

FBScanner User Guide

Version 3.0.67

Contents

| | |
|---|----|
| What is FBScanner?..... | 2 |
| Issues that FBScanner can help to resolve | 2 |
| Performance Impact..... | 2 |
| How to Setup FBScanner? | 4 |
| How to configure FBScanner for local computer? | 5 |
| How to register FBScanner? | 6 |
| How to setup FBScanner for remote computer? | 8 |
| How to setup logging?..... | 10 |
| Logging to text files | 11 |
| Example of text file logging | 12 |
| Logging to Firebird database..... | 14 |
| Transactions markers | 15 |
| Using Embedded Firebird 2.5 for SQL log | 15 |
| How to analyze FBScanner log? | 17 |
| How to track 10054 errors, disconnects and failed login attempts? | 21 |
| Backup/restore and mass load operations | 23 |
| Real-Time Monitoring: FBScanner Viewer | 24 |
| Tags | 25 |
| FBScanner Viewer Menu | 26 |
| Server | 27 |
| Connections..... | 27 |
| Kill..... | 28 |
| Transactions | 29 |
| Tools | 29 |
| SQL log structure | 30 |
| Logical structure | 31 |
| FBScanner Feature Matrix..... | 33 |
| Support..... | 36 |

What is FBScanner?

FBScanner (Firebird Scanner) is a tool that can monitor and view all traffic between Firebird and InterBase servers and their client applications. It shows the real-time activity of connected clients:

- Connections (IP/Name, duration, CPU load),
- Queries (query text, status, parameters)
- Transactions (with parameters).

FBScanner can log all SQL traffic to text files and external Firebird database, it includes FBScanner LogAnalyzer module to analyze SQL performance.

FBScanner can be used to profile database applications, monitor user activity, manage database connections (including client disconnects on Classic, SuperClassic and SuperServer architectures). It's also ideal for troubleshooting INET errors, as well as auditing existing applications and performance tuning.

FBScanner supports Firebird (V1.x, V2.0, V2.1 and V.2.5), InterBase (V4.0 to 2009/XE). It is a useful tool for analyzing production databases, especially if the application has been developed by third-party and there is no source code available.

FBScanner is transparent as far as the database application is concerned and does not require any changes in application or database source code, logic or configuration.

Issues that FBScanner can help to resolve

- Real-time monitoring of connections. FBScanner shows all connections to the selected database server: the IP/DNS name of connected client, database and connection time.
- Real-time monitoring of SQL queries. For each connection FBScanner shows all the currently running SQL queries along with their transaction parameters.
- Detection of the oldest connection and the oldest active transaction to allow you to analyze that may have non-optimal transaction behavior or incorrect transaction design or show users who might be using the application in a manner that may be affecting performance.
- Client disconnects. Check that disconnects are taking place correctly. You can also use FBScanner to disconnect users in order to perform maintenance or database upgrades.
- FBScanner allows the routing of specific applications or particular users to allow you to zoom in on specific applications or users.
- You can log SQL queries. For debugging or for security FBScanner can log all the selected traffic to a special database for further analysis. FBScanner includes LogAnalyzer tool to find bad queries and ineffective SQL plans.

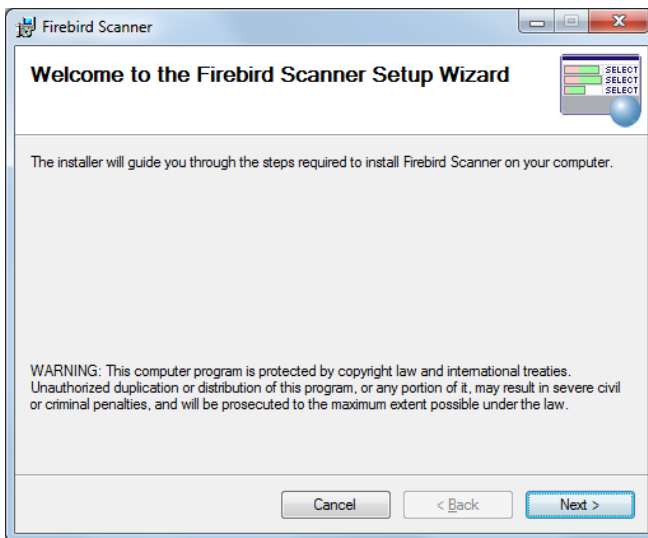
Performance Impact

FBScanner does not change anything in transferred SQL traffic and works simply like a transparent proxy, so all applications will work normally.

FBScanner consume approximately 50-150Mb of memory (for 30-100 active clients), and it will decrease database performance by 5% to 10%.

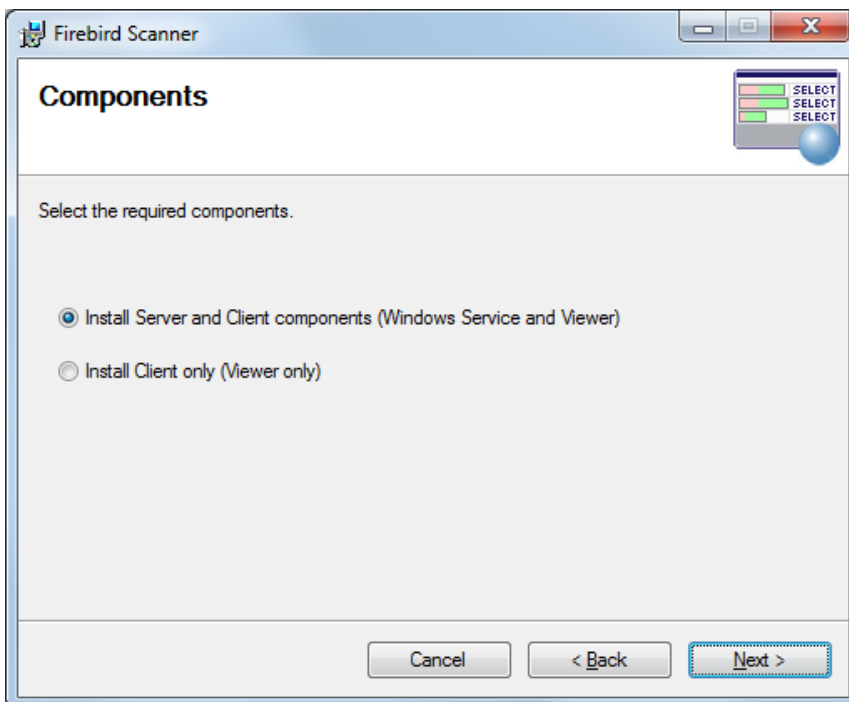
How to Setup FBScanner?

Run Setup.exe

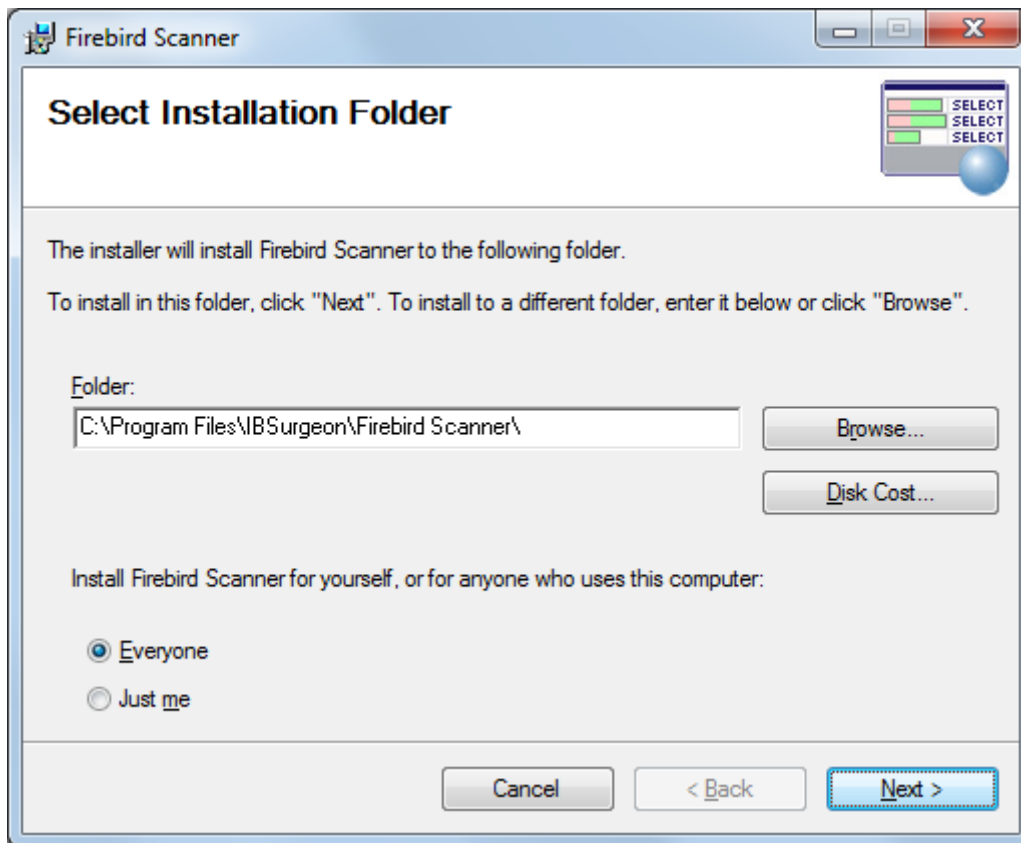


Click "I agree" for License Agreement, and then click Next.

At this step you need to choose do you need full installation of FBScanner or Viewer only.



Choose where to setup FBScanner.



Click Next at several screens, and then wait several seconds for completion of installation.

If you are using Windows 7/Vista, Windows Server 2008 and above, you can see system dialog with permission request for FBScanner setup. Click appropriate button to allow its actions.

After that FBScanner installation wizard will start “FBScanner Configuration” tool. It can be started manually from Start menu, “Firebird Scanner\FBScanner Settings”.

How to configure FBScanner for local computer?

Start “FBScanner Configuration” to configure FBScanner from Start menu (“Firebird Scanner” folder). This tool will help you to setup both basic and advanced configuration parameters for FBScanner.

Basic configuration parameters are shown at the main screen of “FBScanner Configuration”. It scans Windows registry for installed Firebird services and show them in the grid.

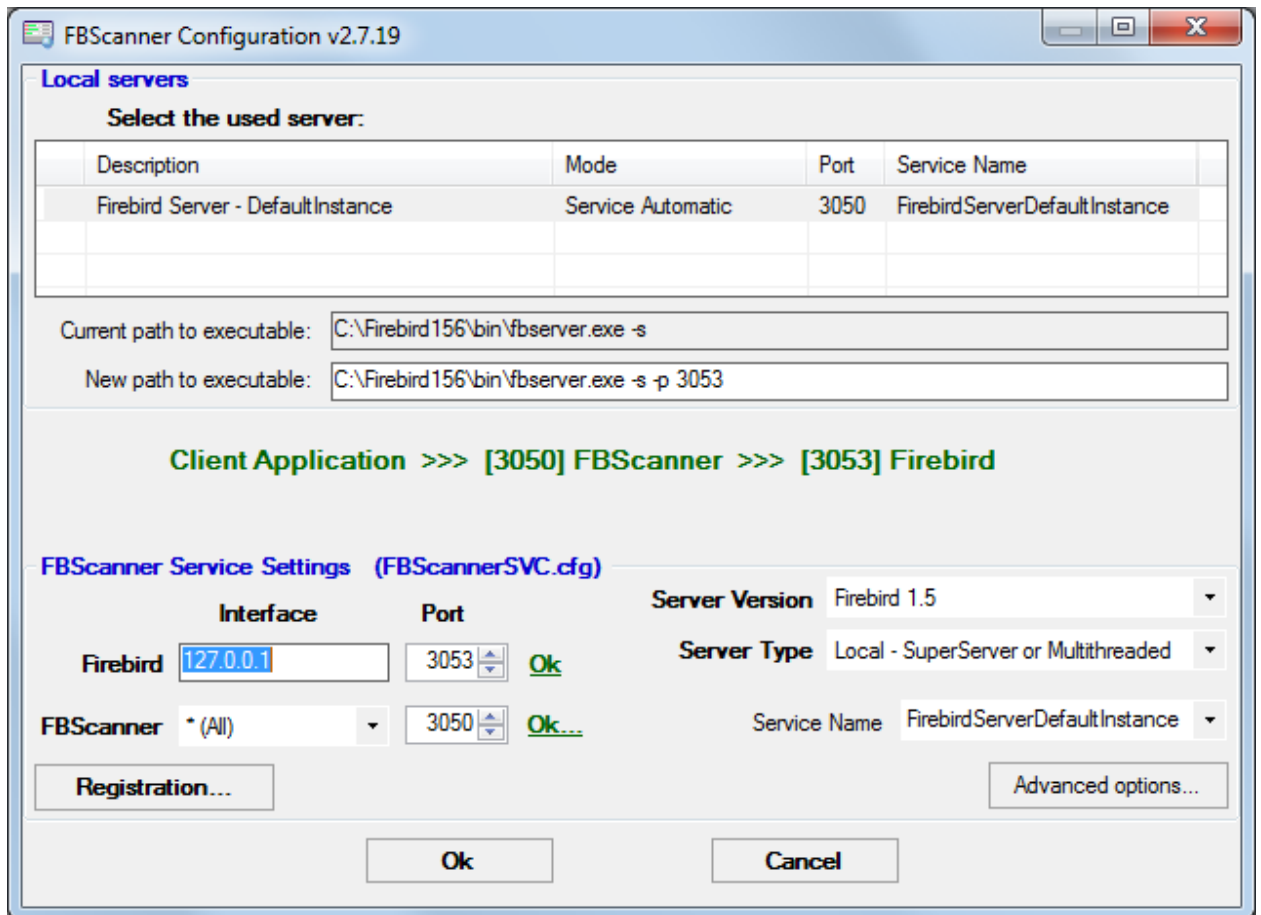
By default Firebird uses port 3050 for network connections. FBScanner works as a transparent TCP proxy – it redirects all SQL traffic from and to Firebird clients to another.

