HQbird User Guide

(v. 2018R4)

www.ib-aid.com

v. 2019022519

Contents

About IBSurgeon

IBSurgeon (www.ib-aid.com) was founded in 2002 with the idea to provide InterBase and Firebird developers and administrator with unique services and tools focused on databases safety, performance and availability. In Russia IBSurgeon is mostly known as iBase.ru, famous by its Russian InterBase and Firebird portal www.ibase.ru.

IBSurgeon is a Platinum sponsor of Firebird Foundation and, as a member of Technical Task Group, has strong relationship with Firebird Project, with direct representatives in Firebird-Admins and in Firebird Foundation Committee.

Today, IBSurgeon serves more than 2000 companies worldwide with emergency, optimization and maintenance tools and various services. Our clients are top Russian and US banks, Russian government institutions, worldwide insurance and finance organizations, medical institutions and, of course, a lot of ISVs and microISVs – all who develop or use applications based on Firebird and/or InterBase.

1. Overview of HQbird

1.1. What is HQbird

HQbird is an advanced distribution of Firebird for enterprises, with the following list of features:

- high availability and replication,
- pool of execute on external connections,
- cached prepared statements,
- optimized configurations,
- backups' automation, including cloud backups,
- database health monitoring,
- automatic performance and transactions reports,
- advanced transactions and queries monitoring (TraceAPI, MON$ and FBScanner),
- database structure analysis,
- recovery toolset and database development GUI.

Also HQbird includes a performance self-test to measure Firebird performance on the specific hardware and OS configuration.

HQBird contains 2 parts: server-side and administration. The server-side part has versions for Windows and Linux, and the administration part works on Windows only.

There are 3 editions of HQbird: Standard, Professional and Enterprise.

1.2. Feature matrix

Below there is the feature matrix for HQbird editions.

Features	Editions		
	Standard	Professional	Enterprise
Optimized configurations	X	X	X
Backups	X	X	X
Automatic Performance Reports	X	X	X
SQL queries tracking	X	X	X
Transactions tracking	X	X	X
Database structure analysis	X	X	X
Performance self-test suite	X	X	X
SQL development, design&debugging		X	X
Recovery		X	X
Replication and high availability			X
Pool of Execute On External statements			X
Cached prepared statements			X

It is possible to single out components in the HQbird modules that are responsible for certain operations (such as backup, monitoring, database repair).

The table below shows how features are distributed among the HQbird modules:

Features	Modules	Description
Backup (automated verified and incremental backup)	FBDataGuard	FBDataGuard runs on the server side and performs all kinds of backup
Optimized Configurations (balanced, read-intensive and write-intensive)	FBDataGuard, a collection of optimized configurations	The optimized configuration file can be customized on the basis of recommendations from FBDataGuard
Performance Test Suite (hardware score)	Performance Test Suite	The test measures the hardware performance
Monitoring SQL Queries (MON$, TraceAPI and FBScanner)	Performance Monitor, MON$Logger, FBScanner	Three different monitoring methods are used in different scenarios
Health Monitoring (online validation, database health check, log analysis)	FBDataGuard	Everything is carried out on the server. FBDataGuard sends notifications by e-mail.

Transaction Tracking (dynamic analysis of transaction markers)	FBDataGuard, Transaction Monitor, MON$Logger	FBDataGuard tracks problems with transactions, Transaction Monitor and MON$Logger show the dynamics of changes and the current status of active transactions.
Database Structure Analysis (table and index sizes, fragmentation, versioning, etc.)	Database Analyst	Database Analyst analyses the database structure in detail and shows warnings and recommendations.
SQL Development & Debugging (a GUI tool for developing databases and queries)	SQL Studio	SQL Studio is a powerful tool for developing and debugging database objects and SQL queries.
Recovery (database recovery, backup recovery, record undeleting)	FirstAID, FBDataGuard, IBBackupSurgeon, IBUndelete	FirstAID repairs databases when they get corrupted, FBDataGuard stores important metadata thus increasing the chances of successful repairs, IBBackupSurgeon exports data from corrupted backup copies. IBUndelete can undo records deletion.
High Availability (replication)	The HQbird Enterprise edition includes replication and high availability tools.	
Performance improvements (pool of Execute On External and Cached prepared)	HQbird Enterprise includes performance improvements.	

2. Installation of HQbird

HQbird contains 2 parts: ServerSide and Admin. Let's consider how to install them.

2.1. Installing server part of HQbird on Windows

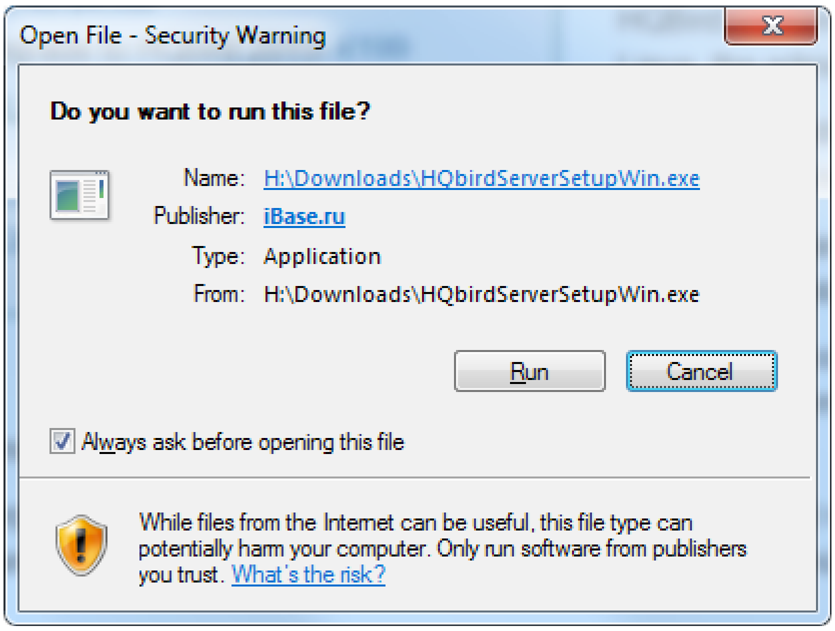
HQbird ServerSide 2018 includes Firebird 2.5 and 3.0 with replication and trace plugins as part of its installer, so Firebird will be installed as part of HQbird. It is mandatory to install Firebird bundled with HQbird ServerSide installer, if you plan to use replication (it also requires HQbird Enterprise license, full or trial).

Optionally you can choose HQBird Standard, which will not install Firebird, so you can use previously installed Firebird – in this case, make sure that installed version is compatible (1.5, 2.0, 2.1, 2.5.2, 2.5.5, 2.5.6, 2.5.7, 2.5.8, 2.5.9, 3.0.x).

Please note, that only Firebird 2.5 and 3.0 are fully supported in HQbird, the old Firebird versions are supported partially. We offer the comprehensive Firebird migration service with guaranteed and fast result to migrate Firebird to the latest version.

Installing HQbird ServerSide for Windows

Download HQbird from http://ib-aid.com/en/hqbird/. The distribution package of the server side is the same for the 32-bit and 64-bit versions of the Firebird engine.
Make sure that HQbird installer is signed with our certificate («iBase.ru») and run it:

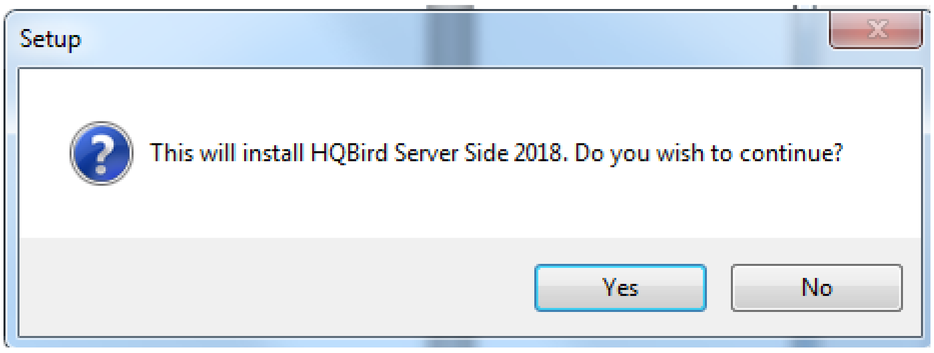

Open File - Security Warning

Do you want to run this file?

Name: H:\Downloads\HQbirdServerSetupWin.exe
Publisher: iBase.ru
Type: Application
From: H:\Downloads\HQbirdServerSetupWin.exe

☑ Always ask before opening this file

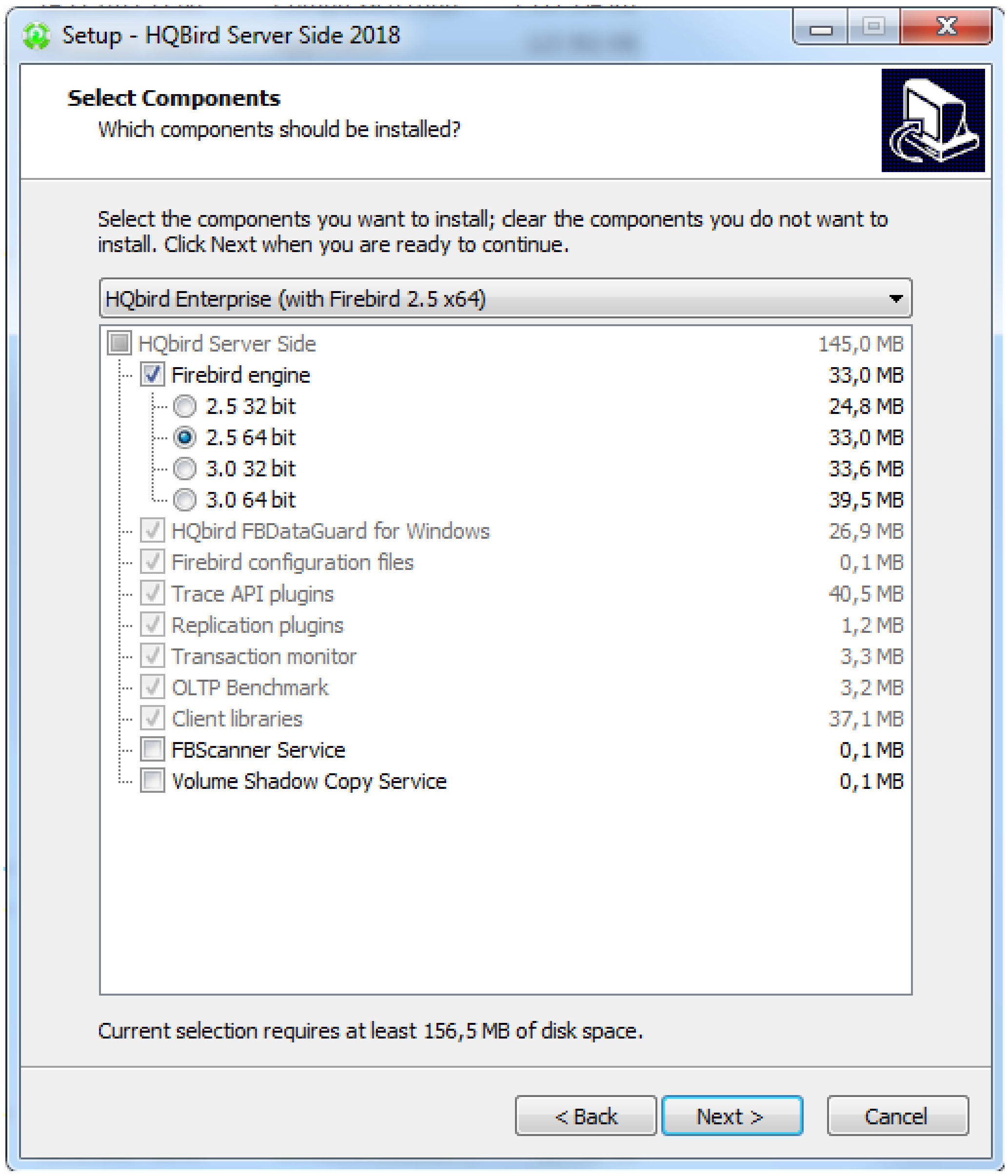
While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?



This will install HQBird Server Side 2018. Do you wish to continue?

The HQbird Server-Side Tools installation wizard will be launched after that and it will take you through several steps, such as agreeing to the license agreement and selecting the installation folder.

After that you will see the installation step where you can select components to be installed:



Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

HQbird Enterprise (with Firebird 2.5 x64)

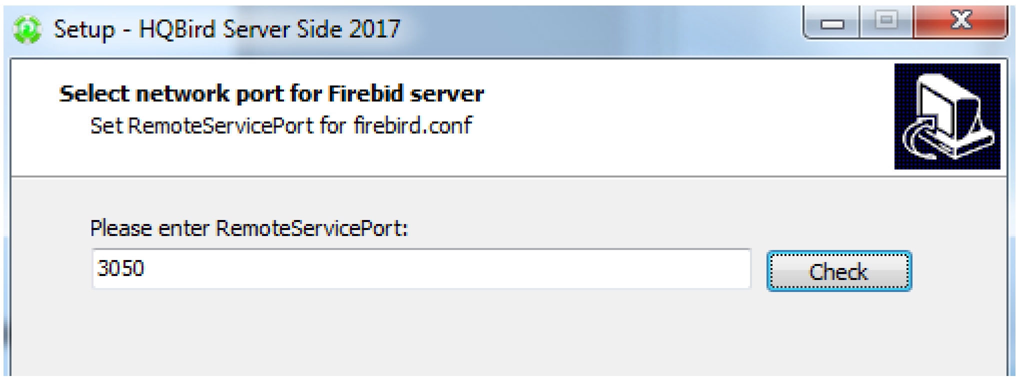
☐ HQbird Server Side	145,0 MB
✓ Firebird engine	33,0 MB
○ 2.5 32 bit	24,8 MB
● 2.5 64 bit	33,0 MB
○ 3.0 32 bit	33,6 MB
○ 3.0 64 bit	39,5 MB
✓ HQbird FBDataGuard for Windows	26,9 MB
✓ Firebird configuration files	0,1 MB
✓ Trace API plugins	40,5 MB
✓ Replication plugins	1,2 MB
✓ Transaction monitor	3,3 MB
✓ OLTP Benchmark	3,2 MB
✓ Client libraries	37,1 MB
☐ FBScanner Service	0,1 MB
☐ Volume Shadow Copy Service	0,1 MB

Current selection requires at least 156,5 MB of disk space.

Select components from HQbird Server-Side to be installed

We recommend that you install all HQbird components and Firebird, to avoid further setup steps. Also, they occupy very little space. All components are installed in the inactive mode and do not affect the operation of the Firebird server until they are configured or used.
If you select to install Firebird with HQbird (recommended), it will install Firebird in the subfolder of HQbird installation, by default C:\HQBird\Firebird25 (C:\HQBird\Firebird30).

