MWA

MWA Software

IBX for Lazarus and Firebird Security

Issue 1.0,
2 February 2018

McCallum Whyman Associates Ltd

EMail: info@ mccallumwhyman.com, http://www.mccallumwhyman.com

CONTENTS **Page**

1 Introduction

This document is a supplement to the IBX For Lazarus User Guide and explores how *IBX for Lazarus* can be used to create applications that manage Firebird users and their access rights to Firebird Databases.

Previously to Firebird 3, there was a single (global) security database (containing user credentials) per server, and the Services API was used to perform User Management. Many access rights were implicit (e.g. creating tables or even whole databases) and the granting and revoking of access rights was performed using DDL statements.

Firebird 3 has improved upon this by:

- Permitting User Management to be performed using a combination of virtual tables and DDL Statements.
- Supporting alternative security databases on a per database basis.
- Allowing all access rights to be explicitly managed through DDL statements.

For legacy support, the Services API is still available for User Management. It is limited to the global security database.

Through the IBServices unit, IBX has made available access to the Firebird Services API and hence could support User Management applications, with the supporting DDL Statements being executed using TIBSQL. This functionality continues to be available.

In IBX 2.1, the TIBUpdate component was introduced. This is intended to support dataset update using DDL statements. That is datasets generated from Firebird virtual tables and presented to the user using TIBQuery. Together they enable Firebird 3 style User Management through virtual tables and supporting DDL statements in a straightforward manner.

The *IBX for Lazarus* source code provides examples for both legacy and Firebird 3 User Management. This guide supports these examples and attempts to:

- explain the Firebird Security Model
- How IBX is used to support legacy User Management through the Services API.
- How IBX is used to support Firebird 3 User Management and the assignment of extended Access Rights

1.1 References

1. IBX for Lazarus User Guide -MWA Software – Issue 1.4 https://mwasoftware.co.uk/downloads/send/5-ibx-current/147-ibx4lazarusguide
2. Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html
3. Firebird 2.0.7 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes207.html
4. Firebird 2.1.7 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes217.html
5. Firebird 2.5.8 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/2_5/rlsnotes25.html
6. Firebird 2.5 Language Reference Update https://www.firebirdsql.org/refdocs/langrefupd25.html
7. US Department Of Defense Trusted Computer System Evaluation Criteria- DoD 5200.28-STD – December 1985
8.

2 The Firebird Security Model

Before discussing how IBX for Lazarus can be used to manage Firebird users, it is useful to have an appreciation of the Firebird Security Model. The aim of this section is to present an understanding of the Firebird Security Model and with reference to how IBX can be used to manage Firebird security. It collects together information from many sources including Release Notes for different versions of Firebird.

Note: throughout this section, endnotes are used to clarify various statements and to provide source references, in addition to direct references to documents. Direct references are in bold and enclosed in square brackets, while superscripts are used for endnotes.

2.1 Overview

From Firebird 3 onwards, Firebird supports database encryption and “over the write encryption”. Firebird 3 and earlier versions also provide Discretionary Access Controls.

Firebird implements a form of Discretionary Access Control (DAC) which has been “defined by the Trusted Computer System Evaluation Criteria **[7]** as:

> A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control).

These criteria require both “and enforcement mechanism (e.g., self/group/public controls, access control lists)” and requires users both to identify themselves and to “use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user.”

In Firebird:

- The objects of the DAC are items such as a Database Table and its data, and Stored Procedures.
- The subjects of the DAC include Firebird "users" as well as "objects" taking on the role of a subject (e.g. Stored Procedures and Triggers).
- The groups (or access control lists) correspond to SQL Roles. There is also special group called 'PUBLIC' that all users implicitly belong to.
- The access rights themselves include the rights to access and change database data and/or metadata.
- All Firebird objects have an "owner" - a Firebird user – and which is able to grant subject access rights on the object. However, the permission to grant rights can also be passed on to other users.

Each Firebird user is identified through a unique login "UserName" (limited to 31 characters and constrained to the same character set as an SQL Identifier) and authenticated through a simple password based authentication mechanism when the user connects to a database. This form of authentication is often known as "weak authentication"[1] - the password being no more than a plain text shared secret (originally limited to 8 characters[2]) and potentially vulnerable to brute force attacks.[3]

Firebird does not currently support any form of strong authentication[4] (e.g. based on some sort of authenticated cryptographic exchange and/or a two factor authentication method). Although the "plugin" architecture of Firebird 3 does make it possible to extend the database engine to better authentication schemes even when they are not part of the standard product[5].

Firebird also includes to notion of a "SuperUser". That is a user that has full access rights to all objects. Firebird calls this user "SYSDBA".

