firebird 4 client library "fbclient" is compatible with previous versions at API level. But there may be compatibility problems for some SQL queries. Some of them were described above. Next is information about other incompatibilities.

6.1. New data types

Some expressions can return new data types that your application cannot process without its modification. This modification may take too much time, or will require changing the code of the data access components, etc. To simplify migration to new versions, you can set the *DataTypeCompatibility* parameter to the compatibility mode with the required version in `firebird.conf` or `databases.conf`.

```
DataTypeCompatibility = 3.0
```

This is the fastest way to get compatibility with new data types. During time you may add new data types support to your applications. Since it will happen gradually, one datatype first, then another, etc, you may set data type binding of the types that your applications still do not support.

Syntax

```
SET BIND OF { <type-from> | TIME ZONE } TO { <type-to> | LEGACY | NATIVE | EXTENDED }
```

Please read detailed description of this command in "Firebird 4.0 Release Notes" and "Firebird 4.0 language reference". Using 'SET BIND OF' you can bind new types right after connecting to the database, and even create AFTER CONNECT trigger with the set of such commands.

For example, you added timezone support to your application, but still do not support INT128 and DECFLOAT. In this case you may create trigger.

```
create or alter trigger tr_ac_set_bind
on connect
as
begin
  set bind of int128 to legacy;
  set bind of decfloat to legacy;
end
```

6.2. Consistent read in READ COMMITTED transactions

Firebird 4 not only introduces Read Consistency for Statements in Read-Committed Transactions, but also makes it a default mode for all READ COMMITTED transactions, regardless of their RECORD VERSION or NO RECORD VERSION properties. This is done to provide users with better behaviour — both compliant with the SQL specification and less conflict-prone. However, this new behaviour may also have unexpected side effects. The most important is so-called "restarts" on update conflicts.

This can lead to the fact that some code not subject to transactional control can be executed multiple times within PSQL. Examples of this type of code may be:

- using external tables, sequences and context variables;
- sending email messages using UDF;
- using autonomous transactions or external queries.

Please read about 'Read Consistency' in the "Firebird 4.0 Release Notes".

Another important effect is that active cursors in READ COMMITTED READ CONSISTENCY transactions prevent garbage collection even in Read Only mode. It is recommended that you stop using single long-running transaction READ COMMITTED READ ONLY, and change it to several same transactions, each of them active for exactly as long as necessary.

If READ CONSISTENCY mode is undesirable for any reason, you may set configuration parameter *ReadConsistency* to previous behavior.

6.3. INSERT … RETURNING requires SELECT privilege

If some INSERT statement contains a RETURNING clause that refers columns of the underlying table, the appropriate SELECT privilege must be granted to the caller.

7. Conclusion

That’s it. We hope that this document will help to upgrade your databases and applications to Firebird 4.0 and get all benefits of the new version!