The installation wizard will ask to specify the port for Firebird engine installed with HQbird:

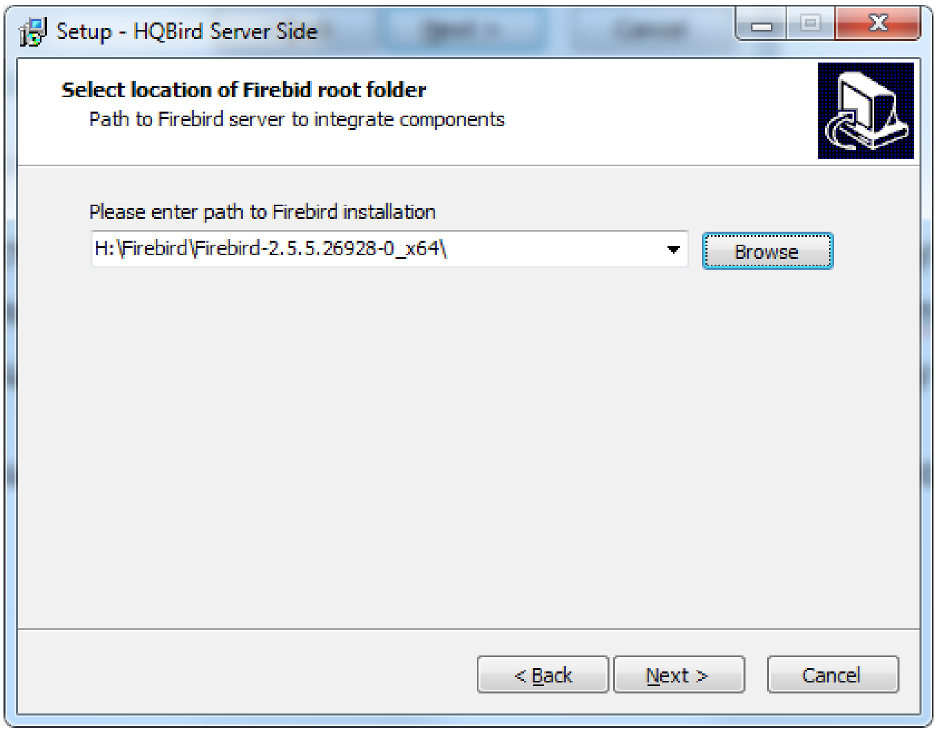

Setup - HQBird Server Side 2017

Select network port for Firebid server
Set RemoteServicePort for firebird.conf

Please enter RemoteServicePort:
3050

By default, the port is 3050. If the port will be occupied by another running Firebird, the installation wizard will warn you and make to choose another port. Or, you can stop and uninstall another Firebird service.

If you don't opt to install Firebird with HQbird, the installation wizard will ask you to specify the folder where Firebird is installed:

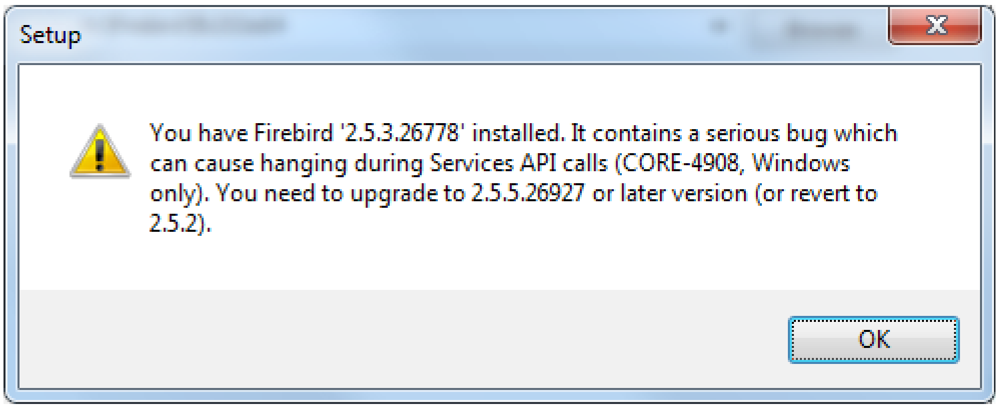

Setup - HQBird Server Side

Select location of Firebid root folder
Path to Firebird server to integrate components

Please enter path to Firebird installation
H:\Firebird\Firebird-2.5.5.26928-0_x64\

Confirm the location of the current Firebird instance

Attention! In this step, the installation wizard checks the availability and compatibility of the installed Firebird version with HQbird. If the specified folder does not contain a correctly installed Firebird version, you will see the following warning:

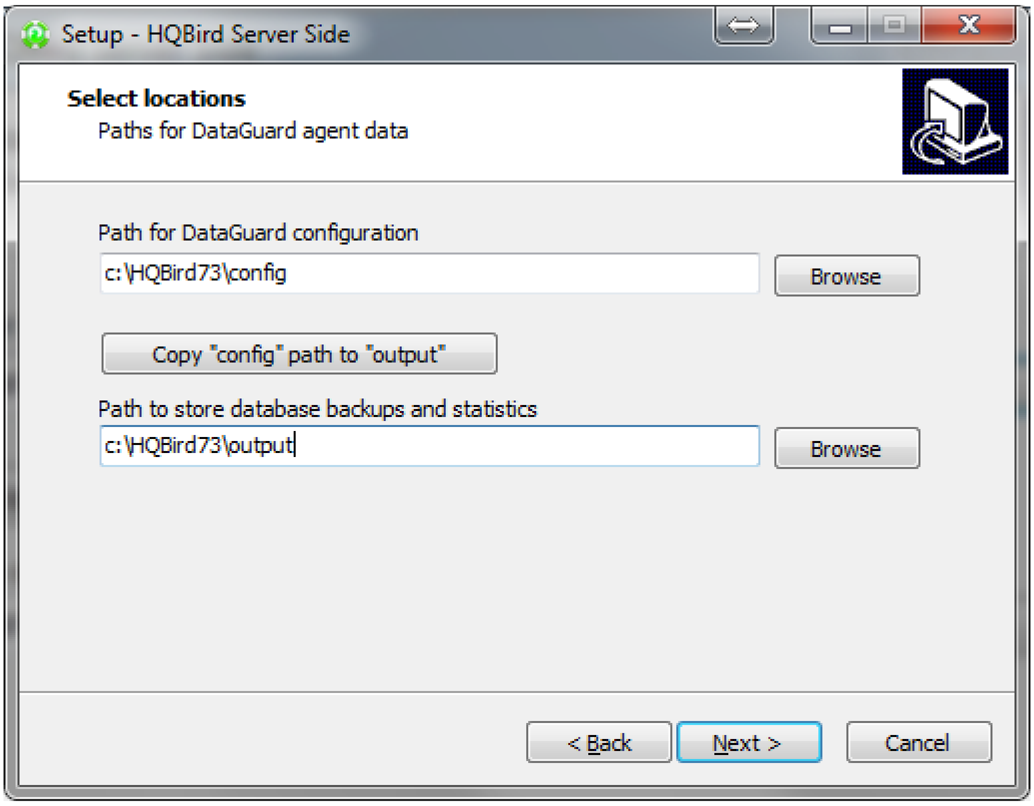


You have Firebird '2.5.3.26778' installed. It contains a serious bug which can cause hanging during Services API calls (CORE-4908, Windows only). You need to upgrade to 2.5.5.26927 or later version (or revert to 2.5.2).

This warning from the installation wizard prompts you to select the correct Firebird folder

You should use Firebird version 2.5.5 or higher for HQbird Standard to be installed (see *How to Update Firebird for Windows*), or use Firebird bundled with HQbird installer.

After that you should select folders for storing configuration files, backup copies, statistics and HQbird log files:



Select locations
Paths for DataGuard agent data

Path for DataGuard configuration
c:\HQBird73\config

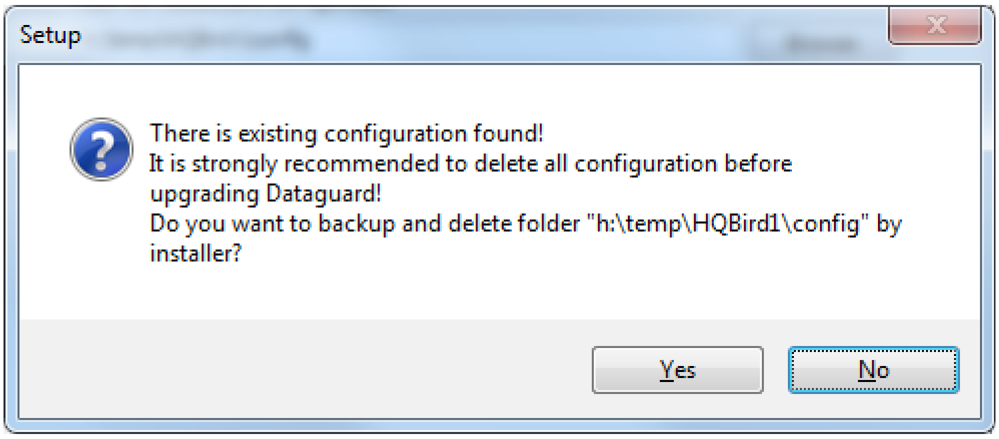
Path to store database backups and statistics
c:\HQBird73\output

Select folders for HQbird configuration and log files

By default, the installation wizard offers to create folders for configuration and log files in C:\HQbirdData.

Usually, we recommend that you select another location for these purposes on a disk with a larger amount of free space.

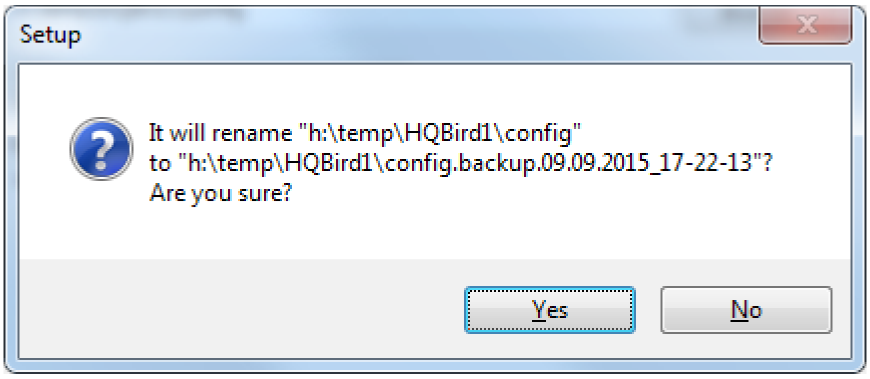
If configuration files already exist in the selected location, the installation wizard will display the corresponding warning:



There is existing configuration found!
It is strongly recommended to delete all configuration before upgrading Dataguard!
Do you want to backup and delete folder "h:\temp\HQBird1\config" by installer?

Warning about existing configuration files

If you decide to rename the existing configuration folder, the installation wizard will display one more warning:

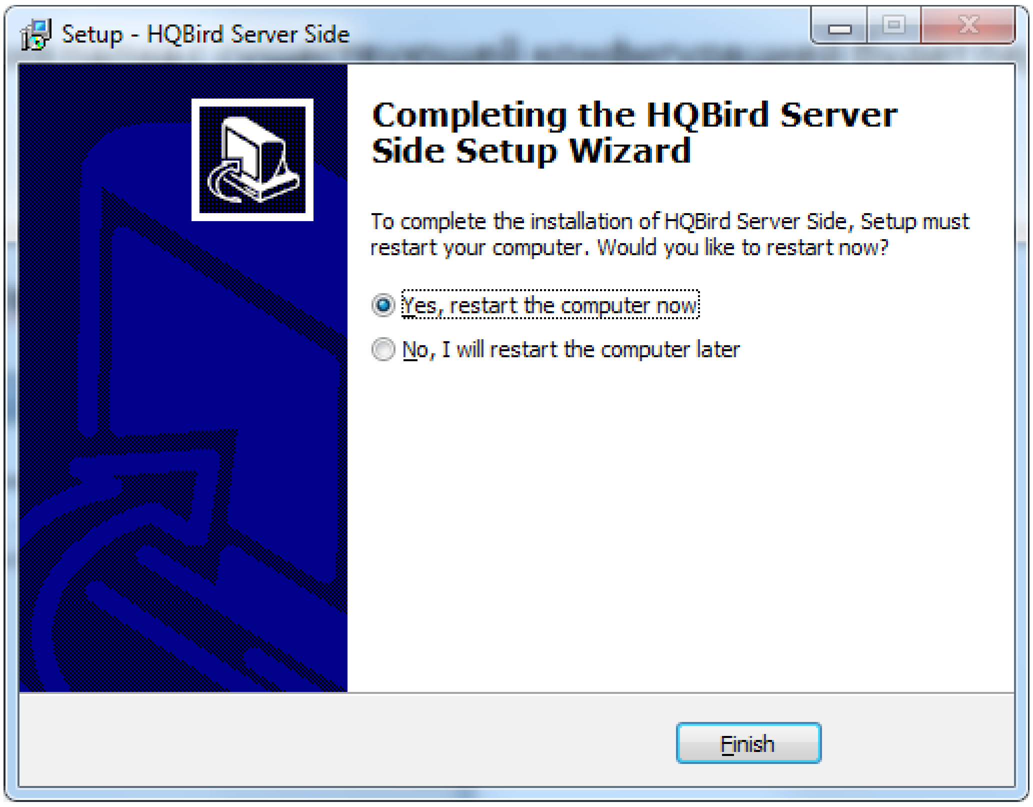


It will rename "h:\temp\HQBird1\config"
to "h:\temp\HQBird1\config.backup.09.09.2015_17-22-13"?
Are you sure?

Rename confirmation

After you confirm it, the folder with the existing configuration files will be renamed and the installation will continue.

You may have to restart the computer after the installation:



Completing the HQBird Server Side Setup Wizard

To complete the installation of HQBird Server Side, Setup must restart your computer. Would you like to restart now?

[✓ Yes, restart the computer now]

No, I will restart the computer later

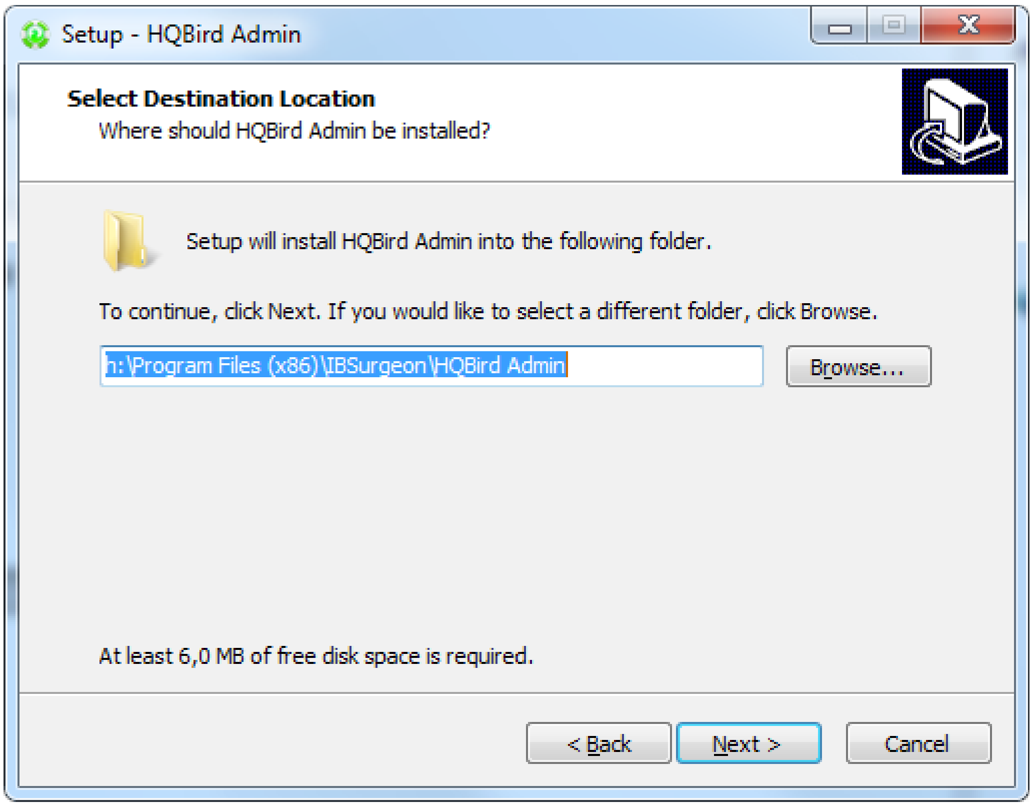
After that you have to activate HQbird (see *How to Activate HQbird*) and proceed to configure the HQbird components.