Access Rights on objects are granted to subjects/groups using DDL Grant and Revoke statements on a per database basis. For example:

```
 Grant Select on MyTable to Alice;
```

The above example grants the "Select" access right to a user called "Alice" to a table called "MyTable". The select access right allows Alice read access to the data stored in MyTable.

Note: DDL statements cannot have placeholder parameters and while they can be executed by TIBSQL, the SQL statement has to be formatted and assigned to the component each time it is used.

Prior to Firebird 3, access rights to metadata were implicit and permissive[6]. Any Firebird user could create an object and become the object's owner. In Firebird 3, DAC has been extended to the metadata, with create/alter/drop access rights being available on metadata objects. There is also an explicit "create database" permission that a user must be given in order to be allowed to create a new database.

Firebird 2.5 had also previously introduced a built-in role "RDB$ADMIN". A user granted this role at the database level has SuperUser rights to all objects in that database. A user may also be granted this role at the server level and hence have the privilege to manage user accounts.[7]

2.2 The Firebird Security Database

Prior to Firebird 3, each Firebird Server maintained its own security database. The security database is a normal Firebird Database but one that is used to keep track of Firebird users and their login credentials. It typically contained a single table called USERS. The Firebird Server reads the security database whenever it needs to authenticate a user login. Maintaining the integrity and security of the data held in the security database is thus critical to managing a secure Firebird Server.[8]

Prior to Firebird 2.0, the security database was directly maintained by the *gsec* utility. It was thus potentially available for read access by various users and hence exposed user credentials which, in turn, became vulnerable to a brute force attack on the password hashes stored in the database. Firebird 2.0 changed *gsec* to use the Firebird Services API and hence permitted the security database to be hidden from user view, accessible only to the Firebird Server.[9]

In Firebird 3, alternative security databases can be defined on a per database level.[10] That is the original Firebird Security Database is the default security database unless an alternative is defined for a given database. Furthermore, the Firebird 3 plugin architecture permits an alternative authentication algorithm to be used with its own security database schema.

Firebird 3 has also restored the ability of users to establish a direct embedded connection to a security database, if only to create alternative security databases. This restored capability needs to be managed with care if Firebird's previous vulnerabilities are not to be re-introduced. However, once a security database has been created (see 2.2.1.2), it can be hidden from view and only needs to be accessible to the Firebird Server.

Note: a Firebird security database is a Firebird database. It contain additional tables to those used by the default User Manager and can hence be shared by alternative User Managers each with their own tables.

2.2.1 Creating a Firebird Security Database

Prior to Firebird 3, there was only a single Firebird Security Database and this was normally provided as part of the initial set of files. The location differed between platforms and OS distributions but the installed file was the same and included a single initial (super) user “SYSDBA” and a default password “masterkey”. It was up to the Database Administrator (DBA) to change the SYSDBA password to something less well known and the create other users.

2.2.1.1 The Firebird 3 Default Security Database

With Firebird 3, different installers may take different strategies. Some may continue to provide a prepared version of the default security database with the SYSDBA user and a default password. However, some may do no more than install an empty security database and leave it to the DBA to create the SYSDBA user with an appropriate password. In many ways, this is better than installing the security database with a very well known password, and the “CREATE USER” DDL statement (introduced in Firebird 3) makes it straightforward for the DBA to create the SYSDBA user even when the security database is empty. The process is described in the Firebird 3 release notes **[2]**:

> Initialization is performed in embedded mode using the *isql* utility. For an embedded connection, an authentication password is not required and will be ignored if you provide one. An embedded connection will work fine with no login credentials and “log you in” using your host credentials if you omit a user name. However, even though the user name is not subject to authentication, creating or modifying anything in the existing security database requires that the user be SYSDBA; otherwise, *isql* will throw a privilege error for the CREATE USER request.

The SQL user management commands will work with any open database. Because the sample database `employee.fdb` is present in your installation and already aliased in `databases.conf`, it is convenient to use it for the user management task.

1. **Stop the Firebird server.** Firebird 3 caches connections to the security database aggressively. The presence of server connections may prevent *isql* from establishing an embedded connection.

2. In a suitable shell, start an *isql* interactive session, opening the employee database via its alias:

   ```
   > isql -user sysdba employee
   ```

3. Create the SYSDBA user:

   ```
   SQL> create user SYSDBA password 'SomethingCryptic';
   SQL> commit;
   SQL> quit;
   ```

4. To complete the initialization, start the Firebird server again. Now you will be able to perform a network login to databases, including the security database, using the password you assigned to SYSDBA.