FBScanner offers to change Firebird port to 3053, in order to start its own instance at 3050. FBScanner checks for the port usage and if either 3050 or 3053 are used by other software (not Firebird), it will warn you with red caption “Port used” near new “Port” text box.

The green figure in the center of “FBScanner Configuration” main screen briefly shows how client applications SQL traffic will be passed.

At the figure below you can see that FBScanner found Firebird 1.5 instance, and offers to change its port to 3053, in order to set own instance to listen at 3050.

Such default scenario will give the maximum compatibility with existing Firebird clients (i.e., end-user applications).

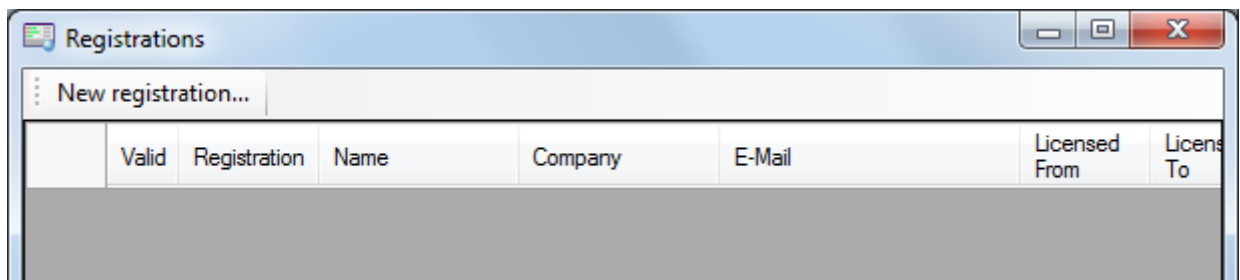


To approve the changes, click “Ok”, otherwise “Cancel”.

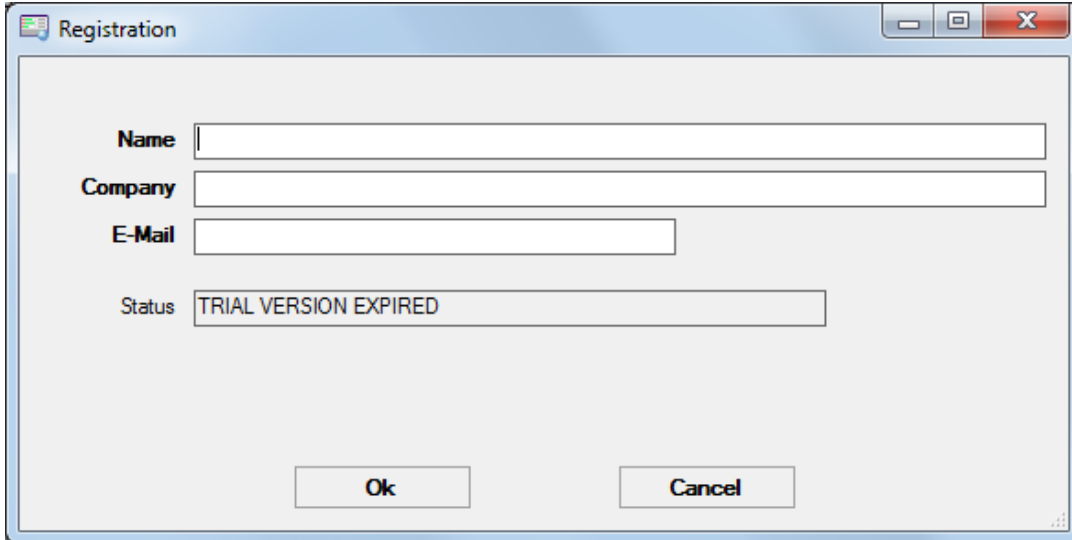
Important! IF FBScanner settings were changed, FBScanner Service will be restarted, and all existing Firebird connections will be dropped! Be careful with changing FBScanner settings in production environment. FBScanner will ask your permission to restart, please decide carefully.

How to register FBScanner?

Run “FBScanner Configuration” tool and click “Registration” button (in the left bottom corner of the main page).



Click “New Registration”:

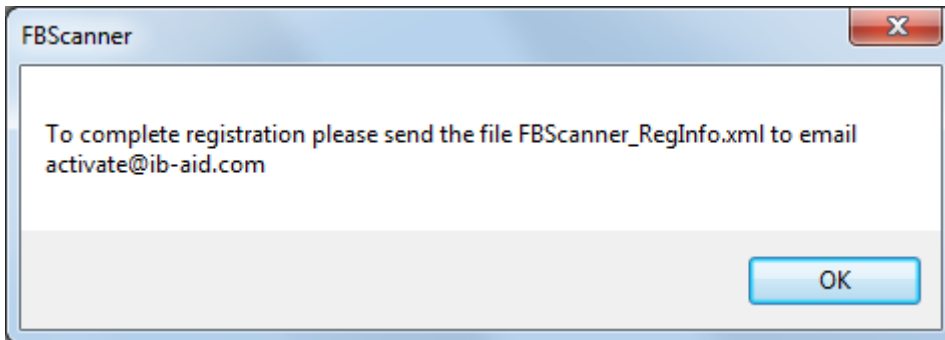


The Registration dialog box contains the following fields:

- Name**: [Empty text box]
- Company**: [Empty text box]
- E-Mail**: [Empty text box]
- Status**: TRIAL VERSION EXPIRED

Buttons: **Ok**, **Cancel**

Then fill out Name, Company and Email and click Ok. The following message box will appear:



The message box contains the following text:

To complete registration please send the file FBScanner_RegInfo.xml to email activate@ib-aid.com

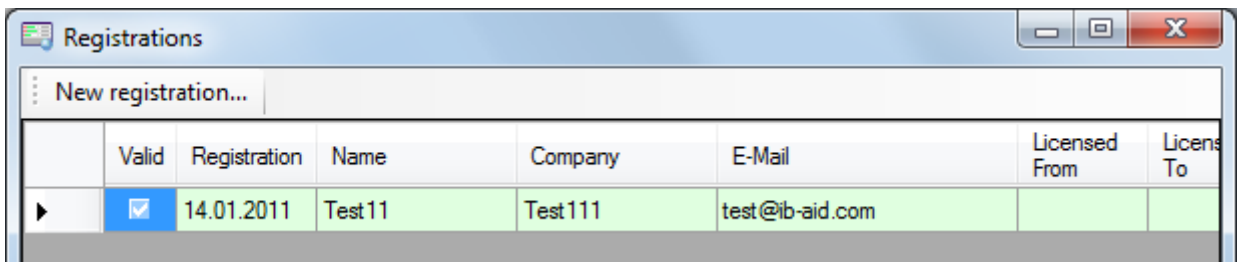
Button: **OK**

Go to the folder where FBScanner is installed (by default it's C:\Program Files\IBSurgeon\Firebird Scanner) and send file FBScanner_RegInfo.xml to activate@ib-aid.com.

Shortly you'll get the answer with the same file attached, but this time it will contain necessary registration key. You need to replace existing FBScanner_RegInfo.xml with new received file.

To apply registration immediately, restart FBScanner (Warning! All existing Firebird connections through FBScanner will be dropped) or wait until 00-00 AM – FBScanner Service will re-read FBScanner_RegInfo.xml and will see new settings.

After that you can check the status of your registration:



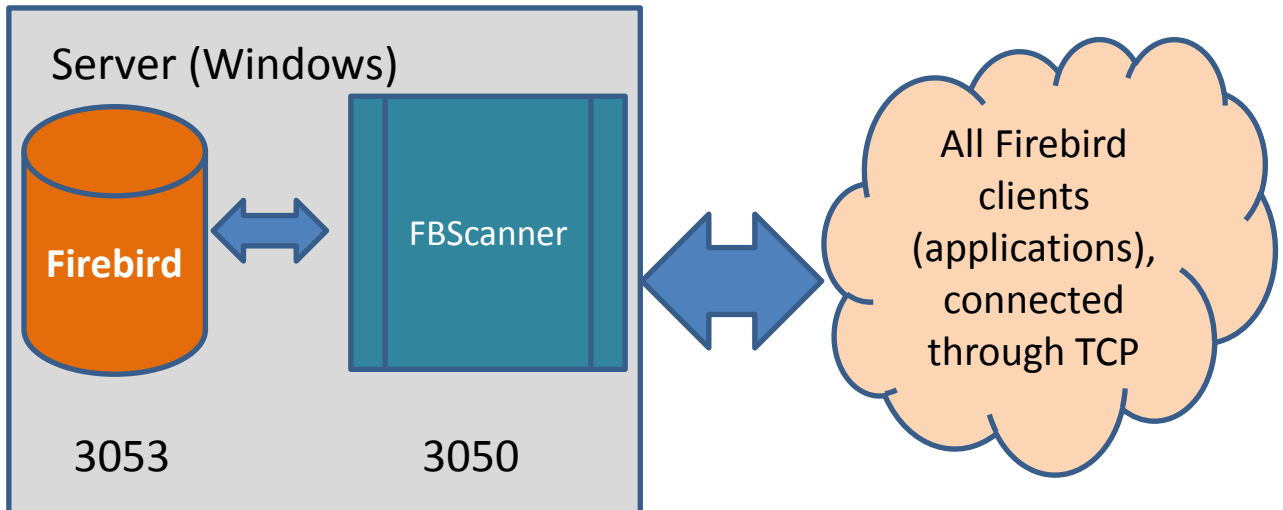
The Registrations window shows a table with the following data:

| | Valid | Registration | Name | Company | E-Mail | Licensed From | Licens To |
|---|-------------------------------------|--------------|--------|---------|-----------------|---------------|-----------|
| ▶ | <input checked="" type="checkbox"/> | 14.01.2011 | Test11 | Test111 | test@ib-aid.com | | |

How to setup FBScanner for remote computer?

FBScanner can route SQL traffic not only as local proxy, but from another computer too. To understand the difference and discover consequences, let's walk through details.

The basic (and default) configuration of FBScanner implies that it works on the same computer where Firebird is working, and process all SQL traffic from Firebird clients (i.e., end-user applications) which use default connection string (and, therefore, port 3050).



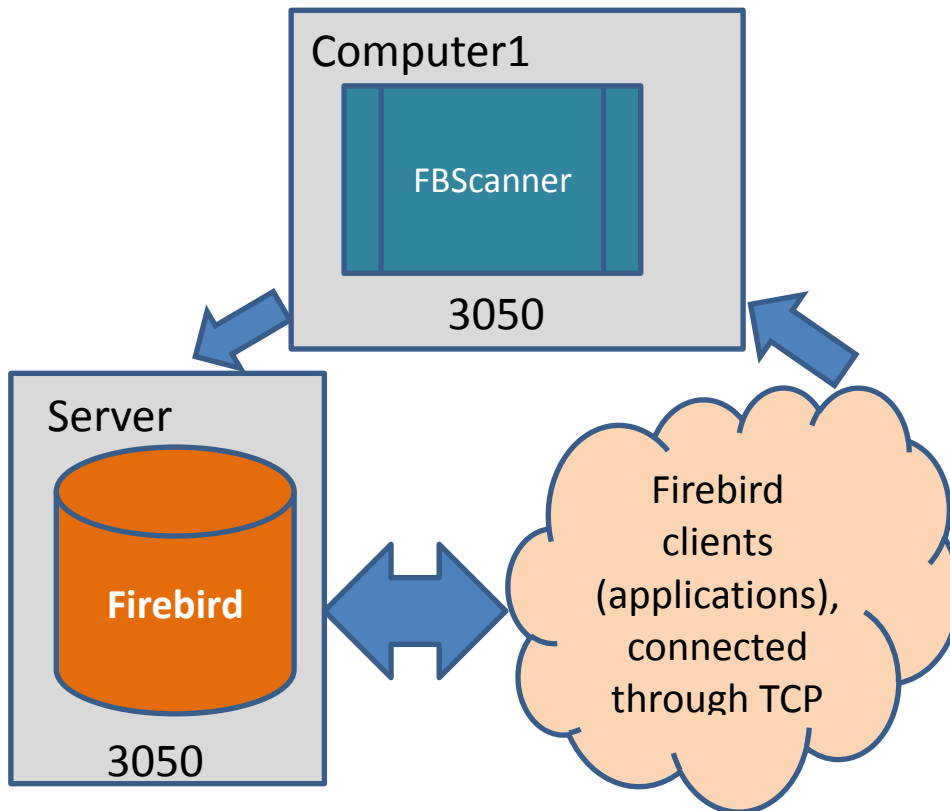
Sometimes it's not convenient to setup FBScanner to process all requests, for example, in case of:

- Only several (may be, the single workstation) workstations need to be profiled/logged
- Only certain application or narrow functionality need to be profiled
- Developers need to check some SQL code on the live database – gather SQL log with execution statistics, plans, etc.
- Heavy load (too many workstations). In case of heavy load FBScanner can consume resources of the main server, and it's better to move FBScanner (as well as FBScanner log, if it's enabled, to the dedicated computer).
- Linux server. If Firebird works on Linux, it's possible to route SQL traffic through remote instance of FBScanner on Windows.

In these cases the good idea is to setup FBScanner at the remote computer and pass only part of SQL traffic through it. It also makes possible to perform necessary analysis of SQL without changing ports or other configuration at server – the only needed adjustment will be change host name in client applications' connections strings.