2.2 Installing HQbird Administrator (Windows)

To install HQBird Administrator, download the distribution package from http://ib-aid.com/en/hqbird/, or from your account at http://deploy.ib-aid.com

The name of HQbird Administrator package is HQbirdAdminNNNN.exe (it is in the zip archive)

Run the installation wizard and follow the standard installation steps: select the installation folder:


Setup - HQBird Admin

Select Destination Location
Where should HQBird Admin be installed?

Setup will install HQBird Admin into the following folder.

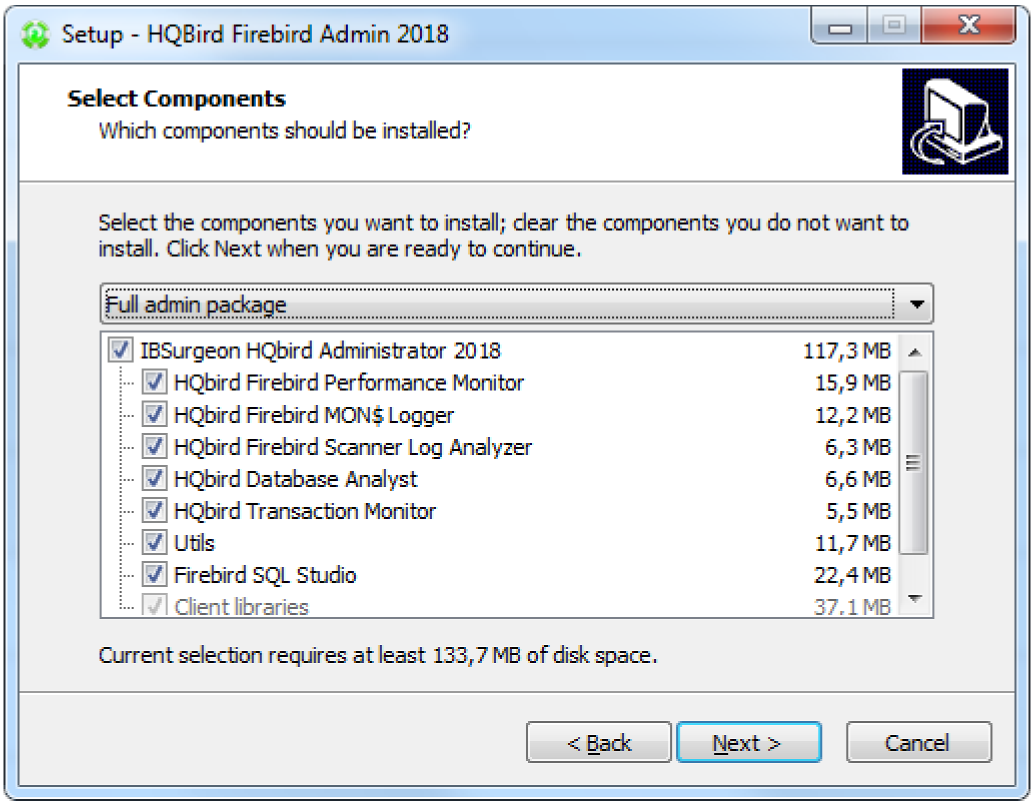
To continue, click Next. If you would like to select a different folder, click Browse.

h:\Program Files (x86)\IBSurgeon\HQBird Admin

At least 6,0 MB of free disk space is required.

Select where to install HQbird Administrator

Select tools to install after that. We recommend that you install all tools.


Setup - HQBird Firebird Admin 2018

Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full admin package

✓ IBSurgeon HQbird Administrator 2018	117,3 MB
✓ HQbird Firebird Performance Monitor	15,9 MB
✓ HQbird Firebird MON$ Logger	12,2 MB
✓ HQbird Firebird Scanner Log Analyzer	6,3 MB
✓ HQbird Database Analyst	6,6 MB
✓ HQbird Transaction Monitor	5,5 MB
✓ Utils	11,7 MB
✓ Firebird SQL Studio	22,4 MB
✓ Client libraries	37,1 MB

Current selection requires at least 133,7 MB of disk space.

Select tools to install

Follow the instructions after that. After the installation is over, you will be offered to launch the activation wizard.

How to install Firebird at Windows

The easiest way is to install Firebird bundled with HQbird – just choose the desired version during the installation.

> Please note – to enable replication features in HQbird Enterprise you need to install Firebird bundled with HQbird ServerSide.

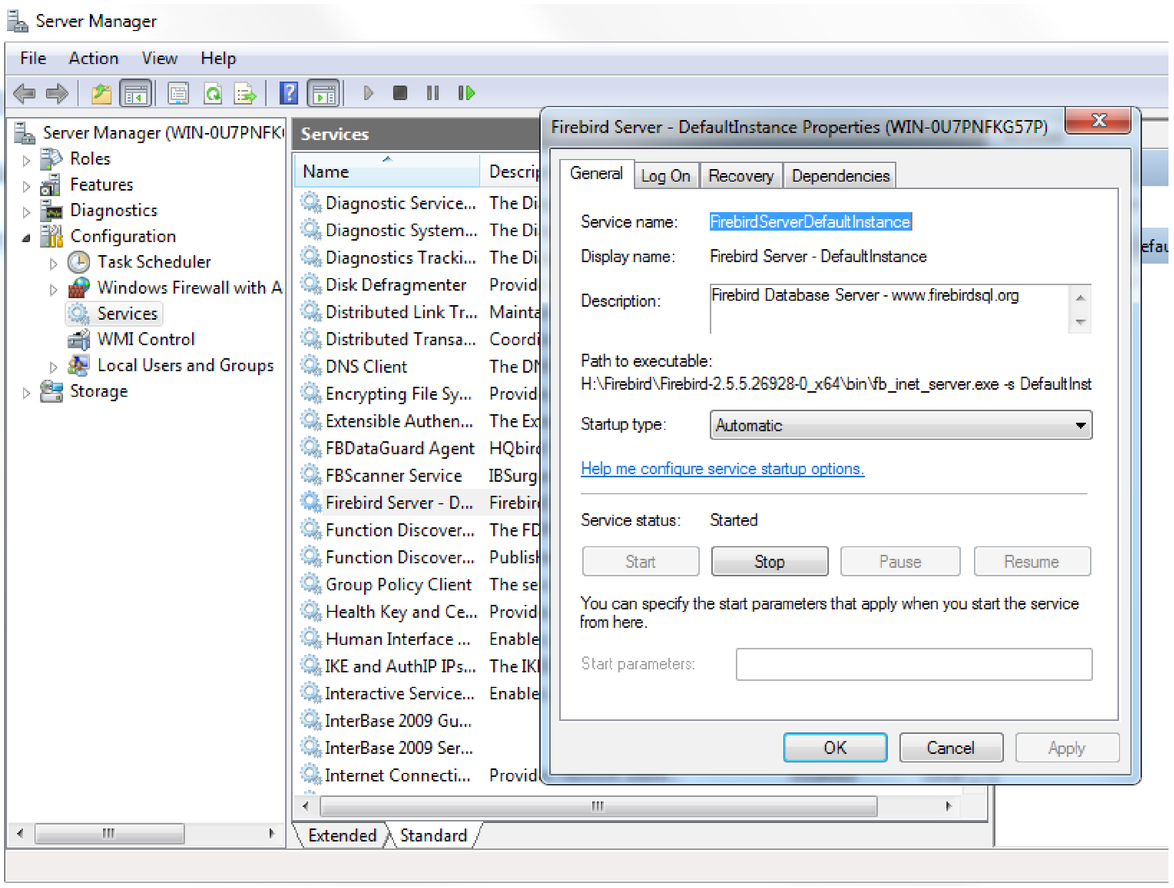
To install Firebird separately, download the Firebird zip archive from www.firebirdsql.org

Unpack the archive file to a suitable location (for instance, C:\Firebird25), after that copy the optimized configuration file **firebird.conf** (see *Optimized Configurations* below) to this folder.

Then, go to the Bin folder and then use the **Run As Administrator** option to run the batch file with the architecture you need.

- For Firebird 2.5 – run **install-superclassic.bat.**
- For Firebird 3.0 – set parameter ServerMode=Super and run **install_service.bat.**

Of course, you can choose the SuperServer for 2.5 or Classic architecture for 3.0 if you know what you need.
As a result of running the command file, Firebird of the selected architecture will be installed and run as a service. You can make sure the Firebird service is installed and running in the **Services** snap-in (services.msc):



Diagnostic Service... The Di
Diagnostic System... The Di
Diagnostics Tracki... The Di
Disk Defragmenter Provid
Distributed Link Tr... Mainta
Distributed Transa... Coordi
DNS Client The DN
Encrypting File Sy... Provid
Extensible Authen... The Ex
FBDataGuard Agent HQbird
FBScanner Service IBSurg
Firebird Server - D... Firebir
Function Discover... The FD
Function Discover... Publish
Group Policy Client The se
Health Key and Ce... Provid
Human Interface ... Enable
IKE and AuthIP IPs... The IKI
Interactive Service... Enable
InterBase 2009 Gu...
InterBase 2009 Ser...
Internet Connecti... Provid

Firebird Server - DefaultInstance Properties (WIN-0U7PNFKG57P)

Service name: FirebirdServerDefaultInstance

Display name: Firebird Server - DefaultInstance

Description: Firebird Database Server - www.firebirdsql.org

Path to executable:
H:\Firebird\Firebird-2.5.5.26928-0_x64\bin\fb_inet_server.exe -s DefaultInst

Startup type: Automatic

Help me configure service startup options.

Service status: Started

You can specify the start parameters that apply when you start the service from here.

Start parameters:

Firebird Service

In this example, Firebird is installed in the folder H:\Firebird\Firebird-2.5.5.26928-0_x64 and running as a service with the SuperClassic architecture.

2.3. Installing server part of HQbird on Linux

To install HQbird Server Side on Linux, you need to download HQbird ServerSide for Linux with integrated Firebird from this location:

https://ib-aid.com/en/hqbird-installation/

This archive contains 2files:

install_fb25_hqbird_server_2018.sh and *install_fb30_hqbird_server_2018.sh*.

You must be root or sudoer to install HQbird on Linux!
General prerequisites*:* install **java version 1.8** before installing HQbird!

Installation of HQbird with Firebird 2.5 on Linux

1. Uninstall all previously installed Firebird versions before running this installer. Make sure you don't have Firebird installed from repositories!
2. Apply execution rights to the installation package:
   chmod +x install_fb25_hqbird_server_2018.sh
3. Run installation script install_fb25_hqbird_server_2018.sh. It will install Firebird into **/opt/firebird** and HQbird into **/opt/hqbird**

4. By default, Firebird 2.5 is installed as Classic. We recommend to install it as SuperClassic – for this run script **/opt/firebird/bin/changeMultiConnectMode.sh** and choose **thread**

Next steps:

1. Please note that Firebird 2.5 will be installed with SYSDBA/masterkey
2. You can stop/start Firebird 2.5 with command **service firebird stop/start**. Check is it running with command **ps aux | grep firebird**
3. You can stop/start HQbird with command **service hqbird stop/start**. Check is it running with command **ps aux | grep dataguard**
4. Run browser, and log in to HQbird FBDataGuard **http://serverurl:8082**, with user/password = **admin/strong password**
5. Choose "I have HQbird Enterprise" and register HQbird with the email and password you have received from IBSurgeon Deploy Center.
6. If necessary, follow steps to setup replication – see the appropriate chapter of this Guide.

Installation of HQbird with Firebird 3.0 on Linux

Prerequisites: make sure you have **libtommath** and **ICU** installed (there will be an appropriate error message if they are not installed).

1. Uninstall all previously installed Firebird versions before running this installer
2. Apply execution rights to the installation package:
   chmod +x install_fb30_hqbird_server_2018.sh
3. Run installation script install_fb30_hqbird_server_2018.sh. It will install Firebird into **/opt/firebird** and HQbird into **/opt/hqbird**
4. By default, Firebird 3.0 is installed as SuperServer. Keep it.
5. Firebird 3.0 will be installed with SYSDBA/masterkey

Next steps:

1. You can stop/start Firebird 3.0 with command **service firebird-superserver stop/start.** Check is it running with command **ps aux | grep firebird**
2. You can stop/start HQbird with command **service hqbird stop/start**. Check is it running with command **ps aux | grep dataguard**
3. Run browser, and log in to HQbird FBDataGuard **http://serverurl:8082**, with user/password = **admin/strong password**
4. Choose "I have HQbird Enterprise" and register HQbird with the email and password you have received from IBSurgeon Deploy Center.
5. If necessary, follow steps to setup replication –see the appropriate chapter of this Guide.

Firewall settings

Firebird is installed on port 3050, HQbird web interface is listening on port 8082, and licensing interface is listening on 8765. These ports can be changed in /opt/firebird/firebird.conf (RemoteServicePort), /opt/hqbird/conf/network.properties (server.port) and /opt/hqbird/conf/license.properties (serverlicense.port).
Make sure to allow these ports in your firewall.

Attention! Make sure that there is only the one copy of HQbird is running! If there are 2 copies, stop them (`service hqbird stop` for the first and `kill <process-number>` for the second instances) and start it again.

2.4 Upgrade existing HQbird version

HQbird installer on Windows (from v 2018R2) and on Linux (from v 2018R3) supports automatic upgrade of the configuration of already installed HQbird version 2017R2 and later.

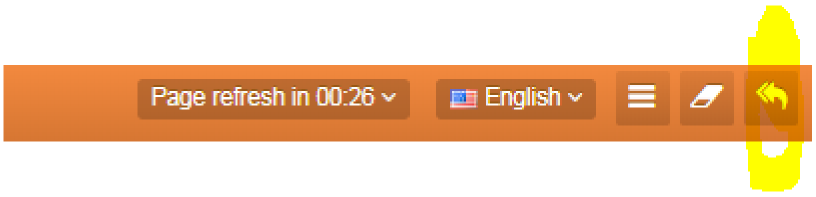
If HQbird installer will notice the previous version of HQbird, it will ask you to confirm the upgrade, and in case of the positive answer, it will stop Firebird, HQbird and upgrade their files.