If the SYSDBA already exists then “ALTER USER SYSDBA...” can be used instead to change the SYSDBA password. From Firebird 3.0.1 onwards the statement `CREATE OR ALTER USER SYSDBA PASSWORD <password>` can be used to initialize an empty security database. **[2]**

After it has been created, File System permissions must be used to protect the security database from unwanted attention:

- On a POSIX system, the Firebird Server usually operates under the user “firebird” and the security database is read/write to the “firebird” user only with no other users permitted read or write access. In the above, *isql* needs to be run as user *root* in order to access the security database. After initialising the SYSDBA user, ensure that security database is owned by user “firebird” and no other user can read it. A typical installation will show the user and group as “firebird” (both read/write) with no “world access”. The directory path should also no be writeable by any user other than root or firebird, otherwise a malicious user could replace the security database with a compromised version.

- On Windows, similar remarks apply. isql should be run by a System Administrator and the Firebird security database only readable/writeable by the Firebird Server or a System Administrator.

2.2.1.2 Firebird 3 Alternative Security Databases

Firebird 3 can also assign an alternative security database on a per database basis. There can be as many alternative security databases as needed and they may be shared between databases. Each security database lists a set of users and their passwords and each may have a different SuperUser password. However, it is also possible to map users from one security database into another. It is even possible to include the security database tables within a user database.

Alternative security databases are defined in the server's *database.conf* configuration file[11]. This replaced the *aliases.conf* file used prior to Firebird 3 and shares a similar syntax. In order to use an alternative security database, you must set up an alias for the database in the configuration file and then associate the alternative security database with the database alias. For example:

```
employee = $(dir_sampleDb)/employee.fdb
{
```

```
        SecurityDatabase = /var/lib/firebird/3.0/system/private.security.fdb
 }
```

assigns an alternative security database to the example employee database. The example is for a Debian derived system and locates the private security database in the same directory as the default security database[12].

Note: there is a “chicken and egg” problem with the creation of an alternative security database. In the above example, if the employee database did not exist when the entry was created then it can only be created if the alternative security database already exists and contains a SYSDBA user. However, if this is the only database served by the security database then it is not possible to create the SYSDBA user (if one does not exist) unless the database exists and can be connected to by isql in embedded mode.

An alternative security database is created using isql in embedded mode e.g. on a Debian derived system:

```
isql-fb -user SYSDBA
SQL>create database '/var/lib/firebird/3.0/system/private.security.fdb';
```

The above simply creates a regular Firebird Database. It is empty with no tables defined as yet. There is nothing to indicate its purpose;

Note: the location in which the database is created should be protected and not writeable by another other than a System Administrator or the Firebird Server. isql will need to be run by a System Administrator (e.g. root on a Posix system) and it may be necessary, after it has been created, to change its ownership and file system access permissions so that it is exclusively available to the Firebird Server.

However, while the above creates the database, there is no connection to it. In order to create any user, including the SYSDBA user, you have to connect to a database which uses the newly created security database. This means that the using database must exist before its security database can be initialised. For example:

```
SQL>connect employee;
SQL>create user SYSDBA password 'some-obscure-password';
SQL>commit;
SQL>quit;
```

Note: in order to create the SYSDBA, you have to connect to the database in embedded mode. Once the SYSDBA user has been created (it is automatically the superuser), you can now connect remotely as SYSDBA and create the other users. Creating the SYSDBA user also creates the tables needed to support the chosen user authentication plugin.

2.2.1.3 Alternative Security Databases and the Services API

In Firebird 3 the Services API can only be used to manage users in the default security database. However, it is still necessary for there to be access from the Services API to databases that use an alternative security database. For example, when using the Services API for backup/restore.

In order to direct the Services API login to the appropriate security database, the Services Parameter Block (SPB) has a new item code: isc_spb_expected_db[13]. The value of this item is the name of a database which uses the alternative security database. The login will then be authenticated using the security database associated with the “expected db”.

Note: From IBX 2.2 onwards, the Services API components support a new login parameter (expected_db) which corresponds to the SPB isc_spb_expected_db item code.

2.3 User Management

2.3.1 Firebird 2.1 and Earlier

Prior to Firebird 2.5, only the *gsec* utility and the Services API could be used to manage users. Either could be used to:

- List all users and their attributes
- Create a new user
- Modify an existing user's password or attributes
- Delete an existing user.

In this case, user attributes are limited: UID, GID, First Name, Middle Name and Last Name and are for information only. Each user is identified by a unique “User Name” and which is the key to the user's entry in the security database. A User Name once created cannot be modified.