One of the frequent use cases for setting up FBScanner in remote configuration is using it as debug console for developer computer, so developer can see in real-time (with FBScanner LogViewer) or afterwards (with FBScanner LogAnalyzer) all SQLs from own computer to the Firebird server..

At the figure below you can see how it can look like:



Now let's back to the configuration and see how easy to setup FBScanner to route SQL traffic at the remote computer.

At the bottom of the main screen of "FBScanner Configuration" you can see the following default settings (for Firebird 1.5 example we considered above):

The screenshot shows the 'FBScanner Configuration' window. At the top, it displays 'Client Application >>> [3050] FBScanner >>> [3053] Firebird'. Below this, there are two sections: 'FBScanner Service Settings (FBScannerSVC.cfg)' and 'Server Settings'. In the 'FBScanner Service Settings' section, there are two rows: one for 'Firebird' with 'Interface' set to '127.0.0.1' and 'Port' set to '3053', and one for 'FBScanner' with 'Interface' set to '(All)' and 'Port' set to '3050'. In the 'Server Settings' section, 'Server Version' is set to 'Firebird 1.5', 'Server Type' is set to 'Local - SuperServer or Multithreaded', and 'Service Name' is set to 'FirebirdServerDefaultInstance'.

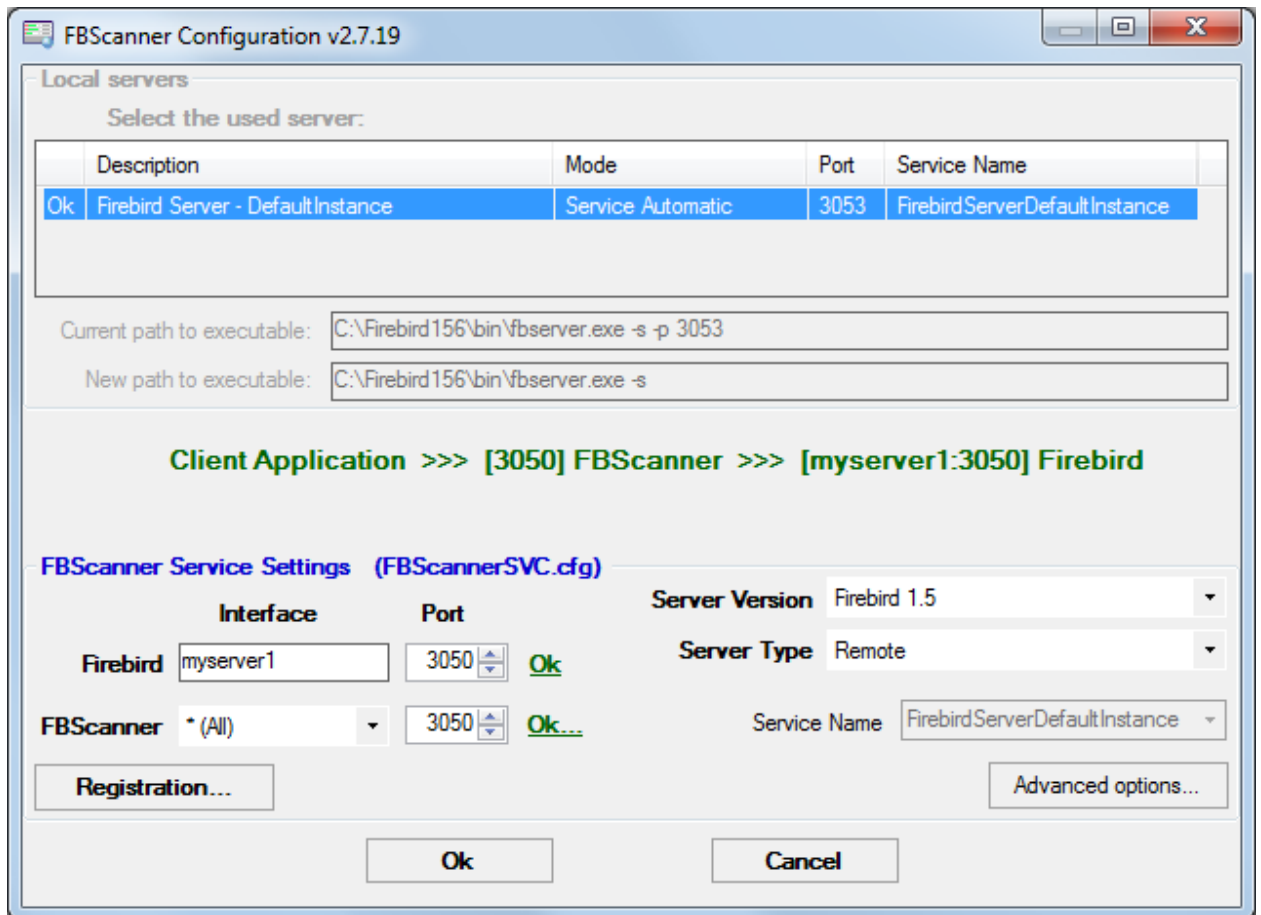
In order to setup FBScanner to route SQL traffic to the remote Firebird, we need to change "Server Type" from "Local..." to "Remote". It will change the main screen of the configuration tool.

First of all, we need to specify network name (or IP) of the computer with Firebird instance and port where it will be used – it should be entered into "Interface" textbox.

Then we need to specify Firebird version – in our example it's Firebird 1.5.

FBScanner instance also has "Interface" –it's the list of network adapters found at the computer. If you need to bind FBScanner to one of them and disable connections from other network adapters, choose one of the adapters from the drop-down list. By default FBScanner will accept Firebird clients' requests from all network adapters.

Below you can see the example of FBScanner configuration to route SQL traffic to remote Firebird instance, which resides on **myserver1** computer and works on default port 3050.

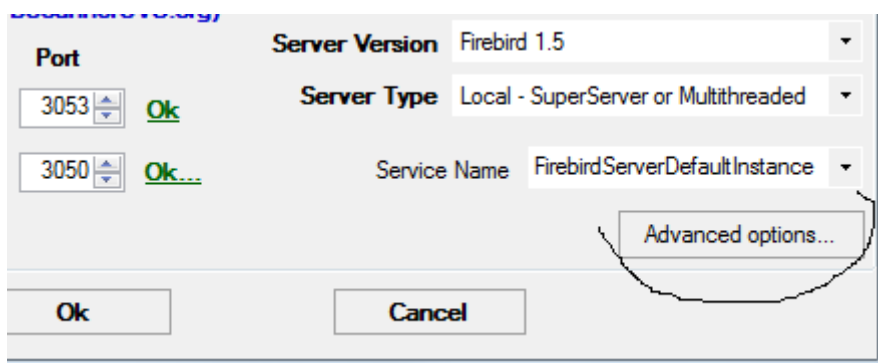


Click “Ok” to confirm new settings, and FBScanner will route SQL requests to the remote Firebird.

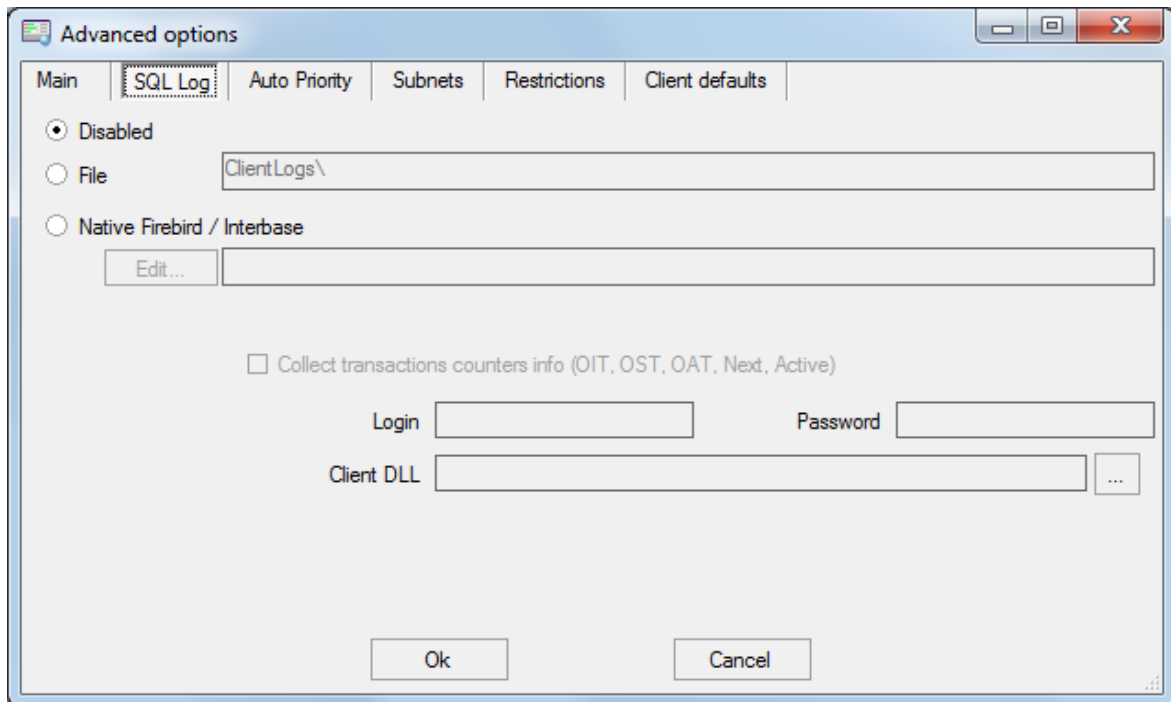
Important! If you need to pass SQL traffic from client applications through remote FBScanner, please change Firebird appropriate connection string. For example, if originally client applications have connected with “**myserver1:C:\Database\data.fdb**”, in order to pass SQL traffic through FBScanner in this example you need to change connection string to “**computer1: C:\Database\data.fdb**” (where computer1 is the network name of the computer where FBScanner works).

How to setup logging?

From Start menu run “Firebird Scanner\FBScanner Settings”, then click button “Advanced options” (in the right bottom of the main screen).



At the dialog click tab “SQL log”.



By default logging is disabled.

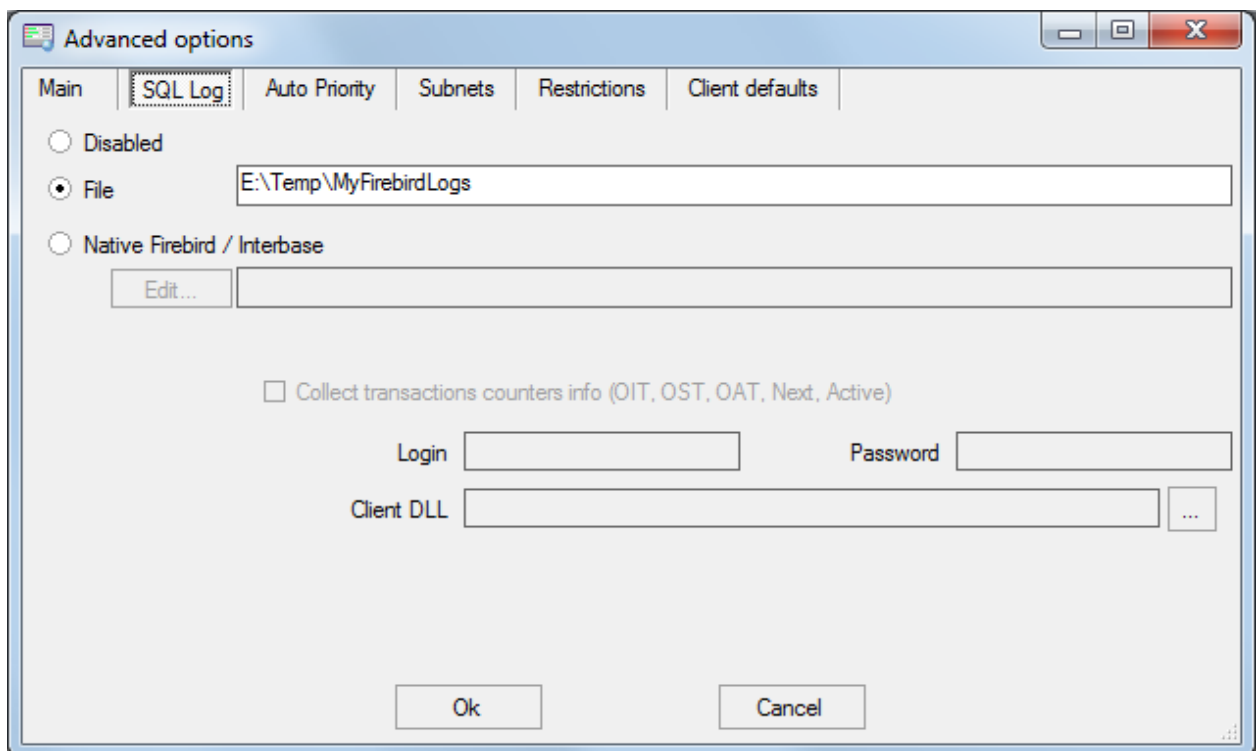
Important! It's important to understand that logging to SQL database will write all SQL operations, including transactions, connects, etc. It means that SQL log database will consume the same amount of resources (CPU, HDD, etc) as the main database does. Due to this fact for heavy load environments we recommend to use remote configuration of FBScanner for SQL logging.

There are 2 options for logging - to file and to Firebird log database.

Logging to text files

File logging creates text file for each connection where FBScanner writes SQL and transactions operators. We recommend file logging for debug purposes and during development – it's suitable to investigate linear SQL code. If there are a lot of connections, file logging becomes not very suitable.