The configuration will be retained – it means that firebird.conf, aliases.conf, securityX.fdb, and HQbird configuration files will not be deleted (HQbird configuration files upgraded to the new configuration version).

The upgrade does not change the Windows service settings for Firebird and HQbird – it means that if you have changed Run As properties of the service, they will be retained.

Note: After upgrade on Linux Firebird and HQbird must be started manually!

Important! After upgrading HQbird, open the web-console and choose in the right upper corner: «Refresh HQbird web-console». It is necessary to clean the cache of JavaScript part of the application.



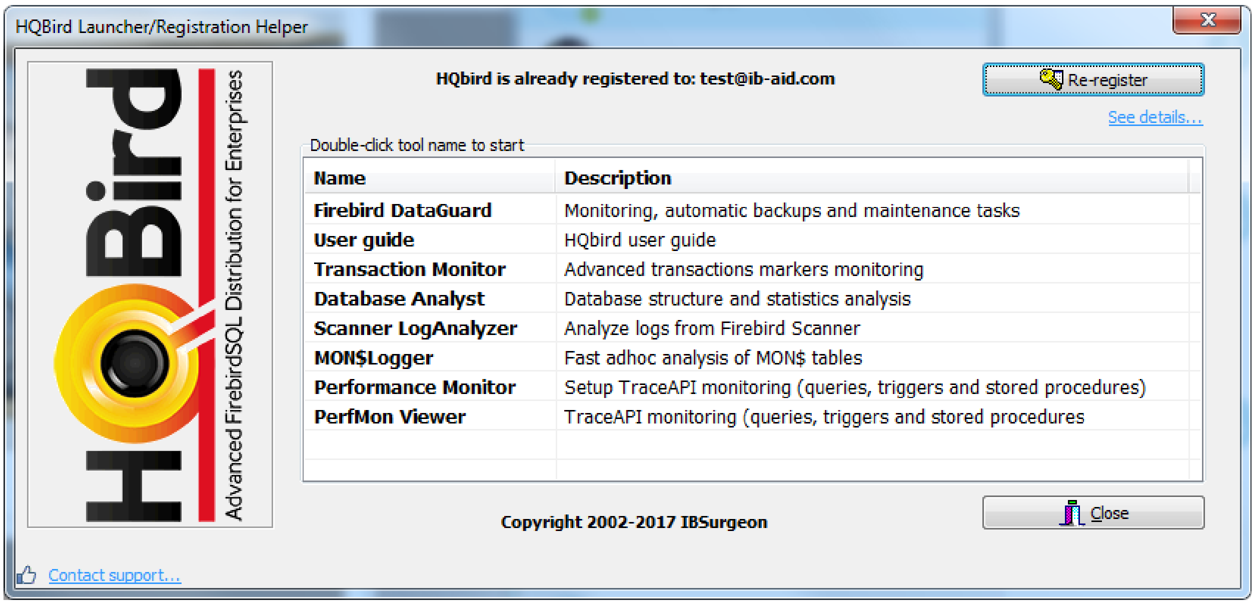
2.5. Registration of HQbird

How to Activate HQbird

To activate HQbird, you can either use a separate utility included in the server and administrator packages for Windows, or use the registration mechanism embedded into the HQBird Firebird DataGuard web interface (for Windows and Linux), or run any tool from the administrator software and use the built-in activation wizard.

The activation wizard looks and works the same in the tools and in the activation tool. It is enough to perform activation once on any computer that can connect to the server where HQbird ServerSide is installed.

You can launch the registration utility from the **Start** menu (IBSurgeon\HQbird Firebird Admin\HQbird):

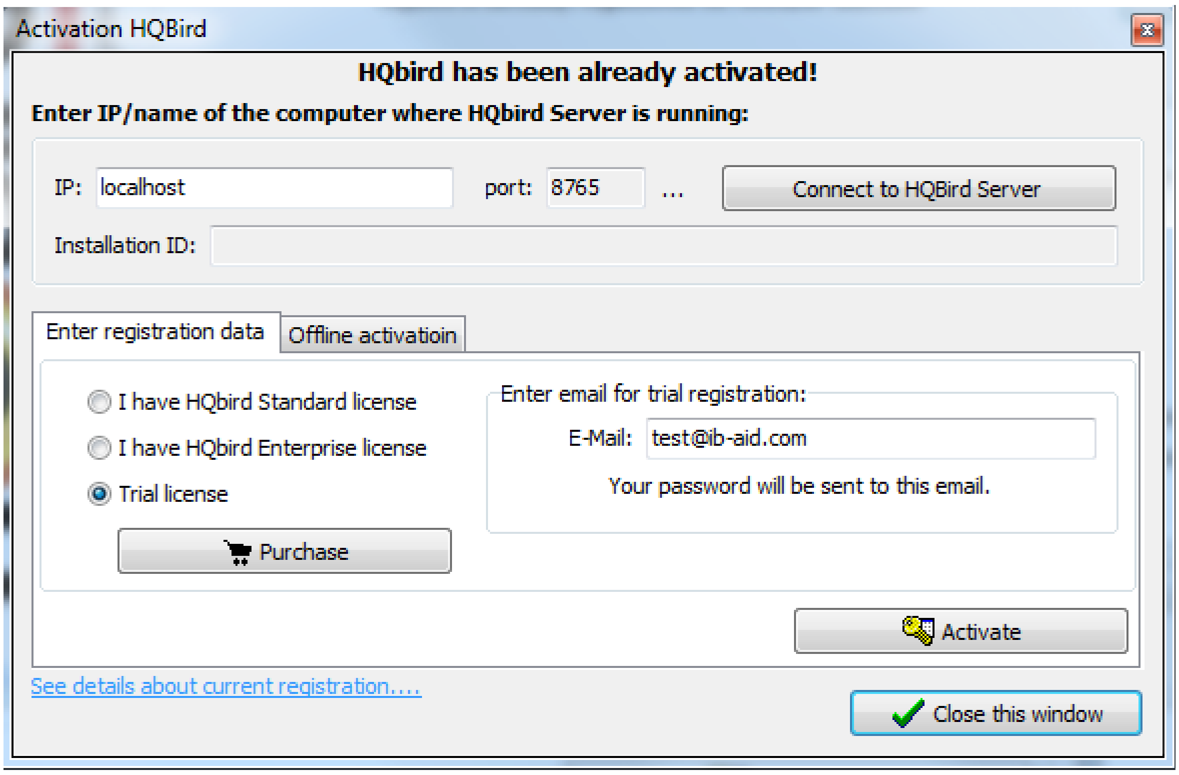

HQBird Launcher/Registration Helper
HQBird
Advanced FirebirdSQL Distribution for Enterprises
Re-register
See details...
Double-click tool name to start
Close
Contact support...

HQbird is already registered to: test@ib-aid.com

Name	Description
Firebird DataGuard	Monitoring, automatic backups and maintenance tasks
User guide	HQbird user guide
Transaction Monitor	Advanced transactions markers monitoring
Database Analyst	Database structure and statistics analysis
Scanner LogAnalyzer	Analyze logs from Firebird Scanner
MON$Logger	Fast adhoc analysis of MON$ tables
Performance Monitor	Setup TraceAPI monitoring (queries, triggers and stored procedures)
PerfMon Viewer	TraceAPI monitoring (queries, triggers and stored procedures

Copyright 2002-2017 IBSurgeon

If you click the **Register** button (or Re-Register for repeated registration), you will see the activation wizard:



HQbird has been already activated!

Enter IP/name of the computer where HQbird Server is running:

IP: localhost port: 8765 ...

Installation ID:

I have HQbird Standard license

I have HQbird Enterprise license

(● Trial license)

Enter email for trial registration:

E-Mail: test@ib-aid.com

Your password will be sent to this email.

After that, specify the **IP address** or the **computer name** of the server HQbird is installed on in the upper input field and click **Connect to HQbird Server**. If you started registration utility on the same computer with HQbird ServerSide, it will be «localhost», otherwise- some remote address.

Then enter your registration data. If you have a license, enter your e-mail address and password that you used to register with the IBSurgeon Deploy Center and click **Activate**.

If you have no license, choose **Trial license**, specify your e-mail address and click **Activate**. You will be automatically registered and the password will be sent to your e-mail address.

Right after you click **Activate**, the registration wizard will try to connect to the IBSurgeon Deploy Center (http://deploy.ib-aid.com) and obtain a license. If it succeeds, you will see the corresponding message. If there are any problems, you will see the error message.
If you forget the password, click the **Forgot password...** button and it will open the browser with the password recovery form.

If you need to purchase a new or additional license or renew your subscription, click **Purchase.**
Click **Close this window** after the registration is over.

Internet Activation via a Client Computer

If the server with HQbird Server-Side Tools installed on the server where it does not have access to the Internet, you can still activate it via the Internet: you can install HQbird Administrator on any client computer that has both access to the Internet and access to the HQbird Server-Side Tools server, and then perform activation.

Offline Activation

If the server and all client computers have no access to the Internet, you should use offline activation. To do it, click Offline activation tab and follow instructions there. In case of any troubles please contact support@ib-aid.com.

3. Configuration of HQbird

3.1. Initial configuration of HQbird FBDataGuard (backups, monitoring, alerts, etc)

Please follow these steps:

1. Make sure that you have Firebird 2.5.5 or later (Firebird 2.1 is also supported with some limitation), and it is working;
2. HQbird FBDataGuard service is installed and started properly.

2.1. You can check it using Services applet in Control Panel (right-click on "My Computer", choose "Manage", then "Services and Applications", "Services" and find in the list "FBDataGuard Agent"

2.2. At Linux you can run command «ps aux | grep dataguard».

3. Make sure the FBDataGuard port is accessible (8082) and it is not blocked by firewall or any other antivirus tools. If necessary, adjust port in FBDataGuard configuration (see ***Adjusting web-console port***).

Launch web-console

To open web-console type in your browser http://localhost:8082 or http://127.0.0.1:8082or use IP address of your server with installed HQbird ServerSide.

Or you can choose in "Start" menu IBSurgeon\HQbird Server Side\Firebird DataGuard\"Launch the DataGuard web console for localhost".

Supported browsers

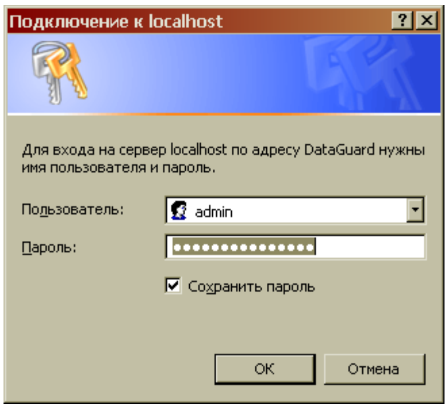
FBDataGuard web-console should work correctly with Firefox, Chrome, Safari and Internet Explorer 11.

Error message regarding webs-site certificate

If you have configured DataGuard to use https, the browser will indicate that this web-site (https://localhost:8082) is not safe, and it will recommend leaving web-site. This message is caused by the default security certificate for FBDataGuard web-console.

Please ignore this message and click to open FBDataGuard web-console.

It will ask you for username and password (login dialog can be different for Firefox, Safari or Chrome).



Для входа на сервер localhost по адресу DataGuard нужны имя пользователя и пароль.

Пользователь: admin

Пароль: •••••••••••••••

☑ Сохранить пароль

Enter username and password for FBDataGuard web-console

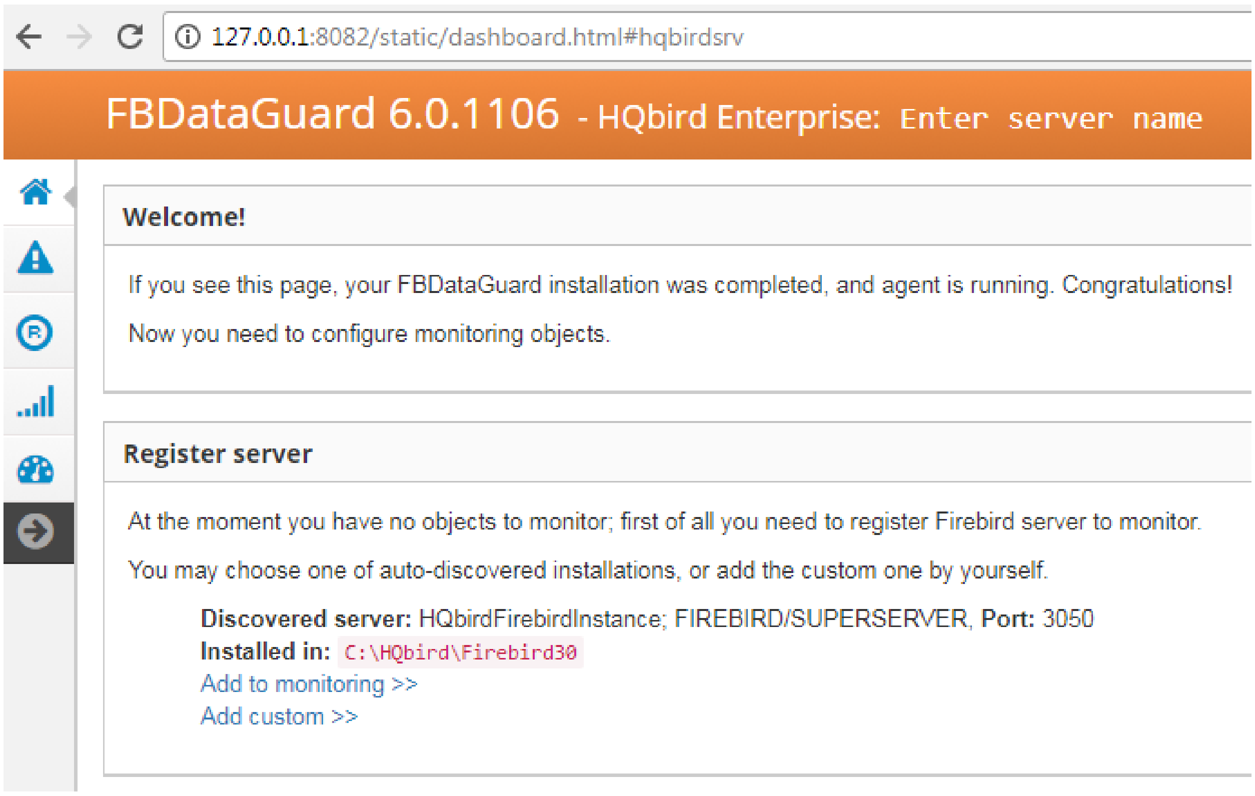
Attention! Default username/password for HQbird FBDataGuard is "**admin"/"strong password**" (without quotes).

Auto discovery feature of FBDataGuard

At the first launch FBDataGuard will check computer for installed Firebird servers. FBDataGuard for Windows search registry for Firebird records, FBDataGuard for Linux checks default locations of Firebird installations.