Prior to Firebird 3, only the SYSDBA could use the Services API to manage users. However, Firebird 3 also permits users with admin rights to the default security database to perform User Management by logging in to the Services API with the role name “RDB$ADMIN”.[14]

2.3.2 Firebird 2.5

In addition to use of *gsec* and the Services API for User Management, Firebird 2.5 also introduced the CREATE/ALTER/DROP USER DDL statements **[6]**. These enable User Management from a Database Connection. The user must be logged in as SYSDBA or with the role RDB$ADMIN (see). However, even a normal user can use ALTER USER to change their own password.

2.3.2.1 CREATE USER

Description: Creates a Firebird user account.

Syntax:

```
CREATE USER username PASSWORD 'password'
   [FIRSTNAME 'firstname']
   [MIDDLENAME 'middlename']
   [LASTNAME 'lastname']
   [GRANT ADMIN ROLE]
```

GRANT ADMIN ROLE gives the new user the RDB$ADMIN role (see) in the security database. This allows them to manage user accounts, but doesn't give them any special privileges in regular databases.

2.3.2.2 ALTER USER

Description: Alters details of a Firebird user account. This is the only account management statement that can also be used by non-privileged users, in order to change their own account details.

Syntax:

```
ALTER USER username
   [PASSWORD 'password']
   [FIRSTNAME 'firstname']
   [MIDDLENAME 'middlename']
   [LASTNAME 'lastname']
   [{GRANT|REVOKE} ADMIN ROLE]

-- At least one of the optional parameters must be present.
-- GRANT/REVOKE ADMIN ROLE is reserved to privileged users.
```

2.3.2.3 DROP USER

Description: Removes a Firebird user account.

Syntax:

```
DROP USER username
```

2.3.3 Firebird 3

The weakness in Firebird 2.5, from the point of view of User Management from a database connection is that there was no way to list the existing users. The Services API has to be used for this purpose. With the introduction of alternative security databases and the restriction of the Services API to the default security database, a new mechanism was needed to list the users.

The SEC$USERS virtual table was introduced in Firebird 3[15] and lists the users and their attributes for users in the security database used to login to the database. If the user is logged in as SYSDBA or with the RDB$ADMIN role then all users are listed. Otherwise, the list is restricted to the current user.

2.3.3.1 DDL Extensions

Firebird 3 has also extended[16] the CREATE/ALTER/DROP USER syntax introduced in Firebird 2.5:

```
CREATE USER username [ options_list ] [ TAGS ( tag [, tag [, tag ...]] ) ]
ALTER USER username [ SET ] [ options_list ] [ TAGS ( tag [, tag [, tag ...]]
) ]
ALTER CURRENT USER [ SET ] [ options_list ] [ TAGS ( tag [, tag [,
tag ...]] ) ]
CREATE OR ALTER USER username [ SET ] [ options ] [ TAGS ( tag [, tag [,
tag ...]] ) ]
DROP USER username [ USING PLUGIN plugin_name ]
```

OPTIONS is a (possibly empty) list with the following options:

```
  PASSWORD 'password'
  FIRSTNAME 'string value'
  MIDDLENAME 'string value'
  LASTNAME 'string value'
  ACTIVE
  INACTIVE
  USING PLUGIN plugin_name
```

Each TAG may have one of two forms:

```
  TAGNAME = 'string value'
```

or the DROP *TAGNAME* tag form to remove a user-defined attribute entirely:

```
  DROP TAGNAME
```

The Active/Inactive option allows a user to be disabled rather than deleted. The "USING PLUGIN" clause allows multiple authentication plugins to be used for the same security database.

2.3.3.2 User Tags

User tags are additional attributes in "key=value" format that are available for use for unspecified purposes. They are managed using the CREATE/ALTER USER DDL statements and those specified for a given user can be read using the SEC$USER_ATTRIBUTES virtual table. The legacy user attributes UID and GID and copied to a Firebird 3 security database as tags when a Firebird 2 security database is upgraded[17].

2.3.3.3 Mapping Users between Security Databases

Firebird 3 introduces the concept of "mapping rules". These are used to map users (or indeed any valid access control "subject" identified by an authentication plugin) authenticated in one security database into a user or role in a database that is configured to use a different security database and/or authentication plugin.

For example, mapping rules allow a user (e.g. SYSDBA) authenticated in (e.g. the default security database) to be mapped to the SYSDBA user in a database using an alternative security database. Such a capability map be used to simplify overall database administration with a unified SYSDBA login.