To enable file logging, click radio button near “File” option and set folder where to store file logs (check that specified folder exists first!):



Then click Ok.

Important! Enabling logging will require restart of FBScanner Service, so all current connections will be dropped. FBScanner will ask your permission to do it immediately.

Example of text file logging

For the following isql commands

```
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect "localhost:E:\Temp\TEST15_2.FDB";
Database: "localhost:E:\Temp\TEST15_2.FDB"
SQL> create table t1(i1 integer, c1 varchar(150));
SQL> create table t2(i2 integer, b1 blob);
SQL> select count(*) from t1;
      COUNT
=====
          0
SQL> insert into t1(i1, c1) values(1, 'test');
SQL> select count(*) from t1;
      COUNT
=====
          1
SQL> exit;
```

FBScanner created the following log:

```
/* Log created by FBScanner v2.7.19
   14.01.2011 16:06:07
      Client IP      = 127.0.0.1
      Client Name    = ibsurgeon3
      Client Process = isql [1884]
*/
CONNECT '127.0.0.1/3053:E:\Temp\TEST15_2.FDB' USER 'SYSDBA';
```

```

/* 14.01.2011 16:06:09 */
/* TrID=20; */
SET TRANSACTION READ WRITE WAIT SNAPSHOT;

/* 14.01.2011 16:06:09 */
/* TrID=22; isc_tpb_version1, isc_tpb_write, isc_tpb_read_committed,
isc_tpb_wait, isc_tpb_no_rec_version */
SET TRANSACTION READ WRITE WAIT ISOLATION LEVEL READ COMMITTED NO
RECORD_VERSION;

/* 14.01.2011 16:06:19 */
/* QrID=26 TrID=22; EXECUTE */
create table t1(i1 integer, c1 varchar(150));

/* 14.01.2011 16:06:19 */
/* QrID=26 TrID=22; INFO */

/* 14.01.2011 16:06:19 */
/* TrID=22; */
COMMIT;

/* 14.01.2011 16:06:33 */
/* TrID=27; isc_tpb_version1, isc_tpb_write, isc_tpb_read_committed,
isc_tpb_wait, isc_tpb_no_rec_version */
SET TRANSACTION READ WRITE WAIT ISOLATION LEVEL READ COMMITTED NO
RECORD_VERSION;

/* 14.01.2011 16:06:33 */
/* QrID=31 TrID=27; EXECUTE */
create table t2(i2 integer, b1 blob);

/* 14.01.2011 16:06:33 */
/* QrID=31 TrID=27; INFO */

/* 14.01.2011 16:06:41 */
/* TrID=32; isc_tpb_version1, isc_tpb_write, isc_tpb_read_committed,
isc_tpb_wait, isc_tpb_no_rec_version */
SET TRANSACTION READ WRITE WAIT ISOLATION LEVEL READ COMMITTED NO
RECORD_VERSION;

/* 14.01.2011 16:06:41 */
/* QrID=36 TrID=20; EXECUTE */
select count(*) from t1;

/* 14.01.2011 16:06:41 */
/* QrID=36 TrID=20; INFO */

/*
Fetch count      = 1
*/

/* 14.01.2011 16:07:11 */
/* QrID=38 TrID=20; EXECUTE */
insert into t1(i1, c1) values(1, 'test');

/* 14.01.2011 16:07:17 */
/* QrID=40 TrID=20; EXECUTE */
select count(*) from t1;

/* 14.01.2011 16:07:17 */
/* QrID=40 TrID=20; INFO */

/*

```

```

Fetch count      = 1
*/

/* 14.01.2011 16:07:26 */
/* TrID=32; */
COMMIT;

/* 14.01.2011 16:07:26 */
/* TrID=27; */
COMMIT;

/* 14.01.2011 16:07:26 */
/* TrID=20; */
COMMIT;

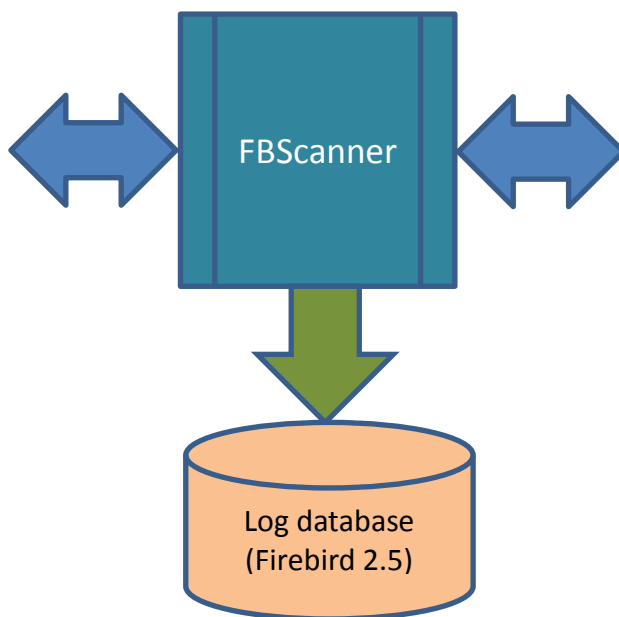
```

As you can see, file log is useful to understand how SQL commands were run inside the single connect.

Logging to Firebird database

Before you start with SQL log, it's necessary to understand some implementation details, which can be important for production systems.

In general logging to Firebird database is implemented in the straightforward way: FBScanner service writes all traffic to the external Firebird database. Firebird database with log can be at the same computer where FBScanner resides, or at the remote computer.

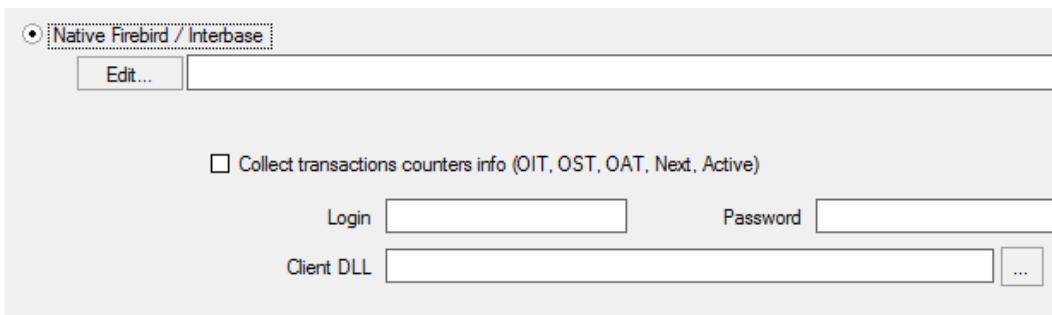


Please consider the following requirements for SQL log configuration:

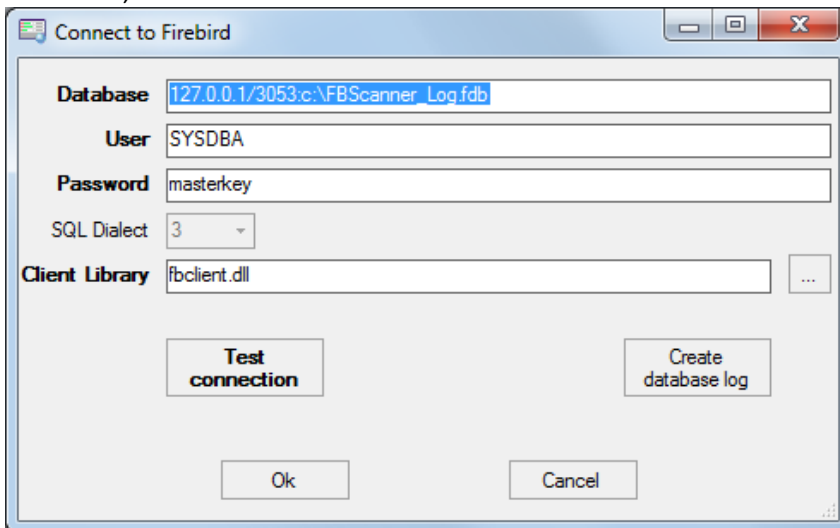
- Log database (and appropriate Firebird instance) should be in Firebird 2.5 format (since FBScanner 2.7.15). If you are forced to use FBScanner at the computer with another Firebird version, you need to use embedded Firebird 2.5 to store log.
- SQL traffic from all logged connections is written into the single table, with appropriate markers (from what computer, application, user, etc. this particular record was created).
- Log database can consume significant amount of resources in case of heavy load. For many connections it's recommended to setup FBScanner and Firebird log database at dedicated computer.
- In many cases it's not necessary to log all connections, because they repeat the same set of SQL queries. Careful investigation of the single connection can be the most effective way to find performance problems.

To enable SQL logging, click on "SQL" radio button. It will enable appropriate text boxes and controls.

© IBSurgeon 2002-2011



First of all, click button “Edit”.



Important! If you intend to use the same Firebird instance to log SQL traffic, you need to specify connections string with explicit and direct port. In our example it will be port 3053, and connection string looks like **127.0.0.1/3053:C:\FBScanner_log.fdb**

In this dialog you also need to specify how to connect to database with log. If there is no database with specified name, create new database – click “Create database log”. Test connection with log database - click “Test connection”.

Click “Ok” to save settings.

Transactions markers

FBScanner can gather information about transactions markers (in the same way like IBSurgeon Transaction Monitors does). Gathered information will be shown as graphs in FBScanner Log Analyzer.

For this purpose FBScanner runs separate connect, which requires Login, Password and path to the appropriate client dll (if you track Firebird 1.5 with FBScanner, fbclient.dll from 1.5 will be required).

If you decide to gather transactions markers information, mark checkbox “Collect transactions counters info” and fill out Login, Password and Client DLL fields.

Using Embedded Firebird 2.5 for SQL log

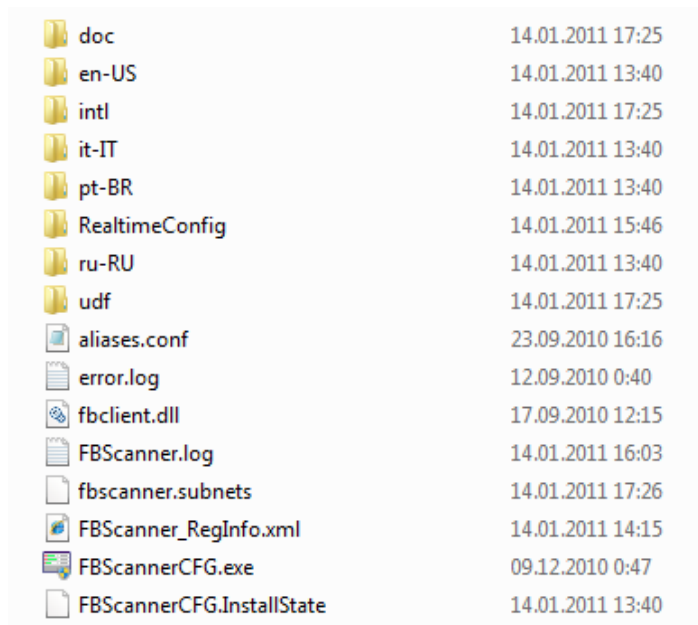
If you need to use SQL log at the computer where old Firebird is used (1.0, 1.5, 2.0., 2.1 or even InterBase), it’s recommended to use Firebird 2.5 Embedded to store log.

Download Firebird 2.5 Embedded from here

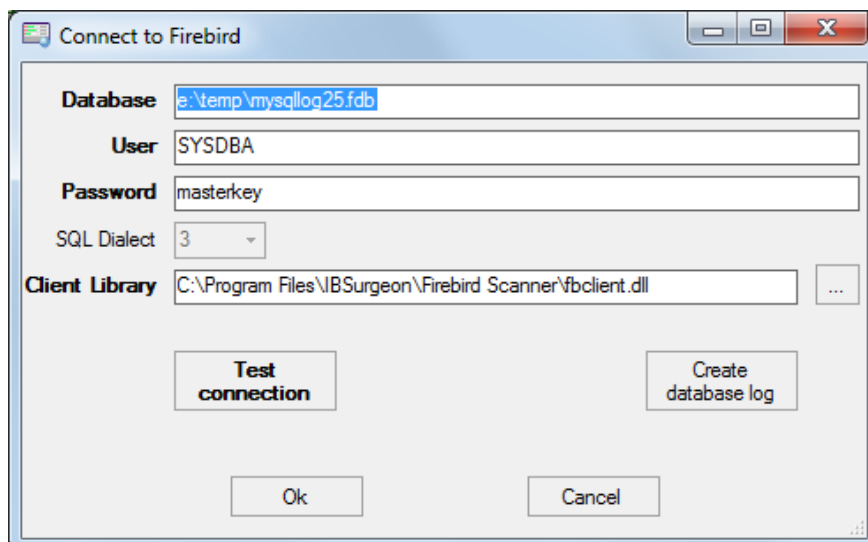
http://sourceforge.net/projects/firebird/files/firebird-win32/2.5-Release/Firebird-2.5.0.26074-0_Win32_embed.zip/download

Unpack the archive right into the FBScanner folder (C:\Program Files\IBSurgeon\Firebird Scanner by default) and rename fbembed.dll into fbclient.dll.

Folder structure will look like this



After that run “Advanced options”, tab “SQL logging”, radio button “SQL” and click “Edit”, then in the “Client library” point to the renamed fbclient.dll, as it shown below.



Tip. In Embedded Firebird fbclient.dll represents the whole engine. It works inside the process of FBScanner and there is no interaction with other installed Firebird instances, both full and embedded.

How to analyze FBScanner log?