FBDataGuard will show the list of all Firebird copies installed, but only the one instance of Firebird can be monitored by FBDataGuard. Choose it by clicking ***"Add to monitoring >>"***

If you don't see Firebird instance in auto discovery list, you can choose ***"Add custom >>"*** and configure instance parameters manually.


127.0.0.1:8082/static/dashboard.html#hqbirdsrv
FBDataGuard 6.0.1106 - HQbird Enterprise: Enter server name

Welcome!

If you see this page, your FBDataGuard installation was completed, and agent is running. Congratulations!

Now you need to configure monitoring objects.

Register server

At the moment you have no objects to monitor; first of all you need to register Firebird server to monitor.

You may choose one of auto-discovered installations, or add the custom one by yourself.

Discovered server: HQbirdFirebirdInstance; FIREBIRD/SUPERSERVER, **Port:** 3050
Installed in: `C:\HQbird\Firebird30`
Add to monitoring >>
Add custom >>

Auto discovery feature of HQbird

Firebird server registration

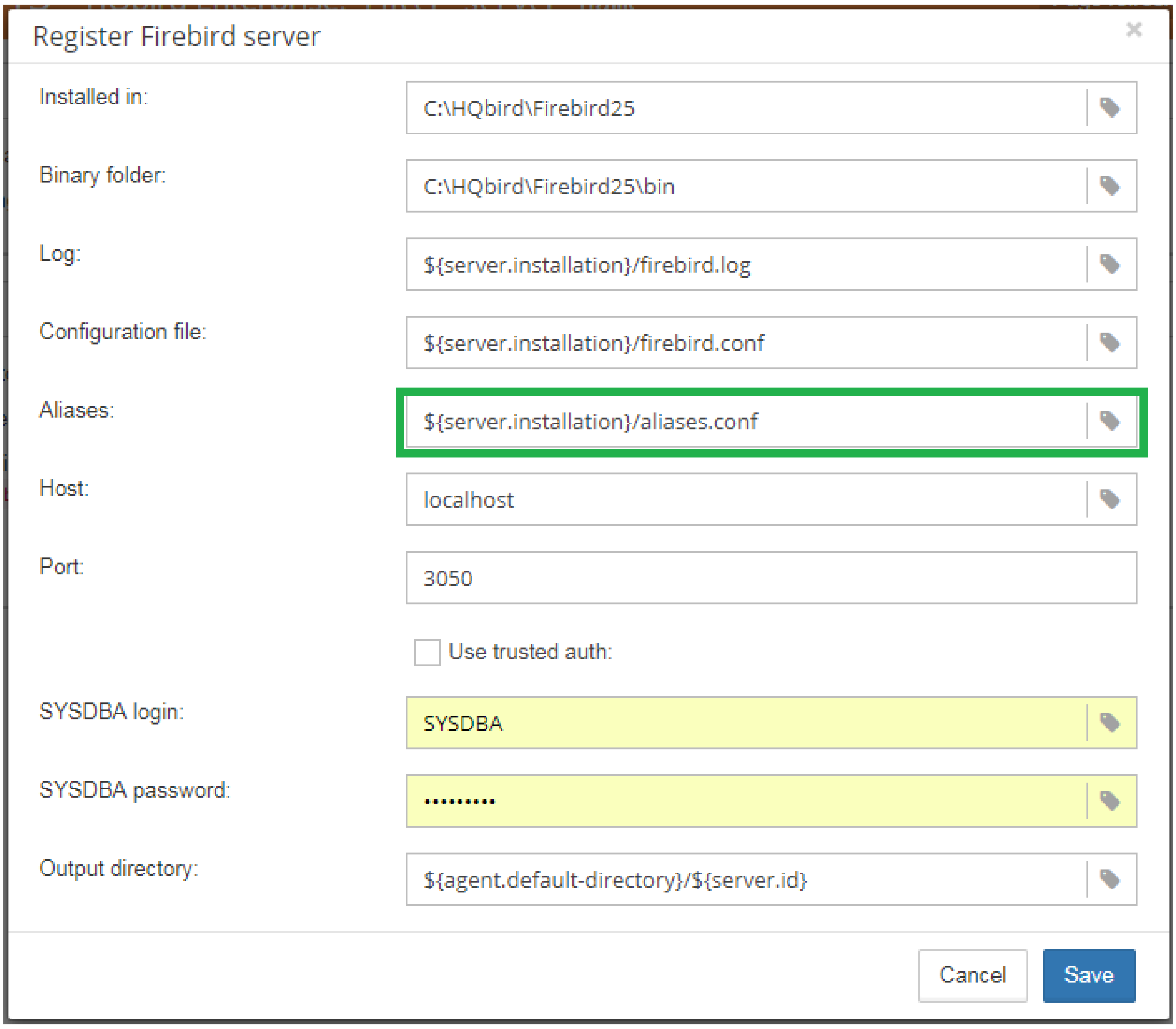
To register auto-discovered server, you need to click at "Add to monitoring>>" and then adjust auto-discovered settings.

Note: to use Windows Trusted Authentication (by default it's off), you need to be sure that libraries jaybird30.dll and fbclient.dll (from appropriate Firebird version) are in searchable Windows paths.

Let's consider what can you see in the Server dialog (and, normally, you don't need to change them):

- **Installed in**: Firebird installation folder

- **Binary folder**: Firebird bin folder (for Firebird 3 on Windows Binary folder is the same as the installation folder)
- Log: location of firebird.log
- **Configuration file**: location of firebird.conf
- **Aliases**: location of aliases.conf or, for Firebird 3, databases.conf (**please change it manually, if needed**)
- **Host**: name of the server, usually localhost
- **Port**: network port for Firebird, according firebird.conf settings
- **Use trusted auth**: use trusted authentication, by default it is off
- **SYSDBA login**: name of SYSDBA user, usually it is SYSDBA
- **SYSDBA password: password** for SYSDBA
- **Output directory**: Folder where backups, statistics and gathered logs will be stored


Register Firebird server

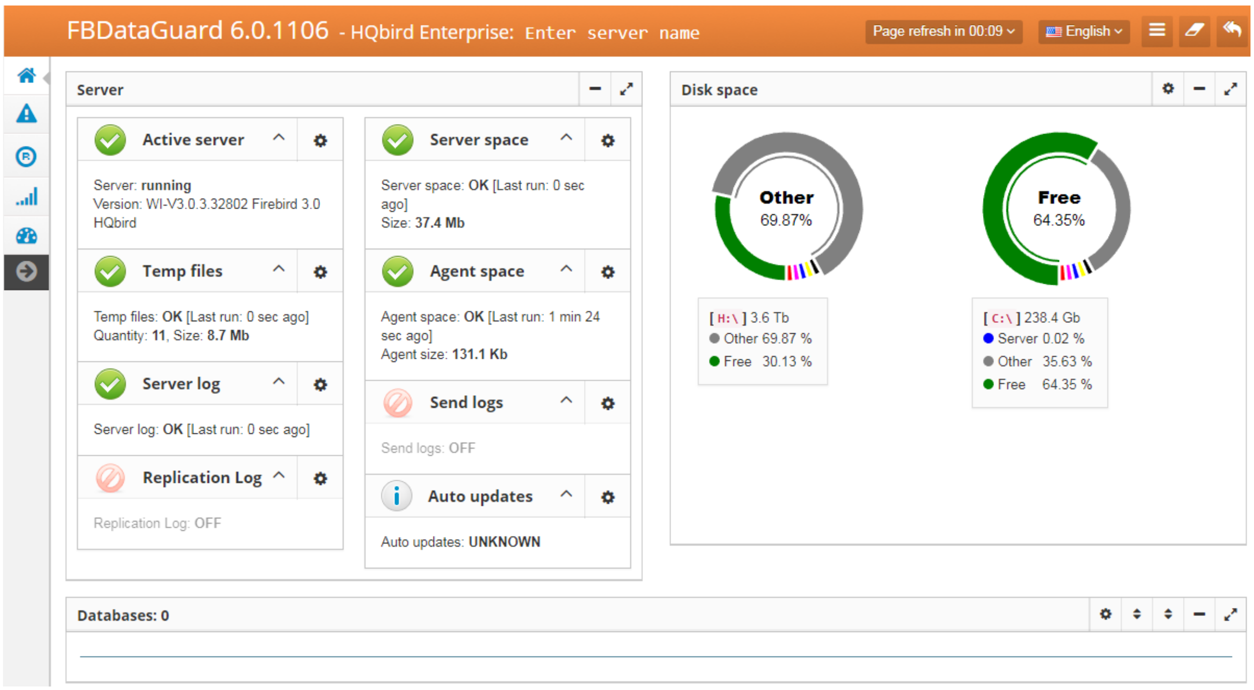
Installed in:	C:\HQbird\Firebird25
Binary folder:	C:\HQbird\Firebird25\bin
Log:	${server.installation}/firebird.log
Configuration file:	${server.installation}/firebird.conf
Aliases:	${server.installation}/aliases.conf
Host:	localhost
Port:	3050
	☐ Use trusted auth:
SYSDBA login:	SYSDBA
SYSDBA password:	•••••••••
Output directory:	${agent.default-directory}/${server.id}

Register server in FBDataGuard

By default “Output directory” for Firebird server is **${agent.default-directory}/${server.id}**

It can be not very convenient, so we recommend pointing FBDataGuard output directory to more simple path, usually located at disk where backups are intended to be stored, for example **F:\myserverdata**

After clicking "Register" FBDataGuard will populate default configurations files and immediately start analysis of firebird.log. It can take a while (for example, 1 minute for 100Mb firebird.log). After that you will see initial web-console with registered Firebird server:



Active server
Server: running
Version: WI-V3.0.3.32802 Firebird 3.0 HQbird

Temp files
Temp files: OK [Last run: 0 sec ago]
Quantity: 11, Size: 8.7 Mb

Server log
Server log: OK [Last run: 0 sec ago]

Replication Log
Replication Log: OFF

Server space
Server space: OK [Last run: 0 sec ago]
Size: 37.4 Mb

Agent space
Agent space: OK [Last run: 1 min 24 sec ago]
Agent size: 131.1 Kb

Send logs
Send logs: OFF

Auto updates
Auto updates: UNKNOWN

[H:\] 3.6 Tb
● Other 69.87 %
● Free 30.13 %

[C:\] 238.4 Gb
● Server 0.02 %
● Other 35.63 %
● Free 64.35 %

Web-console with registered Firebird server

FBDataGuard shows alerts and statuses of monitored objects: if everything is fine, it shows green signs, otherwise there will be yellow or red notifications.

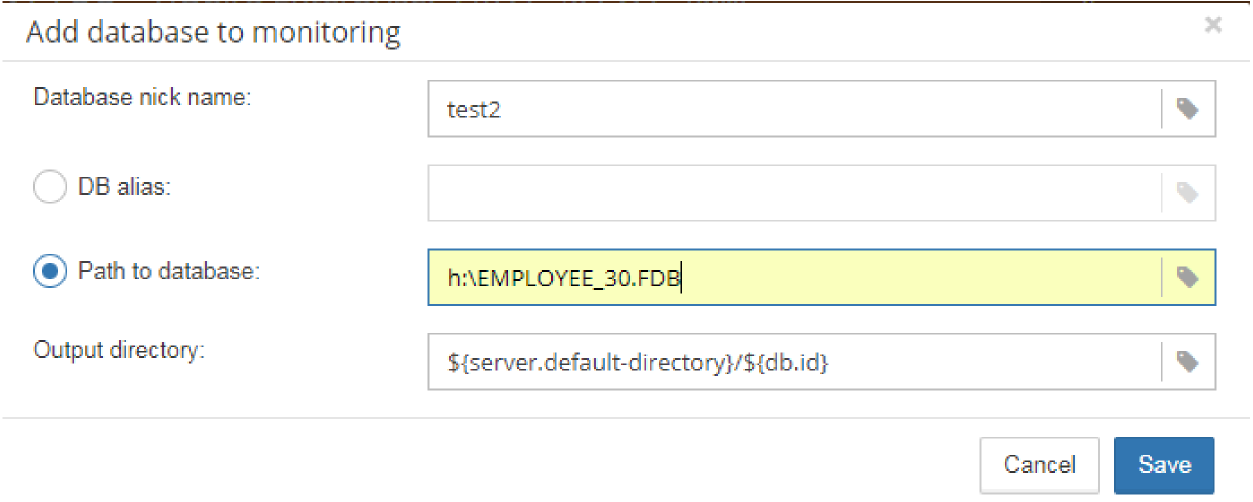
In [3.2. Monitoring and maintenance configuration in FBDataGuard] we will consider in details each monitored objects and its settings.

Note: you cannot delete registered Firebird server in FBDataGuard web-console. The only way to unregister server is to delete its configuration files. In general, there is no reason for deleting registered server, until you want completely uninstall FBDataGuard.

Now we let's proceed with a database registration.

Firebird database registration

To register database in FBDataGuard, you need to click at the symbol «Settings» in the right corner of «Databases» (there will be a hint «Add database to monitoring») and fill the following form:

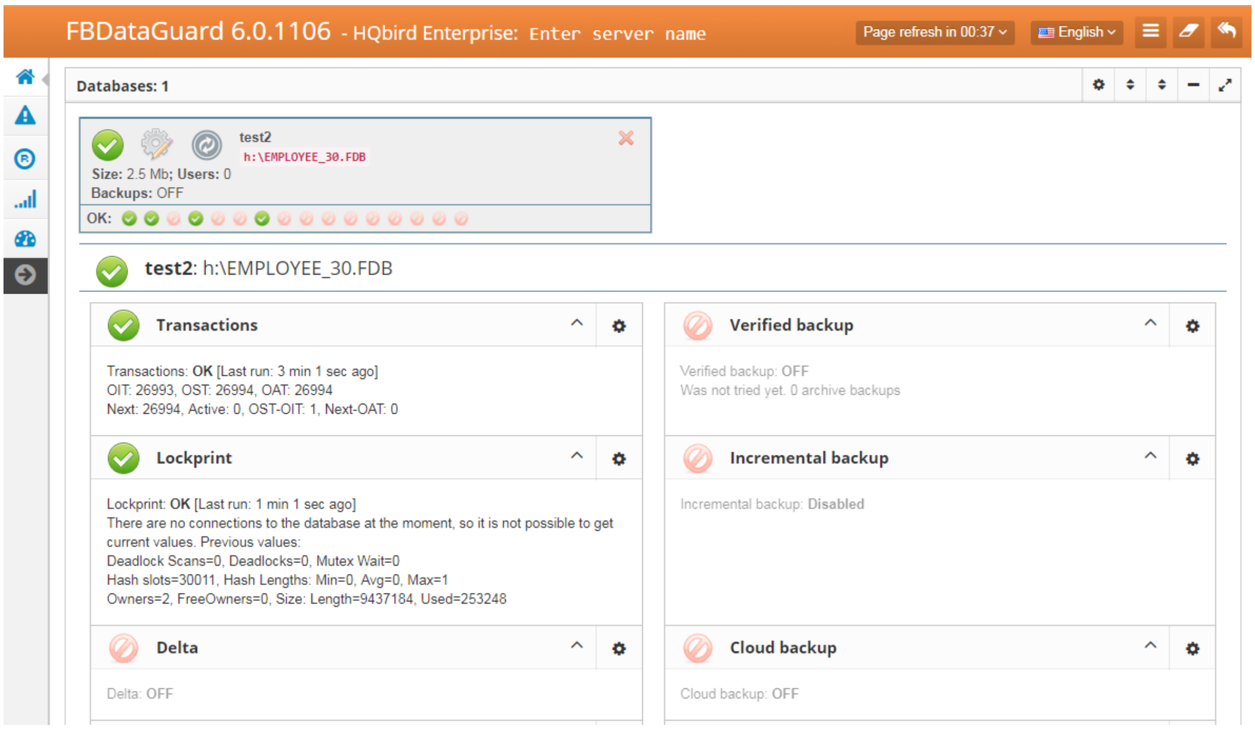

Add database to monitoring

Database nick name:	test2
○ DB alias:	
● Path to database:	h:\EMPLOYEE_30.FDB
Output directory:	${server.default-directory}/${db.id}

- "**Database nick name**" is for your convenience, it is used to refer this database in alerts and email messages.
- "**DB alias**" is a database alias from aliases.conf or in databases.conf. If you specify both "DB Alias" and "Path to database", "DB Alias" will be used.
- "**Path to database**" is the local path to database (remember that FBDataGuard should work at the same computer with Firebird). If you are putting database on external drive, it can raise error "File… has unknown partition". To fix it you need to click on "Configure" at Server widget and click "Save" to make FBDataGuard re-read partitions.
- "**Output directory**" is the folder where FBDataGuard will store backups, logs and statistics for this database. If you do not select HQbirdData folder during the installation, and if you do not specify output folder for the server, it's a good idea to specify "Output directory" to some explicit location like F:\mydatabasedata.