Mapping rules are created by DDL statements executed in the database to which they apply. They are defined in **[2]** as:

```
{CREATE | ALTER | CREATE OR ALTER} [GLOBAL] MAPPING name
  USING {
    PLUGIN name [IN database] | ANY PLUGIN [IN database | SERVERWIDE] |
      MAPPING [IN database] | '*' [IN database]}
  FROM {ANY type | type name}
  TO {USER | ROLE} [name]
  --
DROP [GLOBAL] MAPPING name
```

- A Global Mapping applies to all databases that use the current security database and hence is recorded in the security database. A local mapping applies only to the current database.

- The Using clause is used to identify the source of the mapping defined as an authentication method (plugin) and the security database in which the source of the mapping is defined.

  Note: 'security.db' is conventionally defined in the Firebird databases.conf file to identify the default security database.

- The From clause identifies the type and name of the subject that is being mapped into the local database. The "type" is a subject defined by the plugin (with SRP "user" is a type name)

- The To clause identifies what is being mapped to. Either a user or a role.

2.3.4 Windows Integration

2.4 Access Rights

2.5 Roles

2.5.1 Creating a Role

2.5.2 Users and Roles

2.5.3 The RDB$ADMIN role

2.6 Access Rights and Database Schemas

2.7 Database Encryption

2.8 “Over the Wire” Encryption

2.9 Embedded Databases

3 Legacy User Management and IBX

4 Firebird 3 User Management and IBX

1 A useful definition of Weak Authentication is provided by Jari Arkko, Pekka Nikander as “cryptographic authentication between previously unknown parties without relying on trusted third parties.” (https://www.ietf.org/proceedings/57/slides/enroll-4/enroll-4.ppt)

2 The eight character password limit was still present in Firebird 2.5. However, in Firebird 3, the SRP authentication plugin has an effective password length of 20 bytes – although as International Character Sets are also supported by Firebird 3 (e.g. UTF8), the actual effective password size is not easy to quantiify. See “Plugins Q&A” Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html

3 Firebird's vulnerability to brute force attacks has been significantly reduced over successive releases. For a discussion of this subject see the “Details of the Security Changes in Firebird 2” in the Firebird 2.1.7 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes217.html

4 Firebird 3 does implement an authentication method based upon an unauthenticated Diffie-Hellman exchange: the Secure Remote Password (SRP) protocol – and which is a major improvement on earlier versions. However, SRP simply protects the password exchange and it neither relies on a trusted third party or some form of two factor authentication, it cannot be consider as “strong authentication”. See “New Authentication Method in Firebird 3” Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html

5 See section on “Plugins”. Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html

6 Prior to Firebird 3, the System Tables used to hold a database's metadata were both publicly readable and writeable by a System Administrator. In Firebird 3, they become read only, with explicit permissions controlling which users can update the metadata and which metadata objects they can update (through DDL statements). (e.g. making “Create Table” and “Create Procedure” separate assignable privileges”. See “User Privileges for Metadata Changes” Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html.

7 The Firebird 2.5 release notes describe the RDB$ADMIN role as something that can be granted to a user on a per database basis and/or to the security database. At the database level, it is granted the same as any other role using a GRANT DDL statement (e.g. GRANT RDB$ADMIN TO ALICE). However, an entirely different syntax is used for the security database (e.g. ALTER USER ALICE GRANT ADMIN ROLE”). See RDB$ADMIN Role in Firebird 2.5.8 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/2_5/rlsnotes25.html

8 It should be recalled that a major DAC requirement is that “The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user”. In Firebird, it is very easy to overlook the importance of protecting the security database and that file system protections have to be relied upon to do this. This includes both limiting read and write access to authorised users only and ensuring that the database's file system path is protected from unauthorised modification. Otherwise a malicious user could replace the security database with a compromised version. All directories in the security database's file system path must restrict write access to authorised users only.

9 This subject is discussed at length in the release notes. See section “Details of the Security Changes in Firebird 2.0” in Firebird 2.0.7 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes207.html.

10 A user database can even incorporate its own security database – although in doing this you do need to be confident that the password hashes are not vulnerable to brute force attacks. See “Location of User Lists” in Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html

11 See section on “Creating an Alternative Security Database” Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html.

12 For a list of all parameters available for use in *databases.conf* and the file's syntax see the section “Per-database Configuration” in Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html.

13 This is described in the file “doc/README.services_extension” provided with the Firebird 3 source code.

14 This is also described in the file “doc/README.services_extension” provided with the Firebird 3 source code.

15 See section “New System Tables” in Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html.

16 See section “Security->SQL Features for Managing Access” in Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html

17 The upgrade procedure is described in the section “Upgrading a v.2.x Security Database” in the Firebird 3.0.2 Release Notes https://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rlsnotes30.html.