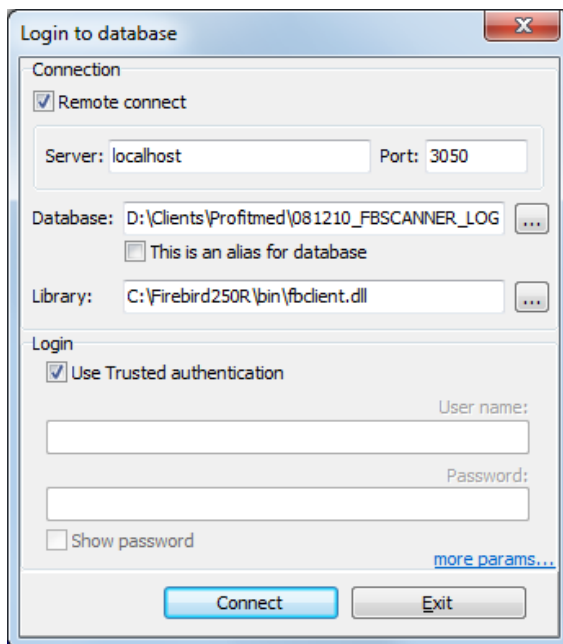
Many users told us that they did not realize how many queries, transactions and other operations are performed by their software. As you remember, FBScanner stores all information into the single table. It uses self-links to reduce the amount of stored information and it makes raw log hard to read and understand.

To facilitate log analysis we have created new module in FBScanner - LogAnalyzer. It's available in IBSurgeon Deploy Center for all FBScanner users (inside "Download" section).

LogAnalyzer requires Firebird 2.5 to work with log database. It also creates new indices and runs heavy reporting queries, so it's recommended the following procedure:

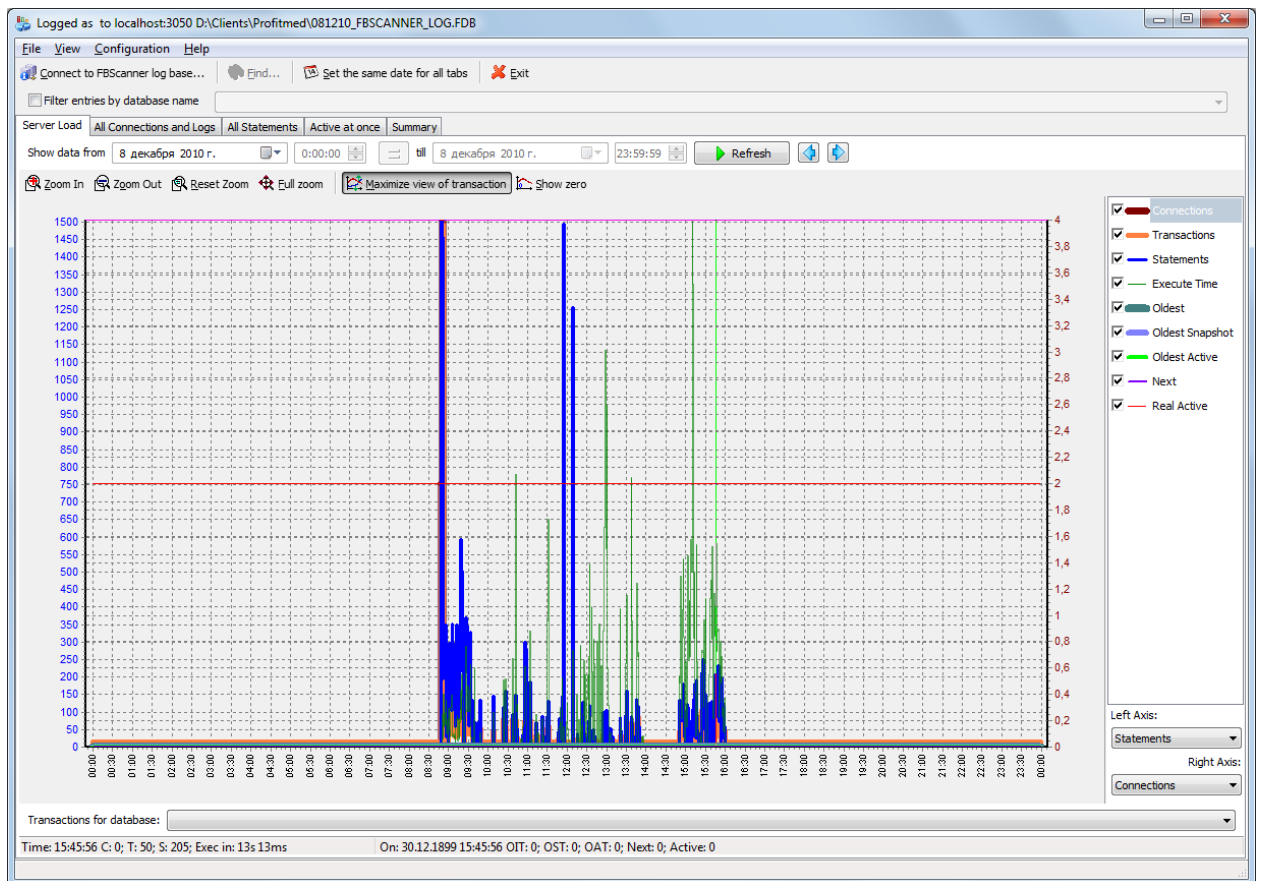
- 1) Setup logging and gather statistics for at least 1 day
- 2) Copy log database to another computer with Firebird 2.5
- 3) Connect to the copy of log database and perform analysis at the developer's computers
- 4) Copy updated versions of log databases as necessary

To analyze log database, start LogAnalyzer and click "Connect to FBScanner log base", then fill out connection parameters and select log database.



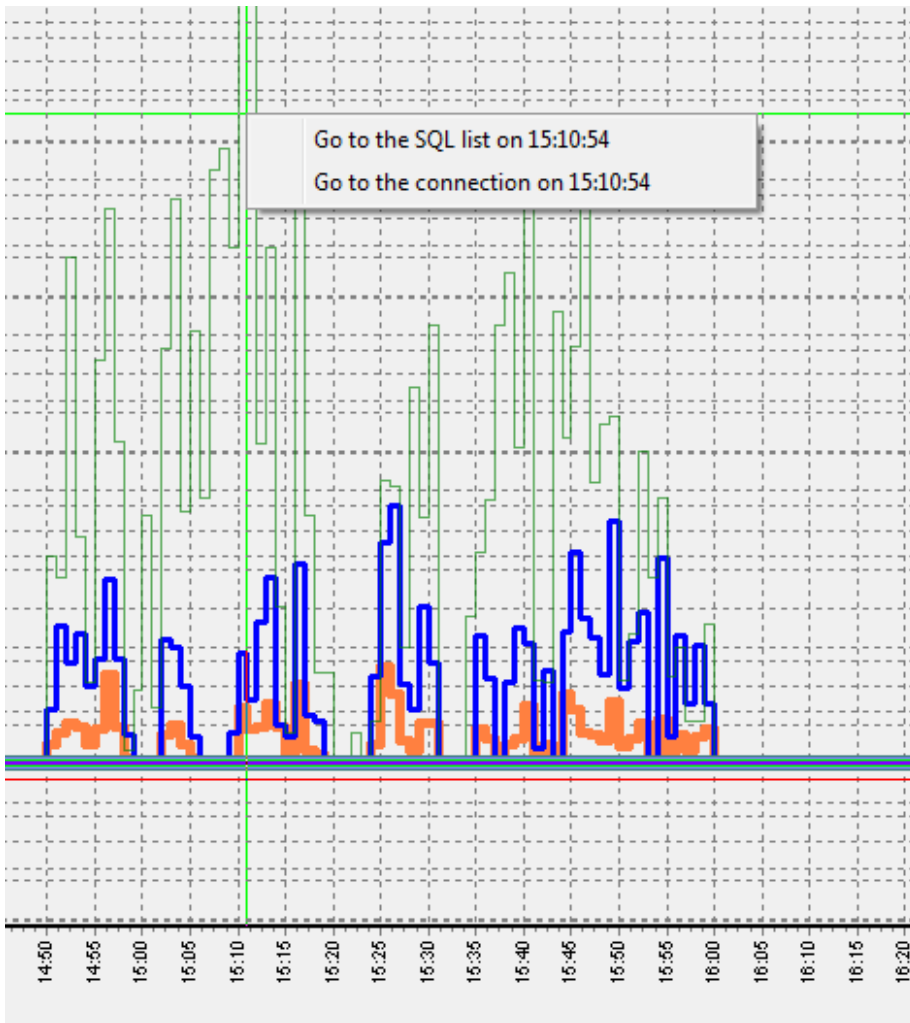
At first start LogAnalyzer will create necessary indices, it can take several minutes.

After that LogAnalyzer will show the last available day in the log at the "Server Load" tab:



“Server Load” tab shows how many SQL queries were run per minute, and how much time they took to execute. Effectively it shows server load, i.e., number of queries and their execution times.

Zoom in (button in the top left corner of the tab “Server load”), drag graph by holding right-button of the mouse and select the peak you are interested to investigate – click right-button to show popup-menu



It will show you tab “All statements”, where you can browse SQL queries

| | | | | | | | | | | | | | |
|--------|--------|--------|-------|--------------------|---|-----|--------------------|--------------------------------|--------------------------------|-----------------------------|----|---|------------|
| 211232 | 186212 | 186215 | 17299 | 08.12.2010 15:11:0 | 0 | 0 | 08.12.2010 15:11:0 | select d.*, dt.*, d1.xec_5034f | select d.*, dt.*, d1.xec_5034f | PLAN (LIN INDEX (LIN_EIDDOC | 1 | 0 | 08.12.2010 |
| 211256 | 186212 | 186215 | 17300 | 08.12.2010 15:11:0 | 0 | 437 | 08.12.2010 15:11:0 | SELECT | SELECT | PLAN JOIN (X51101 INDEX (I | 47 | 0 | 08.12.2010 |
| 211235 | 186212 | 186215 | 17301 | 08.12.2010 15:11:0 | 0 | 0 | 08.12.2010 15:11:0 | SELECT | SELECT | PLAN (XECINT INDEX (IDX_XEC | 1 | 0 | 08.12.2010 |
| 211236 | 186212 | 186215 | 17303 | 08.12.2010 15:11:0 | 0 | 0 | 08.12.2010 15:11:0 | SELECT osdatabase_kind | | PLAN (RDBSDATABASE NATUR | 1 | 0 | 08.12.2010 |


```

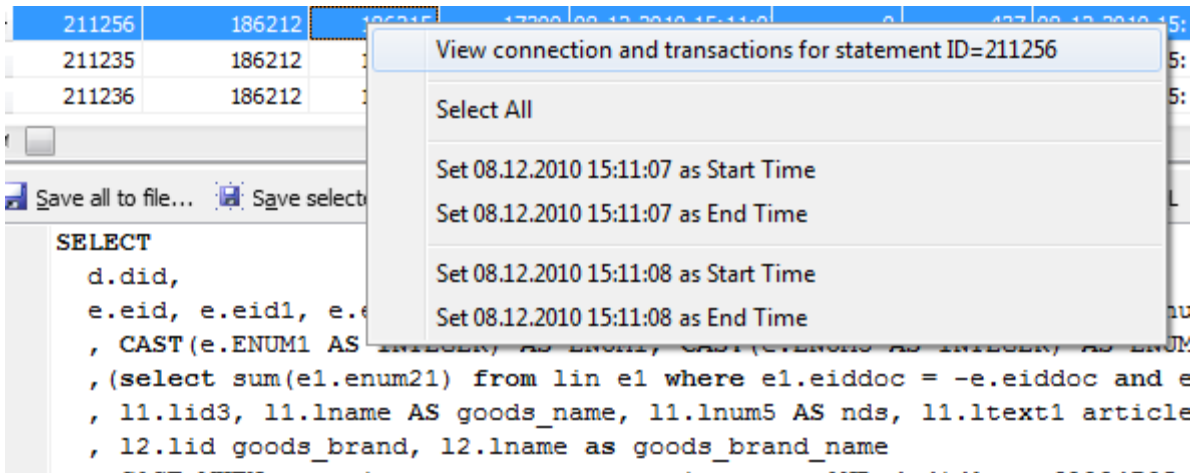
SELECT
  d.did,
  e.eid, e.eid1, e.eid2, e.eid3, e.eid4, e.eid5, e.eid6, e.enum4, e.enum14, e.eint2
  , CAST(e.ENUM1 AS INTEGER) AS ENUM1, CAST(e.ENUM3 AS INTEGER) AS ENUM3, e.enum9, e.enum15, e.e
  ,(select sum(e1.enum21) from lin e1 where e1.eiddoc = -e.eiddoc and e1.eid1 = e.eid1) add_sum
  , l1.lid3, l1.lname AS goods_name, l1.lnum5 AS nds, l1.txtxt1 article
  , l2.lid goods_brand, l2.lname as goods_brand_name
  , CASE WHEN pr.prime_cost_opt > pr.prime_cost AND d.did1 <> 60034705 THEN pr.prime_cost_opt EL
  , l2.lid1
  , l4.lname AS state_name
  , (SELECT f3 FROM percent(e.enum4,e.enum5)) p_price
  , e.enum1 * e.enum4 AS enum14_ent
  , CAST(0 AS INTEGER) AS selected
  , m.managername
    
```

```

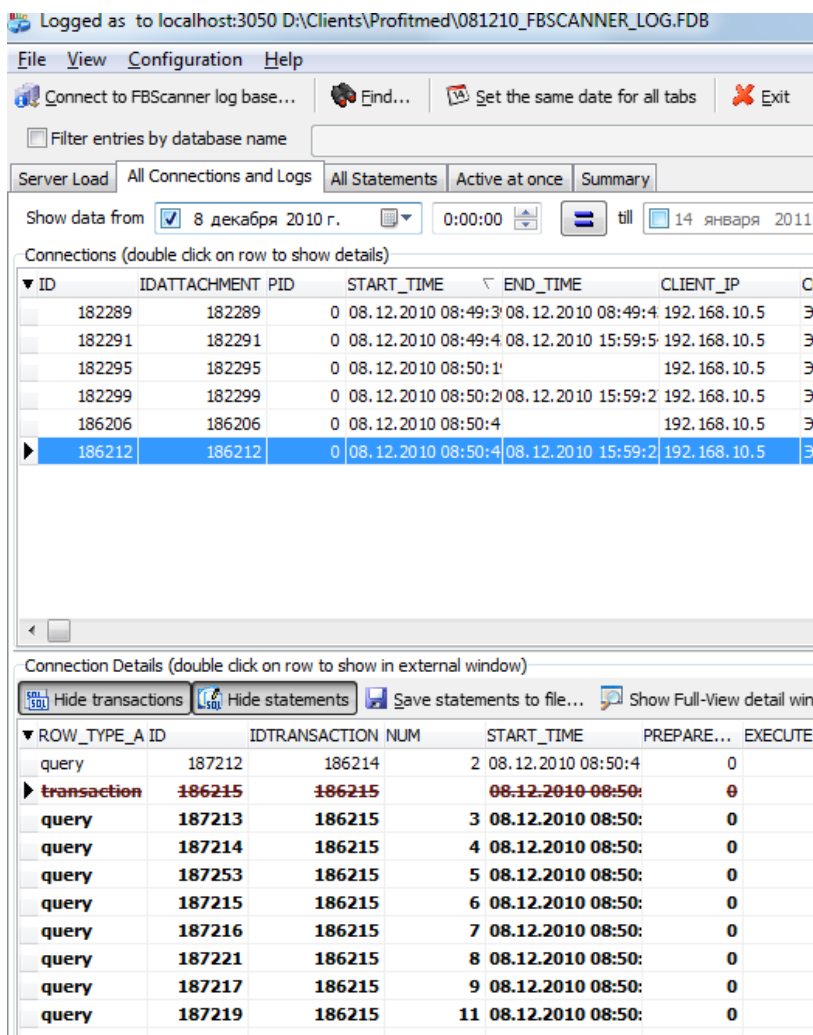
PLAN JOIN (X51101 INDEX (IDX_XECINT5), LIB INDEX (R
PLAN (LIB INDEX (RDB$PRIMARY11))
PLAN JOIN (X51100 INDEX (IDX$XECINT_OWNER_PARENT_NO
PLAN (LAST INDEX (RDB$PRIMARY5))
PLAN (LIN INDEX (LIN_EIDDOC_EID1_EID3_EID5))
PLAN (XECINT INDEX (IDX_XECINT5))
PLAN (XECNUM INDEX (IDX_XECNUM4))
PLAN (LIB INDEX (RDB$PRIMARY11))
PLAN (PERCENT NATURAL)
PLAN (E1 INDEX (LIN_EIDDOC_EID1_EID3_EID5))
PLAN JOIN (JOIN (JOIN (JOIN (JOIN (JOIN (JOIN (D IN
    
```