Note: you can specify exact absolute locations for backups and statistics later in appropriate dialogs (see [3.2. Monitoring and maintenance configuration in FBDataGuard]).

After registration, FBDataGuard will populate database configuration with default values and then show web-console with registered database:


FBDataGuard 6.0.1106 - HQbird Enterprise: Enter server name

Databases: 1

test2
h:\EMPLOYEE_30.FDB
Size: 2.5 Mb; Users: 0
Backups: OFF
OK:

test2: h:\EMPLOYEE_30.FDB

Transactions
Transactions: OK [Last run: 3 min 1 sec ago]
OIT: 26993, OST: 26994, OAT: 26994
Next: 26994, Active: 0, OST-OIT: 1, Next-OAT: 0

Verified backup
Verified backup: OFF
Was not tried yet. 0 archive backups

Lockprint
Lockprint: OK [Last run: 1 min 1 sec ago]
There are no connections to the database at the moment, so it is not possible to get current values. Previous values:
Deadlock Scans=0, Deadlocks=0, Mutex Wait=0
Hash slots=30011, Hash Lengths: Min=0, Avg=0, Max=1
Owners=2, FreeOwners=0, Size: Length=9437184, Used=253248

Incremental backup
Incremental backup: Disabled

Delta
Delta: OFF

Cloud backup
Cloud backup: OFF

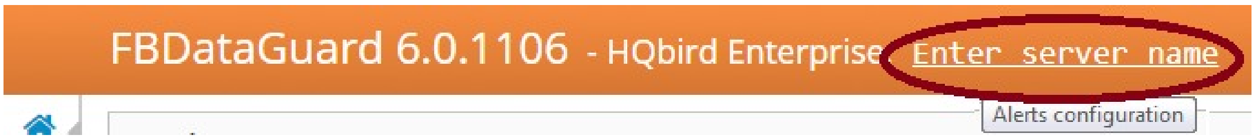
FBDataGuard web-console after adding a database

You can adjust database settings later; now let's proceed with alerts setup.

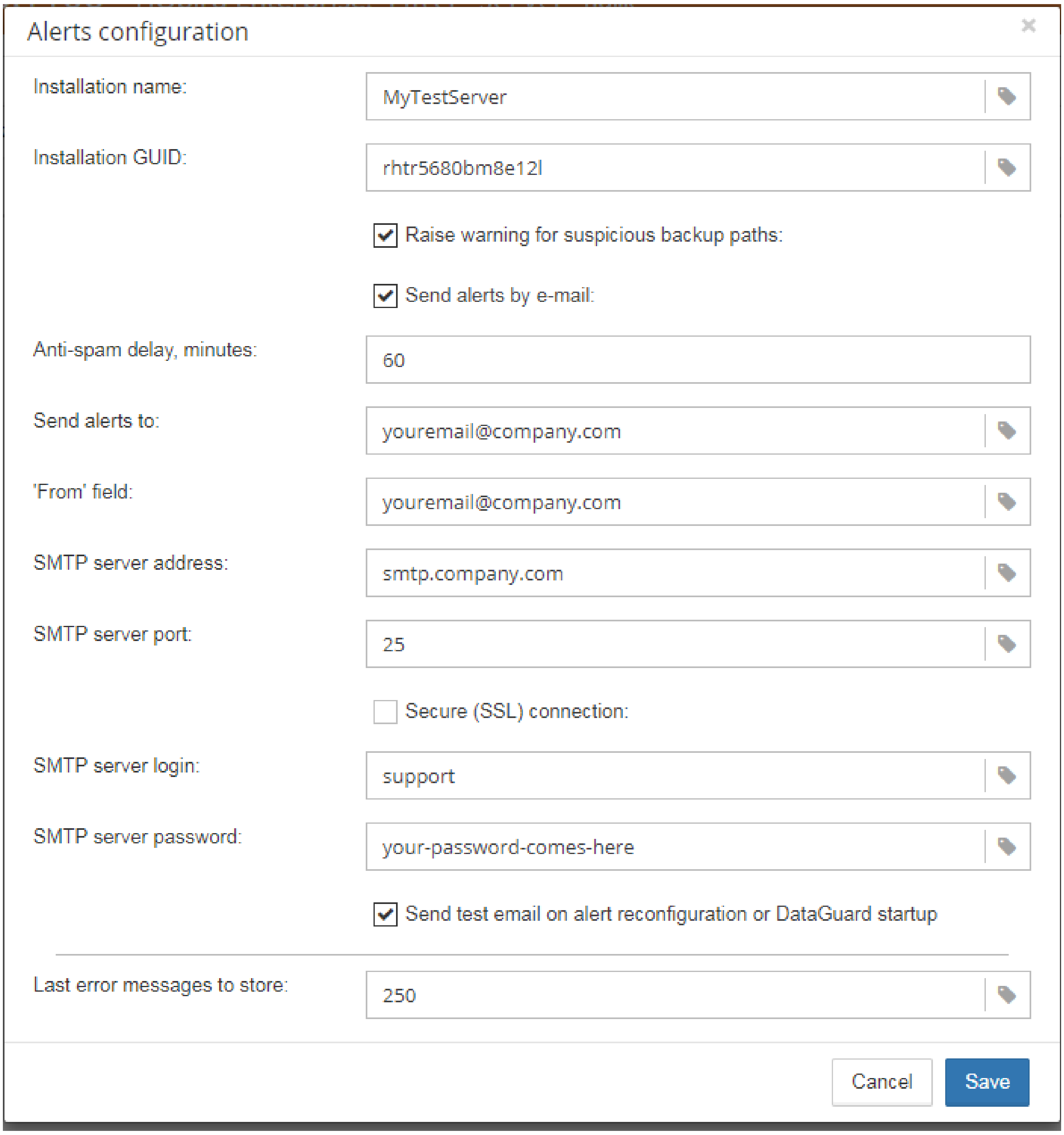
Email alerts in HQbird FBDataGuard

FBDataGuard can send alerts by email to administrator(s): such alerts contain information about successful backups and potential and real problems with databases.

It's a good idea to enable setup email alerts. To do this you need to click on placeholder «Enter server name» in the top of the web-console:



After that you will see the configuration dialog for alerts:



Alerts configuration

Installation name:	MyTestServer
Installation GUID:	rhtr5680bm8e12l
	[✓] Raise warning for suspicious backup paths:
	[✓] Send alerts by e-mail:
Anti-spam delay, minutes:	60
Send alerts to:	youremail@company.com
'From' field:	youremail@company.com
SMTP server address:	smtp.company.com
SMTP server port:	25
	[ ] Secure (SSL) connection:
SMTP server login:	support
SMTP server password:	your-password-comes-here
	[✓] Send test email on alert reconfiguration or DataGuard startup
Last error messages to store:	250

Email alerts configuration dialog in FBDataGuard

First of all, you need to enable alerts sending by enabling checkbox “**Send alerts by e-mail**”.

- “**Installation name**” is some readable name for your convenience; it will be referred in emails and alerts.
- “**Installation GUID**” is a service field; there is no need to change it.
- “**Raise warning for suspicious backup paths**” is to warn about potentially wrong backup file path, for example, if database is stored in “Documents and Settings” or if the length of backup path is too long.
- “**Anti-spam delay**” specifies delay on sending repeating messages. It avoids flooding of administrators' mailbox with repetitive messages. 60 minutes is an optimal value.

- “**Send alerts to**” specifies where to send emails.
- “**From field**” is what will be set as sender in the email.
- “SMTP server address”, “SMTP server port”, “SMTP server login” and “SMTP server password” are data which will be used to send emails.

Please note: if you are using smtp.gmail.com, use port 587 and enable login of third-party applications in the settings of Gmail account.

Click “Save” to save email alerts settings.

Note. There can be delay while saving, since FBDataGuard is checking the settings and send ***test*** email to the specified address.

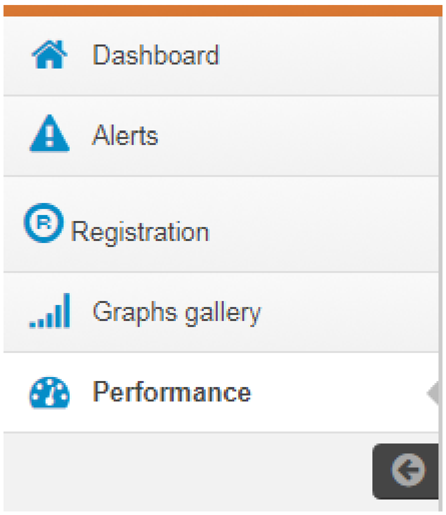
Next steps with FBDataGuard

After you have setup FBDataGuard and added there server and database(s) to be monitored, you need to adjust settings for the most important maintenance activities: backups, sweep and performance monitoring.

3.2. Monitoring and maintenance configuration in FBDataGuard

Overview of web-console

Parts of web-console



FBDataGuard Web-console contains 5 tabs (in the left side of the screen, usually they are collapsed):

- *Dashboard*–it is the main tab where administrator can configure HQbird FBDataGuard and see server and databases statuses.
- *Alerts* – contains the full list of alerts, generated by FBDataGuard
- *Registration – license and registration/activation information,*
- *Graphs gallery– collection of various graphs*
- *Performance – performance monitoring settings and reports*.

Jobs

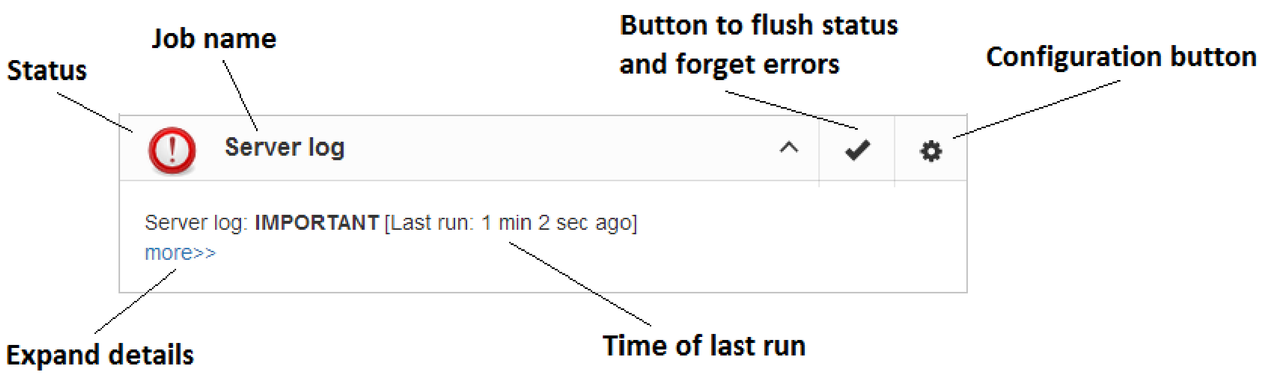
Web-console is intended for easy configuration of activities (called "**jobs**") which are fulfilled by HQbird FBDataGuard.

Almost all FBDataGuard jobs have 2 purposes: the first is to monitor for some values and to raise alerts if necessary, and the second is to store historical values into logs, so later it's possible to see the dynamics of important parameters of Firebird server and database.

In this section we will consider general configuration of jobs parameters, but not an analysis of gathered logs.

Jobs widgets

General approach is the following: each activity is represented by a "widget", which has the following parts:


Status
Job name
Button to flush status and forget errors
Configuration button
Server log
Server log: IMPORTANT [Last run: 1 min 2 sec ago]
more>>
Expand details
Time of last run

Elements of FBDataGuard web-console widget

Status – it is indicated with color icon and name. Status of database is a summary of all included server-level jobs and databases' statuses, and, respectively, status of database is a summary of all database-level jobs.

Status types

CRITICAL means problems, OK means "Everything is fine", WARNING means some issues which require attention, MAJOR means major issue, MINOR – minor issue, MALFUNCTION means that jobs was not succeeded (something prevents its execution), NOT_AVAILABLE means that job is not available in this server or database version.

OFF means that job is not active, UNKNOWN means that job is active but was not started yet, so actual results are unknown.

Job name indicates the name of activity.

Configuration link opens configuration dialog, which is individual for each job.

Resolved is the link to flush the status to UNKNOWN and forgot errors which were discovered previously. The status will be updated according the current situation after the next execution of the job.

Last run shows the time after the last run of this job.

More>> is the link which opens the widget and shows more details and suggested action for administrator to resolve the situation.

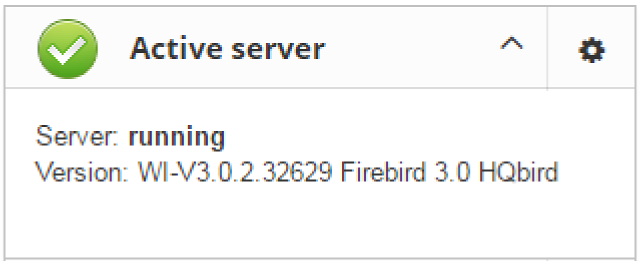
All jobs in FBDataGuard have default settings, which are very close to recommended values for the 80% of Firebird installations, so after initial configuration server and database will be protected at pretty good level comparing with default installation, but anyway we recommend additional configuration and tuning for every job.

In the next sections we will consider each job and its configuration.

Server: Active server

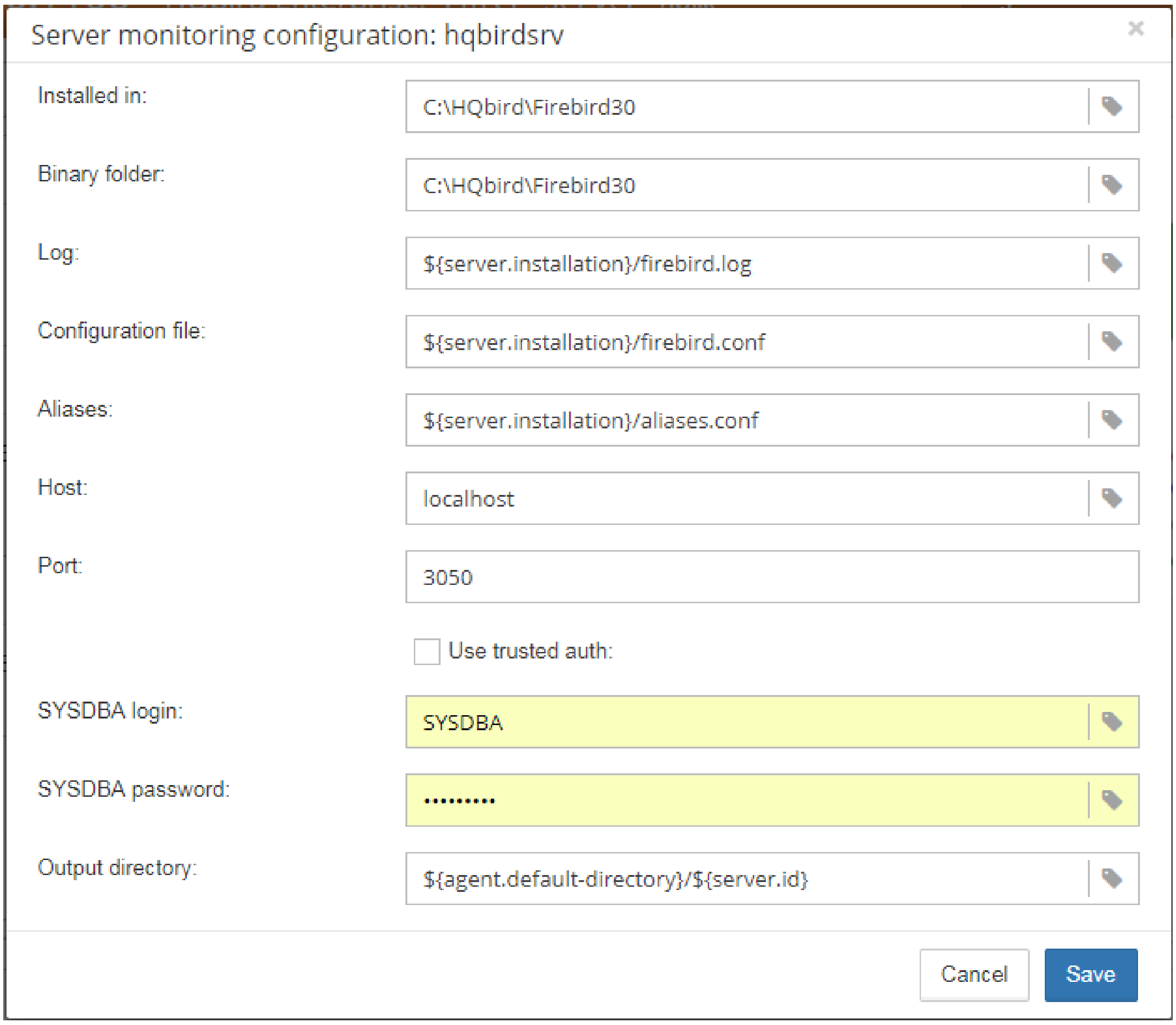
Server: Active server widget shows summarized status of all server-level jobs and statuses of monitored databases.

Server: Active server also indicates Firebird currently running or not and shows detailed version of Firebird and HQbird.



Server: **running**
Version: WI-V3.0.2.32629 Firebird 3.0 HQbird

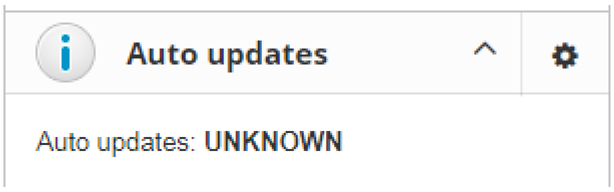
If you click on **configure** link, you will see the same dialog that we have used to register Firebird instance in FBDataGuard, and now it can be used for changing Firebird instance properties:


Server monitoring configuration: hqbirdsrv

Installed in:	C:\HQbird\Firebird30
Binary folder:	C:\HQbird\Firebird30
Log:	${server.installation}/firebird.log
Configuration file:	${server.installation}/firebird.conf
Aliases:	${server.installation}/aliases.conf
Host:	localhost
Port:	3050
	☐ Use trusted auth:
SYSDBA login:	SYSDBA
SYSDBA password:	•••••••••
Output directory:	${agent.default-directory}/${server.id}

In general, there is no need to edit Firebird server details after the registration, until you are not reinstalling Firebird – but in this case we recommend reinstalling HQBird too.

Server: Auto updates

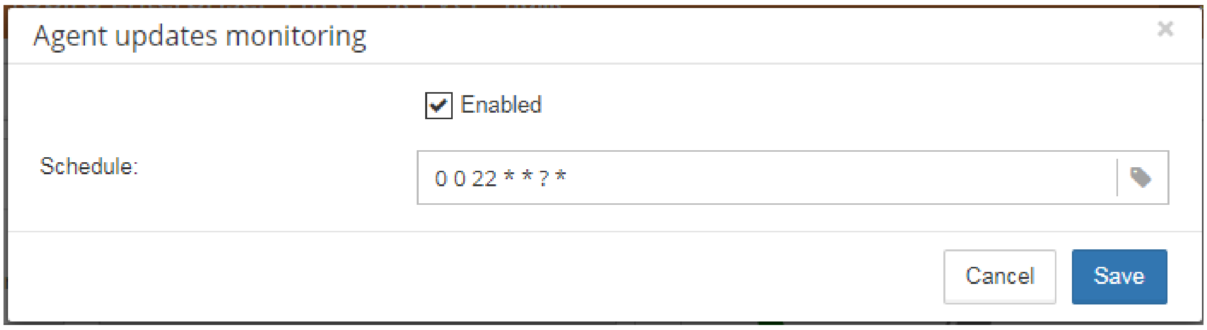


Auto updates: UNKNOWN

Auto update is an important job which notifies you that newer version of HQbird FBDataGuard is available and suggests to update the software. It provides an alert regarding available updates and appropriate download link. Default time to run this job is 22-00 everyday (for information).

At configuration dialog of auto-updates you can disable auto check or set another time for it.

For more information about time format please refer to ***Appendix: CRON Expressions***



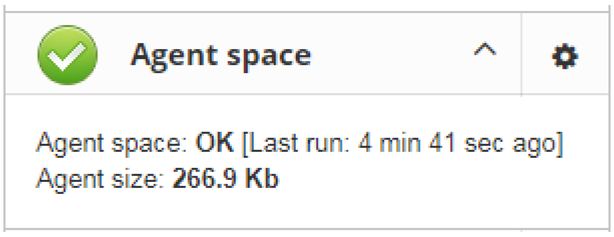
[✓] Enabled

Schedule: 0 0 22 * * ? *

In fact there is a small confusion here: auto update does not perform automatic update of software, it just checks for the new versions periodically. The upgrade is always done manually.

Server: Agent Space

Agent space monitoring is intended to watch space occupied by logs, stats, metadata repository and other data, gathered by FBDataGuard. For databases unattended for a long time (1-2 years) it is possible that FBDataGuard logs will occupy too much space and lack of space can lead to database outage. To prevent it for sure, Agent Space is watching for occupied space.



Agent space: OK [Last run: 4 min 41 sec ago]
Agent size: 266.9 Kb

By default Server: Agent Space job is enabled.

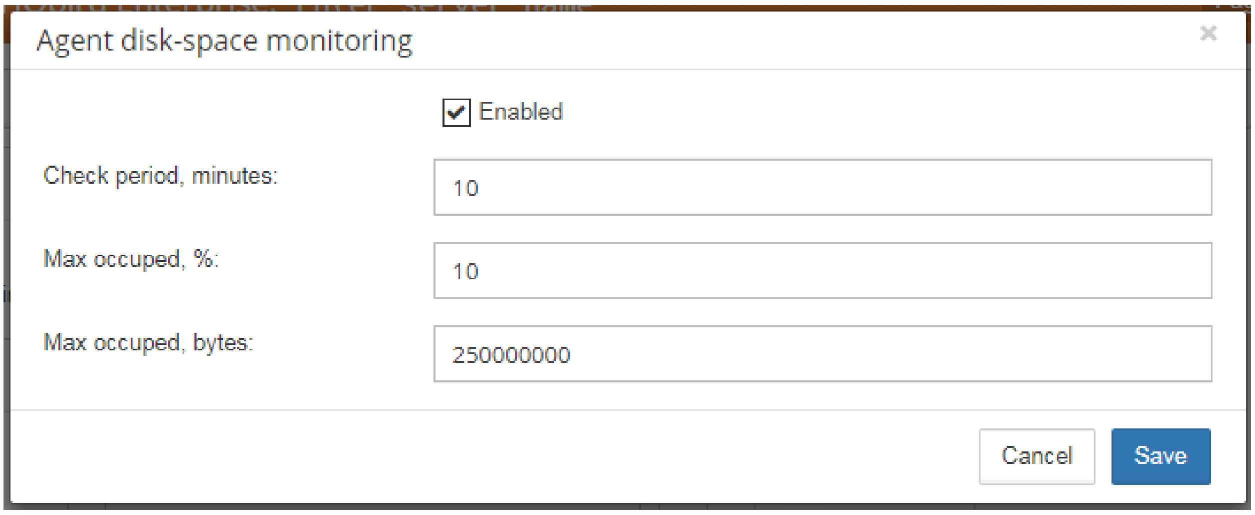
Also, if someone has ignored recommendations to put backups' folders to the explicit locations, it is possible that database backup will be created inside Agent folder. In this case you'll see CRITICAL status immediately – FBDataGuard will recognize and warn you regarding wrong configuration.

And, this job is useful for bundles of FBDataGuard and third-party applications.

In the configuration dialog you can enable/disable this job, set check period (by default it is 10 minutes), and set thresholds for alerts.

Thresholds can be set in % of max size occupied by log or using the explicit size in bytes.

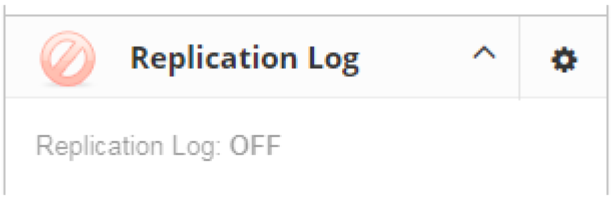
FBDataGuard checks both values and raises alert for the first threshold. If you wish to set % only, you need to set -1 as value to "Max occupied".



[✓] Enabled

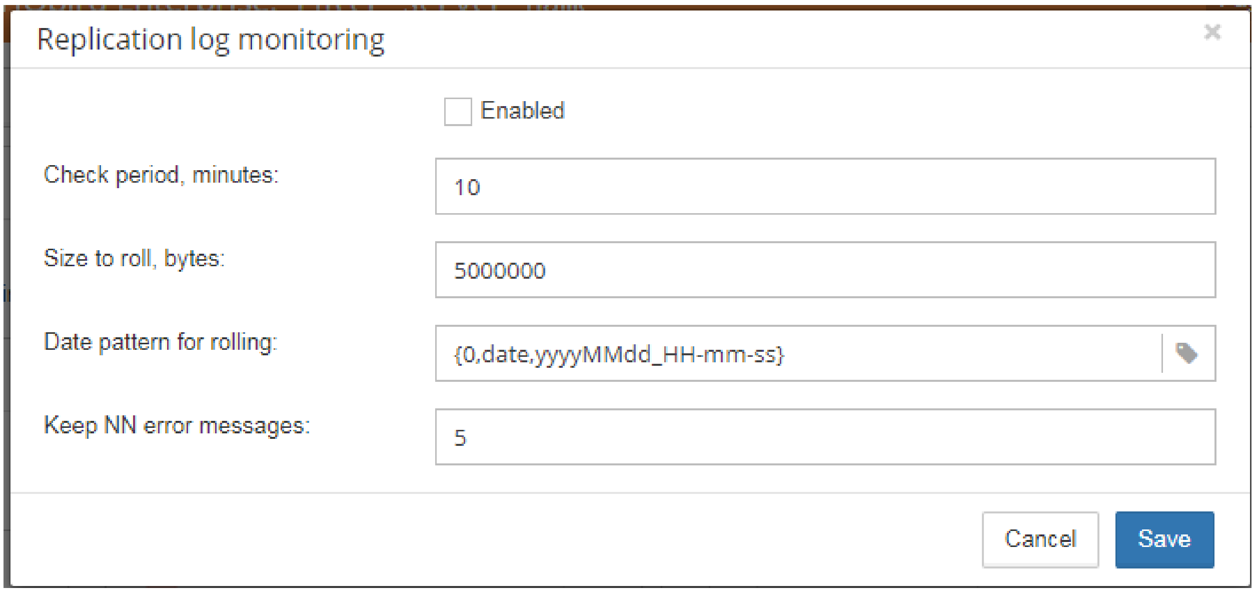
Check period, minutes:	10
Max occuped, %:	10
Max occuped, bytes:	250000000

Server: Replication Log



If you are using HQbird Enterprise, FBDataGuard can check replication.log for errors. In case of error it sends an appropriate alert (by email) to the administrator.

To enable this job please check «Enabled.

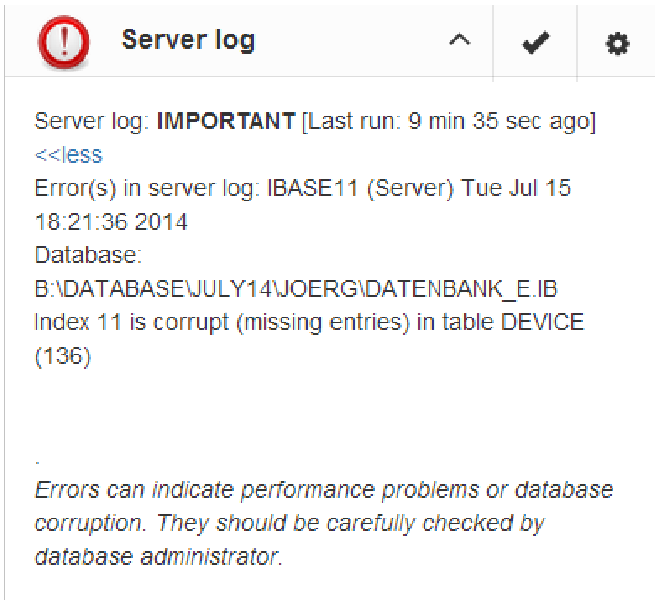

Replication log monitoring

	[ ] Enabled
Check period, minutes:	10
Size to roll, bytes:	5000000
Date pattern for rolling:	{0,date,yyyyMMdd_HH-mm-ss}
Keep NN error messages:	5

- Check period – how often to check replication.log for changes
- Size to roll, bytes – if replication.log will exceed will value, it will be renamed according date-time pattern
- Date pattern for rolling – how to rename replication.log
- Keep NN error messages: how many errors will be stored in the list of the recent errors.