Select any query to see its text and, if plan logging feature is enabled, its plan.

To follow the execution flow, you can right-click on the query and look for connection and transactions for this query



LogAnalyzer marks bold queries in the same transaction:



You can sort queries and, for example, find query with the longest execution time:

| ▼ ROW_TYPE_A ID | IDTRANSACTION NUM | START_TIME | PREPARE... | EXECU... | END_TIME |
|-----------------|-------------------|---------------|--------------|--------------------------|---------------------------|
| query | 207407 | 186215 | 14361 | 08.12.2010 12:57: | 0 32594 08.12.2010 |
| query | 211270 | 208928 | 17338 | 08.12.2010 15:11:2' | 0 30624 08.12.2010 1 |
| query | 207518 | 201265 | 14539 | 08.12.2010 13:00:0' | 0 14766 08.12.2010 1 |
| query | 207372 | 201225 | 14392 | 08.12.2010 12:58:0' | 0 14437 08.12.2010 1 |
| query | 214522 | 186215 | 19618 | 08.12.2010 15:43: | 0 14109 08.12.2010 |
| query | 214109 | 186215 | 19176 | 08.12.2010 15:37: | 0 13672 08.12.2010 |
| query | 211668 | 186215 | 17139 | 08.12.2010 15:09: | 0 12470 08.12.2010 |
| query | 208263 | 201460 | 15296 | 08.12.2010 13:37:5' | 0 11702 08.12.2010 1 |
| query | 207597 | 186215 | 14544 | 08.12.2010 13:00: | 0 10937 08.12.2010 |

To know more about this query - double-click on it and see more details

The screenshot shows a SQL Developer window titled 'SQL'. The window contains a SQL query and its execution plan. The query is a complex SELECT statement with multiple columns and subqueries. The execution plan shows several steps, including index scans and joins.

```

SELECT d.did,d.dcode,d.ddate1,d.ddate3,d.ddate4,d.did1,d.did6,d.did8,d.did10
, d.dstate,d.ddate,d.dparent,d.dnum1,d.dnum2,d.dnum5,d.dnum8,d.dint2,d.dda
, d.dnum9
, l.lname, l.ltext1
, (SELECT FIRST 1 rf2_b_line(val,1) FROM xecblob WHERE id = dt.taddress) a
, (SELECT lname FROM lib WHERE lid = d.did8) trend
, (SELECT kname FROM cfg WHERE kid = d.dstate) state_name
--
, (SELECT lname FROM lib WHERE lid = d.did9) manager_name
, lib2800.lname as manager_name
, lib5800$1.lname as dolgnost
, lib5800$2.lname as gruppa
, lib5800$3.lname as otdel
, (SELECT lname FROM lib WHERE lid = d.DAUTOR) author_name
, CASE WHEN dt.day_in_day = 2201 THEN 'Да' ELSE 'Нет' END day_in_day_name
, CASE WHEN dt.in_night = 2201 THEN 'Да' ELSE 'Нет' END in_night_name

```

```

PLAN (XECINT INDEX (IDX_XECINT5))
PLAN (XECINT INDEX (IDX_XECINT5))
PLAN (XECDATE INDEX (IDX_XECDATE5))
PLAN JOIN (XECINT INDEX (IDX_XECINT5),PMTWTERRTREE INDEX (PK_PMTWTERRTREE))
PLAN (X770100 INDEX (IDX_XECINT5))
PLAN JOIN (XI1 INDEX (IDX_XECINT5),XD1 INDEX (IDX_XECDATE5))
PLAN JOIN (JOIN (XI2 INDEX (XECINT_IDX2),XV1 INDEX (RDB$PRIMARY42)),XD1 INDEX

```

How to track 10054 errors, disconnects and failed login attempts?

FBScanner automatically logs all 10054 errors, disconnects and failed login attempts with detailed description in the FBScanner.log file, which is in FBScanner main directory.

```

19.08.2010 21:43:09
Connect Error
Client IP      = 192.10.1.2
Client Name   =
DB Name       =

```

```
DB User          = MORTON
Client Process   = SUPC [5520]
Client Process   (by fbclient) = E:\TEMP\TEST1.EXE [5520]
STATUS           = [file is not a valid database]
```

19.08.2010 21:43:25

Login Failed

```
Client IP        = 127.0.0.1
Client Name      = ibsurgeon3
DB Name          = C:\Program Files\Jupiter2010\Data\data.gdb
DB User          = MORTON
Client Process   = Jupiter.exe [3032]
Client Process   (by fbclient) = E:\TEMP\TEST1.EXE [3032]
STATUS           = [Your user name and password are not defined.]
```

Ask your database administrator to set up a Firebird login.]

Backup/restore and mass load operations

To perform operations which do not require monitoring or debugging, like backup and restore or mass load of records (in billing systems) we recommend bypassing FBScanner service.

If FBScanner is installed in default recommended configuration, i.e., on port 3050 and Firebird is on port 3053, connection strings should be like this

server_name/3053:Disk:\Path\database.fdb

example of connection string

connect "localhost/3053:C:\TEMP\database.fdb" user "SYSDBA" password "masterkey";

Example of using backup command

**gbak.exe -b -g -v -user SYSDBA -pass masterkey localhost/3053:C:\TEMP\database.fdb
C:\temp\backup.gbk**

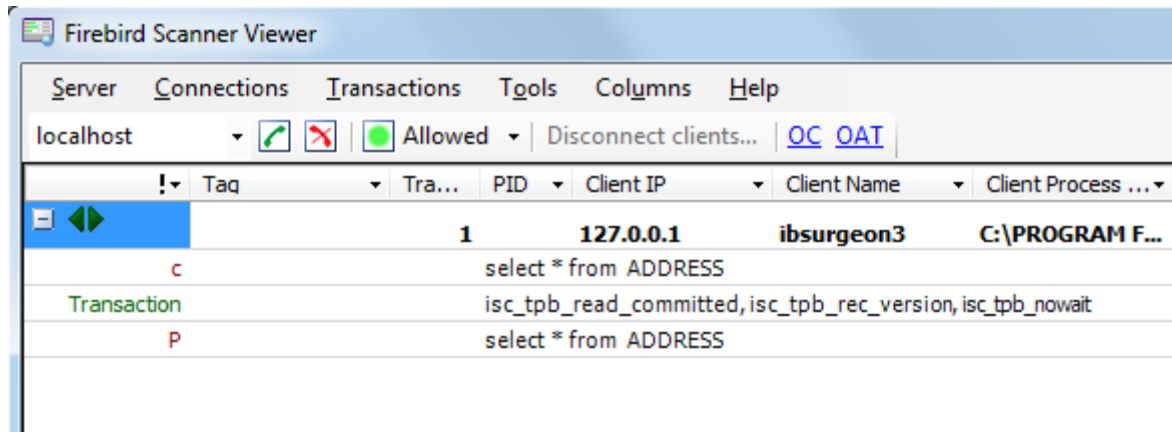
and, of course, using local connection string will always bypass FBScanner:

**gbak.exe -b -g -v -user SYSDBA -pass masterkey C:\TEMP\database.fdb
C:\temp\backup.gbk**

Real-Time Monitoring: FBScanner Viewer

To monitor connections, queries and transaction in real-time FBScanner includes special tool namely FBScanner Viewer.

FBScanner Viewer shows momentary snapshot of SQL traffic between Firebird and monitored client applications.



In the first column we can see type of record – connection, statements or transaction.

In the table below you can find description of all columns at main page of FBScanner Viewer (some columns are hidden by default, use menu Columns to turn them on/off):

| Column title | Column description |
|---------------------|---|
| ! (first column) | Indicates type of record in FBScanner Viewer – there are separate set of values for SQL statements, transactions and connections. They are described in the next table below. Sign “!” in the title of this column means active filter - click on the triangle at right side of sign “!” to adjust it. |
| Tag | Green/red background shows CPU Usage in % (red – Kernel, green – Firebird). Text is shows tags value (if it was specified in SQL query). Example how to set tag values: SELECT * FROM RDBDATABASE /*FBSCANNER\$CON_NAME=MyConnect; FBSCANNER\$TR_NAME=MyTransaction; FBSCANNER\$ST_NAME=SomeImportantQuery; */; Also in this column you will see execution of gbak and gfix tools |
| Transaction Count | Applicable for connection row. Number of active transactions in the connection is shown. It’s very useful to find applications with auto-commit and other ineffective transaction management issues. |
| PID | Process ID for Firebird. Only for Classic Architecture |
| Client IP | IP of connection |
| Client Name | DNS of connection (if possible to resolve) |
| Client Process Name | Starting from Firebird 2.1, fbclient.dll shows name of client application. For example, C:\Program Files\Firebird\Firebird_2_1\bin\isql.exe |
| Priority | Priority of Firebird instance (Classic only) |
| Database | Database name or its alias, as it appears in the connection string |
| User | Users name - for example, SYSDBA (it does not supported Trusted Authentication) |

| | |
|------------------|---|
| Role | Role of user |
| Start | For connection row – connection time, for transaction –start time of transaction, for statement – query start time. |
| Time | 'NOW' – Start; Time from the start moment |
| Last Activity | Time of last action for current connect/transaction/statement. |
| Inactive | 'NOW' – Last Activity; Period of inactivity |
| Latest Retaining | Time of the most recent "COMMIT RETAINING" or "ROLLBACK RETIANING" in the current transaction |
| Retaining | 'NOW' – Latest Retaining |
| Received | Bytes, received by client |
| Sent | Bytes, sent by client |
| CPU Time | Shows overall time consumed in connection/transaction/query. If there is more than 1 query in transactions, execution time of all queries will be summarized. The same rule is for connection time calculation. |
| Prepare Time | |
| Execute Time | |
| Fetch Count | Applicable only for statements. Number of rows, as it's reported by fbclient.dll |
| Protocol | Firebird protocol version for current session. |
| Version | Version of fbclient.dll/gds32.dll. Version detection is not 100% correct: <ul style="list-style-type: none"> - minor versions are considered as the same, - JayBird and .NET Provider are considered as the same, - InterBase 8.x = Interbase 9.x |

In the following table you can see details for the values appeared in the first column in FBScanner Viewer for SQL statements rows:

| Flag | Description |
|------|---|
| A | Allocated. Initial phase of SQL query life cycle |
| P | Prepared. Indicates that statement was prepared |
| E | Execute. Query is being executing at the moment |
| C | Closed statement. Execution is finished |
| D | Dropped statement. |
| F | Fetching is in progress |
| f | Fetching is in progress, but suspended at the moment (recordset is not fetched) |
| c | Closed cursor. All data was fetched. |

Tags

Tags allow assigning readable identifiers (names) to Connections, Queries and Transactions. You just need to add these commentaries:

```
SELECT COUNT(*) FROM RDB$DATABASE

/* FBSCANNER$CON_NAME=My_application;
FBSCANNER$TR_NAME=Read_only_transaction_N1;
FBSCANNER$ST_NAME=Customers_list_query; */
```

- FBSCANNER\$CON_NAME= sets the name of connection. After the first assignment this name will be kept during the whole connection life.

- FBSCANNER\$TR_NAME= sets the name of transaction. After the first assignment this name will be used during the whole life of transaction.
- FBSCANNER\$ST_NAME= sets the name of query.

Tags are showed in the first column in FBScanner Viewer grid, and it's possible to filter tags by their names.

Tags are useful to quickly answer the following frequent questions:

- What program has launched this query? (developers need to mark with FBSCANNER\$CON_NAME tag each database connection)
- What is the transaction for this query? (developers need to use FBSCANNER\$TR_NAME tag to mark transactions)
- What is this very long query? (developer can mark long queries with readable names like "Annual report").

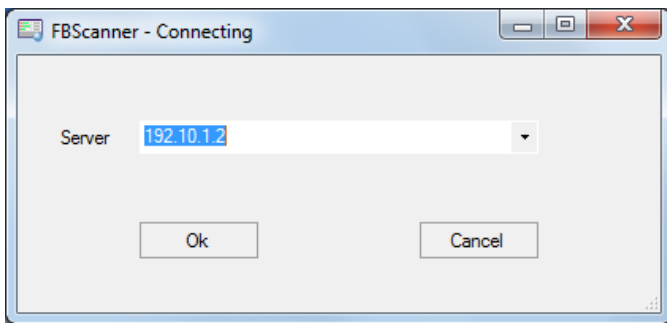
FBScanner Viewer Menu

FBScanner Viewer offers wide range of options to make debugging and optimization easier, which are accessible through its menu:

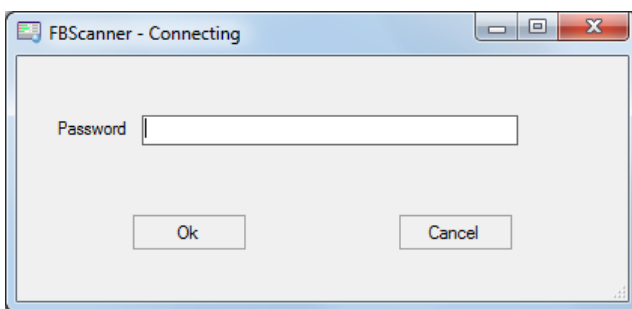
- **Server**
 - Connect To
 - Disconnect To
 - Recent Servers
 - Exit
- **Connections**
 - Disconnect
 - Disconnect Clients...
 - Kill Process
 - Latest Queries
 - Oldest Connection
 - Process Priority...
 - Ping Client
 - Ping All Clients
 - Extract Plans
- **Transactions**
 - OAT
- **Tools**
 - **View Style**
 - Database Administrator (connections only)
 - Database Developer (without transactions)
 - Database Developer (with transactions)
 - Language – English, Italian, Russian, Portuguese
 - Plugins
 - Options
- **Columns** – list of columns
- **Help**

Server

To connect to the FBScanner Service select **Service\Connect To**. The following dialog will appear:



After selecting the server FBScanner Viewer will ask for password. There are 2 passwords – for read-only access and for administrator (full) access. By default the password for read-only access is blank.



Tip. To setup passwords for FBScanner Viewer access you need to go to “FBScanner Configuration” – “Advanced Settings”.

Server\Disconnect disconnects FBScanner Viewer from FBScanner Service.

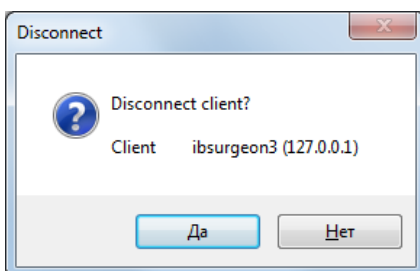
Server\Recent Servers shows list of most recent FBScanner Services where FBScanner Viewer connected to.

Exit closes FBScanner Viewer.

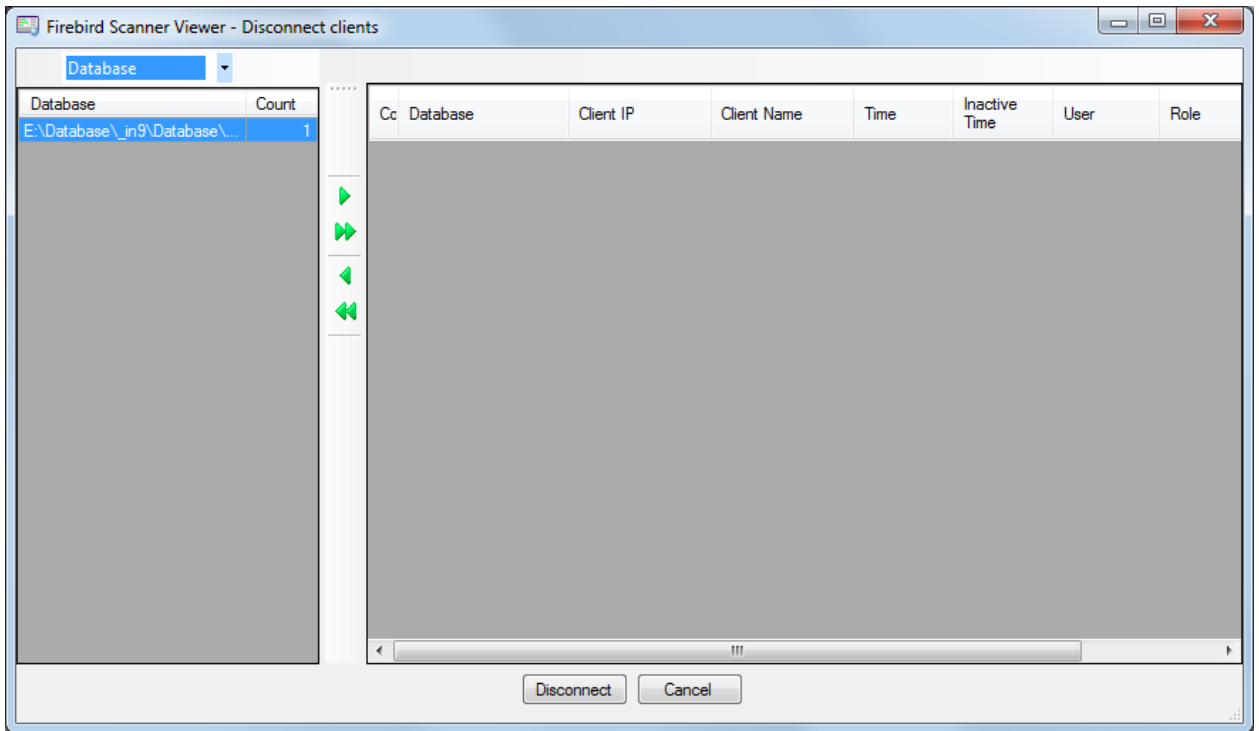
Connections

“Disconnect”, “Disconnect clients” and “Kill Process” menu options are available only when connected to FBScanner Service with administrative rights.

Disconnect will ask to close the current connection (highlighted in the main FBScanner Viewer grid):



“**Disconnect clients**” runs the following dialog:



In the right side there is a list of connections, represented by databases names, or clients, or user, according the filter above.

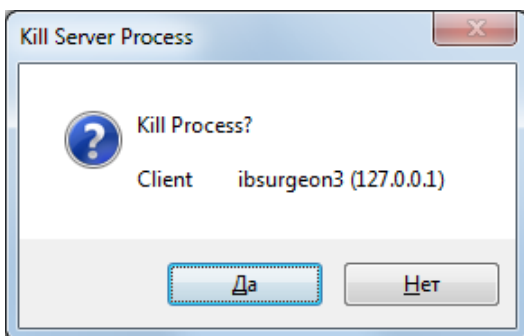
Using > and < buttons, administrator can select connections to be disconnected and then click “Disconnect” button.

Disconnect will be done by emulation of 10054 error – there will be appropriate record(s) in the firebird.log (interbase.log) and in FBScanner.log.

Kill

There are few cases when you need to kill Firebird process, and we do not recommend it.

“Kill process...” asks for explicit killing of Firebird process, and it works only at local FBScanner and Classic Architecture:



It will not work with SuperServer or SuperClassic architectures.

“Latest Queries” shows list of 20 most recent queries in the selected connection:

| Number | Query Start | Processor Time | Prepare Time | Execute Time | Fetch Count | SQLCODE | SQL |
|--------|-------------------|------------------|------------------|------------------|-------------|---------|--|
| 1 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 7 | 0 | select cast(1 as integer), cast(d.rdb\$character_set_name as |
| 2 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 3 | 0 | select RDB\$FIELD_NAME, RDB\$SYSTEM_FLAG |
| 3 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 417 | 0 | select RDB\$RELATION_NAME, RDB\$SYSTEM_FLAG, |
| 4 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 5 | 0 | select RDB\$RELATION_NAME, RDB\$OWNER_NAME |
| 5 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | select RDB\$PROCEDURE_NAME, RDB\$OWNER_NAME |
| 6 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 0 | 0 | select T.RDB\$TRIGGER_NAME, T.RDB\$TRIGGER_INACTIVE, |
| 7 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 2 | 0 | select RDB\$GENERATOR_NAME |
| 8 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 0 | 0 | select RDB\$EXCEPTION_NAME, RDB\$MESSAGE, |
| 9 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 9 | 0 | select RDB\$FUNCTION_NAME |
| 10 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | select RDB\$RELATION_NAME from RDB\$RELATIONS |
| 11 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 0 | 0 | select RDB\$ROLE_NAME, RDB\$OWNER_NAME FROM RDB\$ROLES |
| 12 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1780 | 0 | select RDB\$INDEX_NAME, RDB\$RELATION_NAME, |
| 13 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 51 | 0 | select RDB\$CHARACTER_SET_ID, RDB\$CHARACTER_SET_NAME, |
| 14 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 145 | 0 | select RDB\$COLLATION_ID, RDB\$CHARACTER_SET_ID, |
| 15 | 3/17/2011 1:36:14 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | select RDB\$RELATION_NAME from RDB\$RELATIONS |
| 16 | 3/17/2011 1:36:15 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | select RDB\$EXTERNAL_FILE from RDB\$RELATIONS |
| 17 | 3/17/2011 1:36:15 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | select RDB\$RELATION_TYPE from RDB\$RELATIONS |
| 18 | 3/17/2011 1:36:15 | 00:00:00.0000000 | 00:00:00.0000000 | 00:00:00.0000000 | 1 | 0 | SELECT DISTINCT RDB\$FIELD_NAME FROM |

It's useful for ad-hoc debugging, it works like "Rewind" button.

Tip. For full-fledged logging of SQL traffic enable SQL logging feature in FBScanner Service, and use FBScanner LogAnalyzer to look through the log.

"**Oldest Connection**" shows the oldest connection in the grid.

"**Process Priority**" is applicable only for local FBScanner installation with Classic architecture. It enables to set process priority for Classic instances.

"**Ping Client**" allows to check - is selected connection still alive?

"**Ping All Clients**" checks all connections in the same way.

"**Extract plans**" starts plan extracting for selected connect. Extracted plans are shown in the grid, and also stored in the SQL (or text) log. If logging is not enabled, nothing happens. To enable plan extraction for all connects, use appropriate setting in "FBScanner Configuration".

Transactions

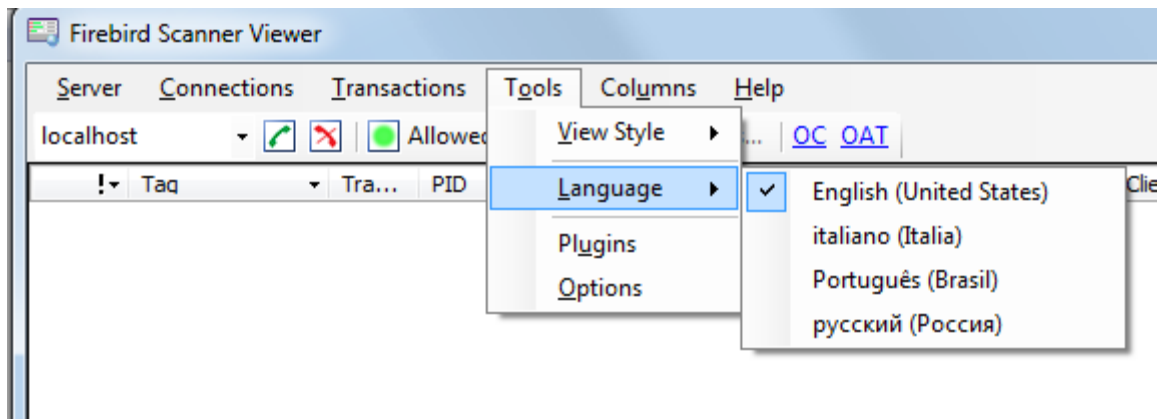
The single option **Transactions\OAT** will put selection in the grid to the oldest active transaction.

Tools

In menu "Tools" we can see several options. With "**View Style**" user can select the most suitable representation of grid data:

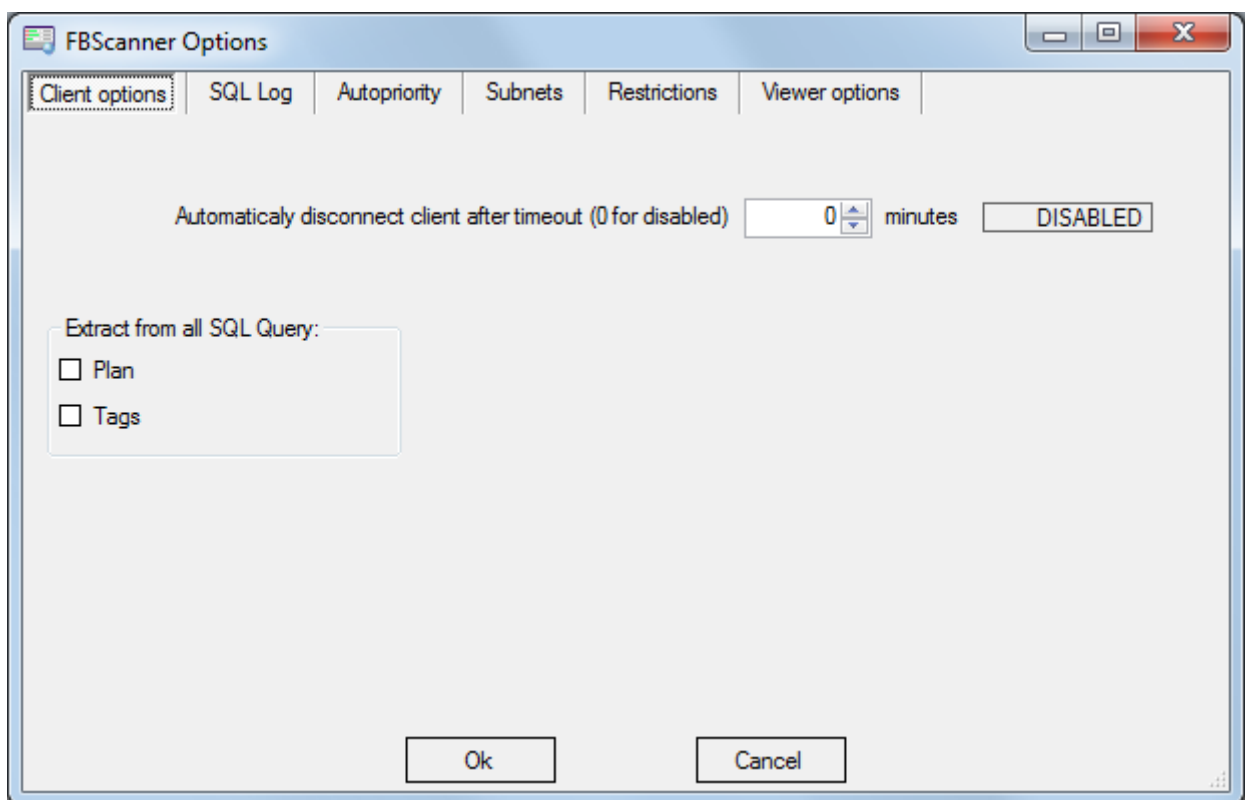
- Database Administrator (connections only)
- Database Developer (without transactions)
- Database Developer (with transactions)

FBScanner Viewer is localized in 4 languages. Use **Tools\Language** to switch between languages:



“**Plugins**” option enables plugins. For more information please contact support@ib-aid.com

“**Options**” is an another way to change some of FBScanner Service parameters.



Please consider appropriate session of this guide for details of FBScanner Service Configuration.

SQL log structure

FBScanner stores SQL traffic in the following table:

```
CREATE TABLE FBSCANNER$LOG
(
  ID          BIGINT NOT NULL,
  IDATTACHMENT  BIGINT,
  IDTRANSACTION  BIGINT,
  PID          INTEGER,
  ROW_TYPE     INTEGER NOT NULL,
  CLIENT_IP    VARCHAR(24),
```

```
CLIENT_NAME    VARCHAR(256),
CUSTOM_NAME    VARCHAR(256),
SUBNET_NAME    VARCHAR(256),
DB_FILENAME    VARCHAR(512),
DB_USER        VARCHAR(512),
DB_ROLE        VARCHAR(512),
START_TIME     TIMESTAMP DEFAULT 'NOW' NOT NULL,
END_TIME       TIMESTAMP,
LAST_ACTIVITY  TIMESTAMP DEFAULT 'NOW' NOT NULL,
LAST_RETAINING TIMESTAMP,
WORK_TIME      INTEGER DEFAULT 0 NOT NULL,
CPU_TIME_USER  INTEGER DEFAULT 0 NOT NULL,
CPU_TIME_PRIVILEGED INTEGER DEFAULT 0 NOT NULL,
FETCH_COUNT    INTEGER DEFAULT 0 NOT NULL,
RESULT         INTEGER,
SQL_TEXT       BLOB SUB_TYPE 1 SEGMENT SIZE 80,
SQL_TEXT2     BLOB SUB_TYPE 1 SEGMENT SIZE 80,
SQL_PLAN       BLOB SUB_TYPE 1 SEGMENT SIZE 80,
PREPARE_TIME   INTEGER DEFAULT 0 NOT NULL,
EXECUTE_TIME   INTEGER DEFAULT 0 NOT NULL
);
```

Logical structure

There are 3 levels of hierarchy in this table:

- ID – primary key
- IDATTACHMENT and IDTRANSACTION – foreign keys referenced to FBSCANNER\$LOG.ID
- ROW_TYPE - hierarchy level (0, 1, 2)

Level 1. Connection. ROW_TYPE = 0

| | |
|-------------|---|
| PID | Process ID (only for local FBScanner) |
| ROW_TYPE | 0 |
| CLIENT_IP | IP address of client |
| CLIENT_NAME | DNS name |
| CUSTOM_NAME | Connection tag (if assigned in query text) |
| SUBNET_NAME | Logical name of subnet See file FBScanner.subnets |
| DB_FILENAME | Database alias or full database path |
| DB_USER | User name |
| DB_ROLE | User role |
| START_TIME | Start of connection |
| END_TIME | End of connection |

Level 2. Transaction. ROW_TYPE = 1

| | |
|----------------|--|
| IDATTACHMENT | Connection ID |
| ROW_TYPE | 1 |
| CUSTOM_NAME | Transaction tag (if assigned) |
| START_TIME | Transaction start time |
| END_TIME | Transaction end time |
| LAST_RETAINING | Time of most recent commit retaining or rollback retaining |
| RESULT | 0 – transaction is active 1 – Commit 2 – Rollback |
| SQL_TEXT | Transaction flags |

Level 3. Query. ROW_TYPE = 2

| | |
|---------------------|---|
| IDATTACHMENT | Connection ID |
| IDTRANSACTION | Transaction ID |
| ROW_TYPE | 2 |
| CUSTOM_NAME | Query tag (if assigned) |
| START_TIME | Query start time |
| WORK_TIME | Time till the answer from server |
| CPU_TIME_USER | CPU Time (local only) |
| CPU_TIME_PRIVILEGED | CPU Kernel Time (local only) |
| FETCH_COUNT | Number of records, returned by query |
| RESULT | 0 – query executed successfully, otherwise this field contains SQLCODE of error |
| SQL_TEXT | Query text (with parameters) |
| SQL_TEXT2 | Original query text (NULL if equal to SQL_TEXT) |
| SQL_PLAN | Query execution plan (if “Extract plans” setting is enabled) |
| PREPARE_TIME | Prepare time |
| EXECUTE_TIME | Query execution time |

Indices in the log

Initially log database contains only primary key index. FBScanner Log Analyzer creates necessary indices at the first connect.

FBScanner Feature Matrix

| # | Feature | FBScanner mode | |
|------------|---|----------------|--------|
| | | Agent | Remote |
| | OPERATION SYSTEMS SUPPORT | | |
| | Windows | X | X |
| | Linux, Mac OS X, Free BSD | | X |
| | Firebird and InterBase versions supported | | |
| | Firebird 1.0, Yaffil 1.0 (including logging) | X | X |
| | Firebird 1.5 (including logging) | X | X |
| | Firebird 2.0 (including logging) | X | X |
| | Firebird 2.1 (including logging) | X | X |
| | Firebird 2.5 (including logging + SuperClassic support) | X | X |
| | InterBase 6.0-2009/XE (including logging) | X | X |
| 1 | Connections | | |
| 1.1 | <i>Information about established connections in the FBScanner Viewer:</i> | | |
| | Firebird/InterBase user login | X | X |
| | IP-address or computer name | X | X |
| | Connection time and time of the latest activity | X | X |
| | Priority of processes (only for Classic architecture) | X | |
| 1.2 | <i>Connection management (requires logging to FBScanner Viewer with Admin rights)</i> | | |
| | Safe disconnect of one or several connections using TCP/IP connection interruption (imitation of 10054 error) | X | X |
| | Changing of processes priority in Classic architecture (for example, to adjust priority of long running report or something like this. Using tags administrator can recognize connection where report is working – see below in “Tags”) | X | |
| | Automatic priority settings for Firebird with Classic architecture. In FBScanner configuration administrator can set up automatic correspondence: | X | |

| | | | |
|-------------|--|---|---|
| | <ul style="list-style-type: none"> Specified IP or subnet of IPs – set priority X Specified hostname – set priority X Specified database name – set priority X Specified user login name – set priority X | | |
| | Killing of Classic processes, not recommended to use, but sometimes it is helpful | X | |
| | Ability to restrict all connections (to perform some operations which require exclusive access) | X | X |
| | Filtering connections viewing using all connections parameters (except time information) | X | X |
| | White and black list of databases to connect | X | X |
| | White and black list of IP addresses (clients) | X | X |
| | Restriction of connections # - administrator can limit the number of connections | X | X |
| | Emulation of “Wrong login/password” error for denied connections | X | |
| | Detection of old/incorrect versions of fbclient.dll/gds32.dll | X | X |
| 1.3 | <i>Logging events related with connections</i> | X | X |
| | FBScanner logs unsuccessful login attempts in the FBScanner.log. For each unsuccessful login attempt FBScanner writes the following information: IP-address, login name, database and time of login attempt. | X | X |
| | <p>If connection was broken (10054 error), FBScanner determines and logs one of the 5 type of disconnects:</p> <ol style="list-style-type: none"> Client application was closed improperly (for instance, application was closed by Task Manager) Connection was closed by time-out (it's possible to set forced disconnect in FBScanner to close connect by time-out too) Server crashed (fbserver or fb_inet_server crashed) Server process (fbserver or fb_inet_server) was killed from the FBScanner Disconnect of connections from FBScanner Viewer <p>For all cases above FBScanner writes the IP-address of disconnected client(s) and the reason of disconnect. This is very useful feature to find and eliminate 10054 errors.</p> | X | X |
| 2. | Transactions | | |
| 2.1. | <i>Transactions are shown inside appropriate connections</i> | | |

| | | | |
|------------|--|---|---|
| | Transactions' flags | X | X |
| | Lifetime of transactions | X | X |
| | Using OAT button you can find the oldest active transaction in real-time and review related connection/queries | X | X |
| 3. | Queries (statements) | | |
| 3.1 | Information about queries (statements) | | |
| | Start time | X | X |
| | Query text | X | X |
| | Transaction of the query | X | X |
| | Status (prepare/execute/...) | X | X |
| | Filtering by statement status (by default Closed statements are hidden) | X | X |
| | Instant CPU load indicator | X | X |
| | If query PREPARE or execution caused error, FBScanner writes SQLCODE to the log (for example, primary key violation) | | |
| 3.2 | Additional operations with queries | | |
| | <p>Ad-hoc plan extraction for queries</p> <ul style="list-style-type: none"> • Can be performed for all connections (should be set ON in FBScanner configuration utility) • Can be turned ON/OFF for selected connection only in the FBScanner Viewer <p>In both cases plans will be logged to the overall log if logging is ON.</p> | X | X |
| 4. | Tags | | |
| | <p>Tags allow assigning readable identifiers (names) to Connections, Queries and Transactions. You just need to add these commentaries:</p> <pre>SELECT COUNT(*) FROM RDB\$DATABASE /* FBSCANNER\$CON_NAME=My_application; FBSCANNER\$TR_NAME=Read_only_transaction_N1; FBSCANNER\$ST_NAME=Customers_list_query; */</pre> | X | X |
| | FBSCANNER\$CON_NAME= sets the name of connection. After the first assignment this name will be kept during the whole connection life. | X | X |

| | | | |
|-----------|---|---|---|
| | FBSCANNER\$TR_NAME= sets the name of transaction. After the first assignment this name will be used during the whole life of transaction | X | X |
| | FBSCANNER\$ST_NAME= sets the name of query. | | |
| | Tags are showed in special column in FBScanner Viewer | X | X |
| | It's possible to filter tags by their names | X | X |
| | Tags are useful to quickly answer the following frequent questions: <ul style="list-style-type: none"> • What program has launched this query? (developers need to mark with FBSCANNER\$CON_NAME tag each database connection) • What is the transaction for this query? (developers need to use FBSCANNER\$TR_NAME tag to mark transactions) • What is this very long query? (developer can mark long queries with readable names like "Annual report") | X | X |
| 5. | Logging | | |
| | Logging allows intercepting all queries and writing them to the external Firebird database. FYI, logging cannot be replaced with Firebird 2.1 or InterBase system tables, because they provide only snapshots of programs. | X | X |
| | Connections, queries and transactions are logged | X | X |
| | All executed queries are logged (only prepared quires skipped) | X | X |
| | Queries are stored with information about their connection and transaction | X | X |
| | All transactions are logged, even rolled back. Transaction log record has column RESULT which shows was transaction committed or rolled back. | X | X |
| | If plan extraction is on, queries plans are logged too | X | X |
| | Automatic creation of database for logging | X | X |
| | Automatic creation of tables to logging in any Firebird database | X | X |

Support

If you have any issues with FBScanner, please feel free to ask any questions: support@ib-aid.com