Server: Server log

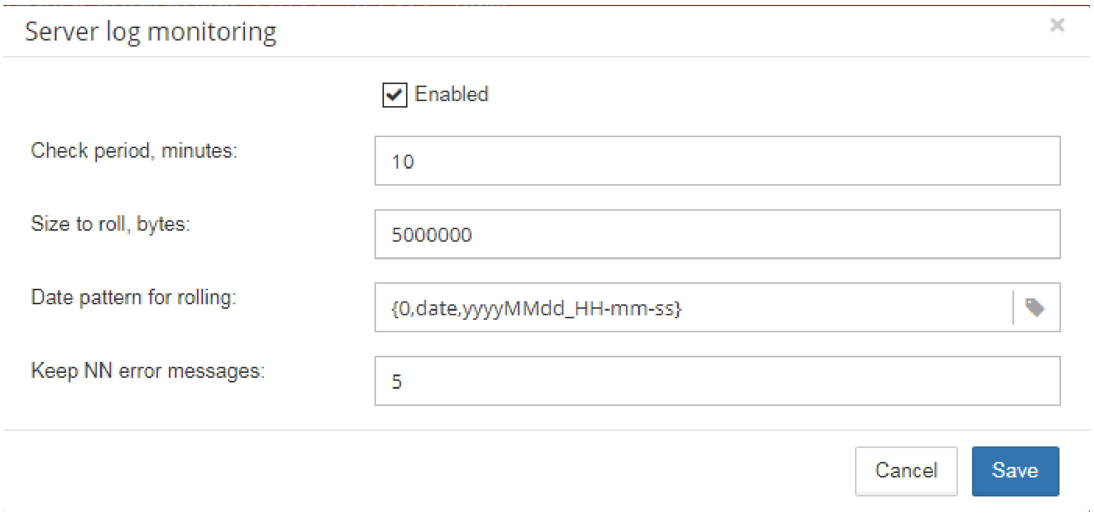
"Server log" job periodically checks firebird.log and if it detects that file was changed, log analysis starts.
The embedded analytic engine checks each entry in firebird.log and categorizes them into several categories with different levels of a severity.
According the severity of messages status of job is assigned and appropriate alerts are generated.
Once administrator has reviewed errors and alerts (and performed necessary actions to solve the reason of error), he clicks on "**Resolved"** link and FBDataGuard will forget old error messages in firebird.log.



Server log: **IMPORTANT** [Last run: 9 min 35 sec ago]
<<less
Error(s) in server log: IBASE11 (Server) Tue Jul 15 18:21:36 2014
Database: B:\DATABASE\JULY14\JOERG\DATENBANK_E.IB
Index 11 is corrupt (missing entries) in table DEVICE (136)

Errors can indicate performance problems or database corruption. They should be carefully checked by database administrator.

In the configuration dialog of "Server log" you can enable/disable this job and set the check period (in minutes).

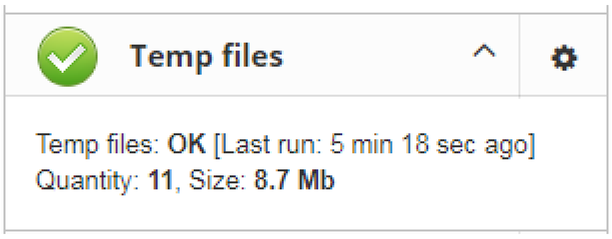


	✓ Enabled
Check period, minutes:	10
Size to roll, bytes:	5000000
Date pattern for rolling:	{0,date,yyyyMMdd_HH-mm-ss}
Keep NN error messages:	5

Also this job watches for the size of firebird.log and if its size exceeds "Size to roll", FBDataGuard will split firebird.log and rename it according to the date-time pattern.

Parameter "Keep NN error messages" specifies how many the most recent error messages will be stored.

Server: Temp files

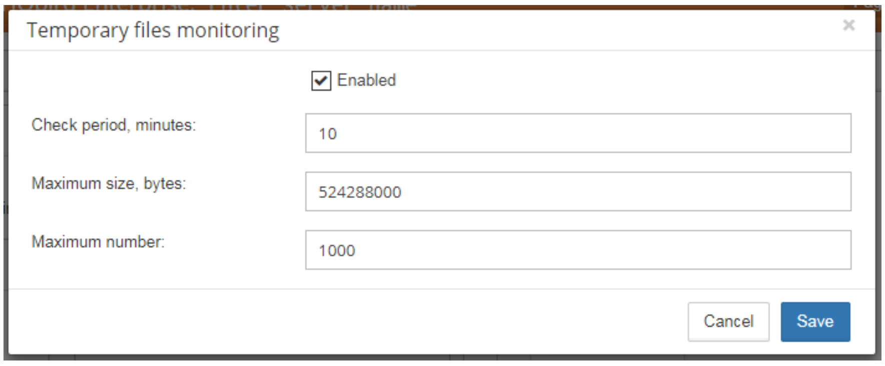


Temp files: OK [Last run: 5 min 18 sec ago]
Quantity: 11, Size: 8.7 Mb

"Server: Temp files" job is useful to catch and solve performance problems with Firebird database.

While performing SQL queries Firebird stores intermediate results for sorting and merging data flows in temporary files, which are allocated in specified TEMP locations. FBDataGuard shows at "Server: Temp files" widget information about quantity and size of temporary files.

FBDataGuard recognizes locations of TEMP folders and monitors quantity and size of temporary files. Lack of space can lead to the performance problem or more serious errors, too many (or too large) temporary files can indicate problems with SQL queries quality.

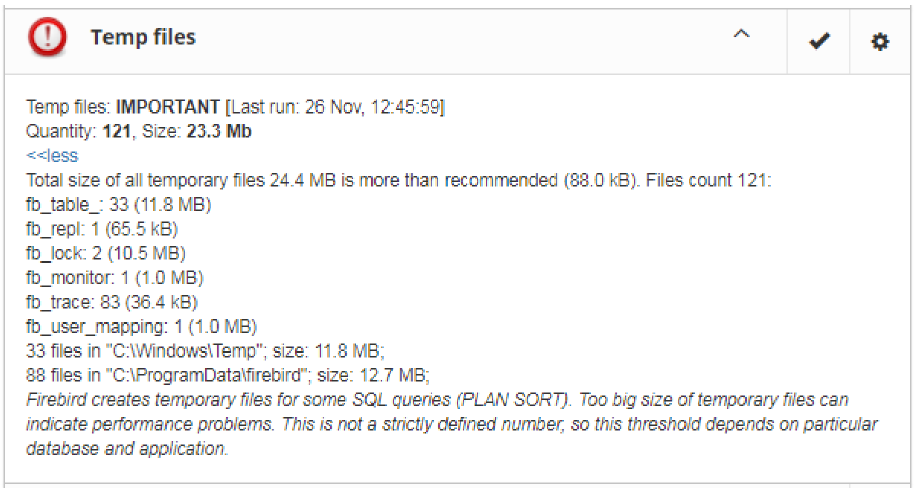


	✓ Enabled
Check period, minutes:	10
Maximum size, bytes:	524288000
Maximum number:	1000

Using configuration dialog you can enable/disable this job, set period and thresholds to the maximum size of temporary files (size of all files) and quantity.

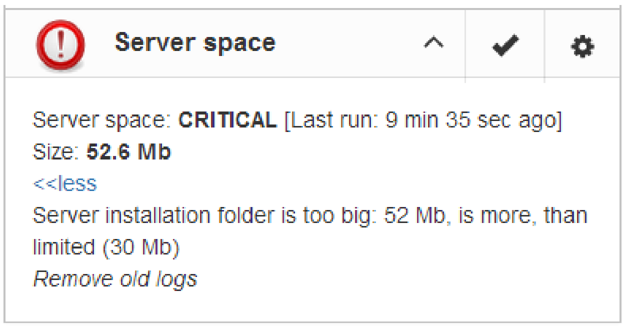
If you see that size of temp files is too high and if there is enough RAM on the server, increase TempCacheLimit parameter in firebird.conf to fit all temporary tables into RAM.

Also, HQbird checks other temp files used by Firebird – if you see extreme values (several Gb) for trace or monitor, the good idea will be check the FIREBIRD_TMP folder for outdated files (with old modification timestamps). Please note – the screenshot below is not a real alert (i.e., values are Ok), it was created to demonstrate the output in case of large temporary files.



Temp files: **IMPORTANT** [Last run: 26 Nov, 12:45:59]
Quantity: **121**, Size: **23.3 Mb**
<<less
Total size of all temporary files 24.4 MB is more than recommended (88.0 kB). Files count 121:
fb_table_: 33 (11.8 MB)
fb_repl: 1 (65.5 kB)
fb_lock: 2 (10.5 MB)
fb_monitor: 1 (1.0 MB)
fb_trace: 83 (36.4 kB)
fb_user_mapping: 1 (1.0 MB)
33 files in "C:\Windows\Temp"; size: 11.8 MB;
88 files in "C:\ProgramData\firebird"; size: 12.7 MB;
Firebird creates temporary files for some SQL queries (PLAN SORT). Too big size of temporary files can indicate performance problems. This is not a strictly defined number, so this threshold depends on particular database and application.

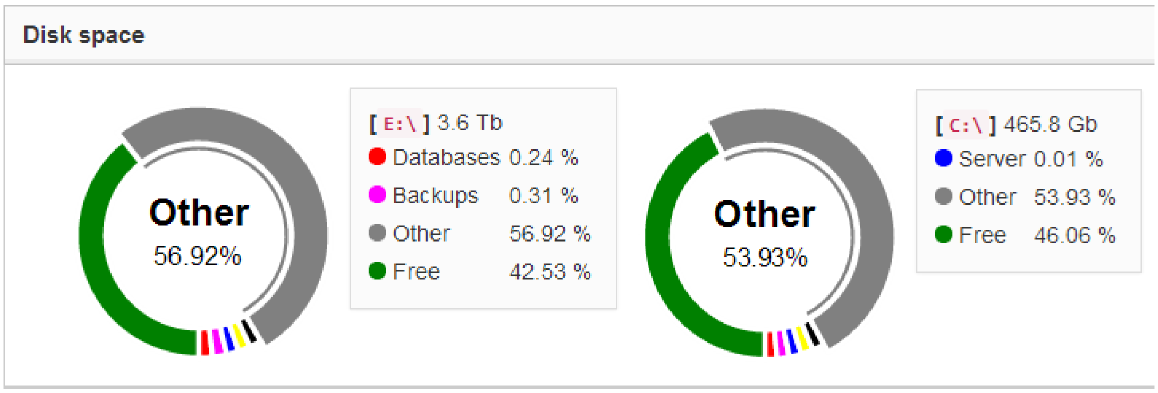
Server: Server space



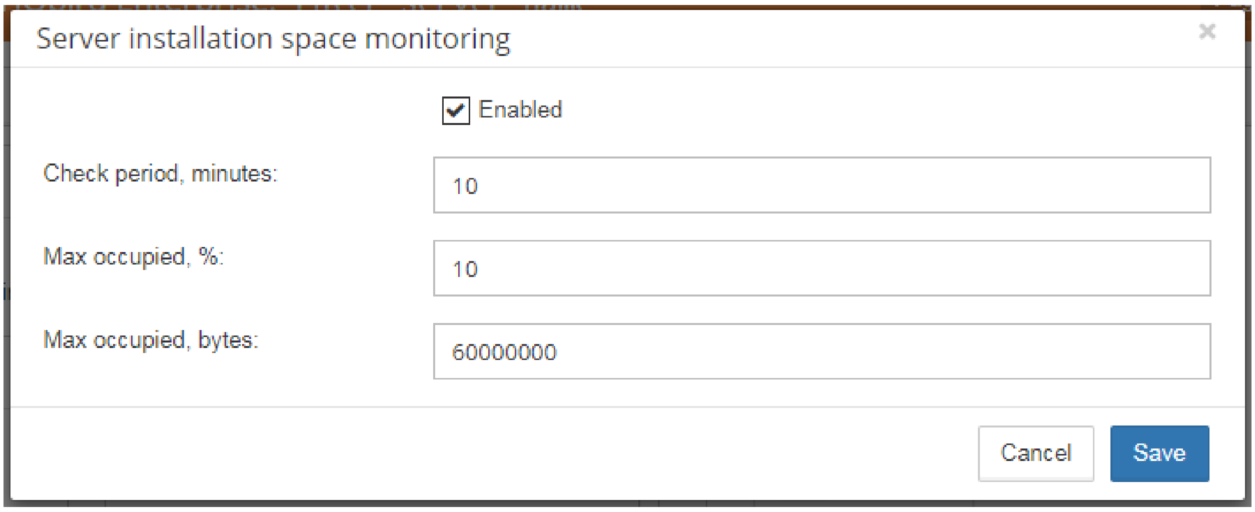
Server space: **CRITICAL** [Last run: 9 min 35 sec ago]
Size: **52.6 Mb**
<<less
Server installation folder is too big: 52 Mb, is more, than limited (30 Mb)
Remove old logs

"Server space" jobs monitors size, occupied by Firebird installation.
It's enabled by default.
There are several threats prevented by this job: maladministration issues when database volumes or external tables are being created in %Firebird %\Bin folder, very big **firebird.log** which can exhaust all places at drive with Firebird installation, and some other problems.

Also this job monitors and analyses information, gathered by all space-related jobs (including database-level jobs). At the picture below you can see quick representation of space analysis for all drives where Firebird, databases and backups are stored.



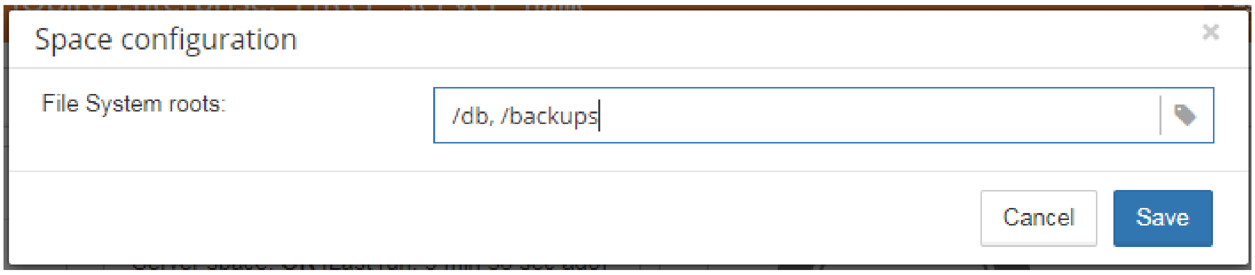
Using configuration dialog you can enable/disable this job, set period of checking and thresholds for server folder size.



[✓] Enabled
Check period, minutes: 10
Max occupied, %: 10
Max occupied, bytes: 60000000

By default we use 57Mb is a standard setting for Firebird installation. If the size of your Firebird is larger, please consider clean-up of old logs and other unwanted artifacts, or increase parameter **Max occupied** (in bytes) to prevent false alerts.

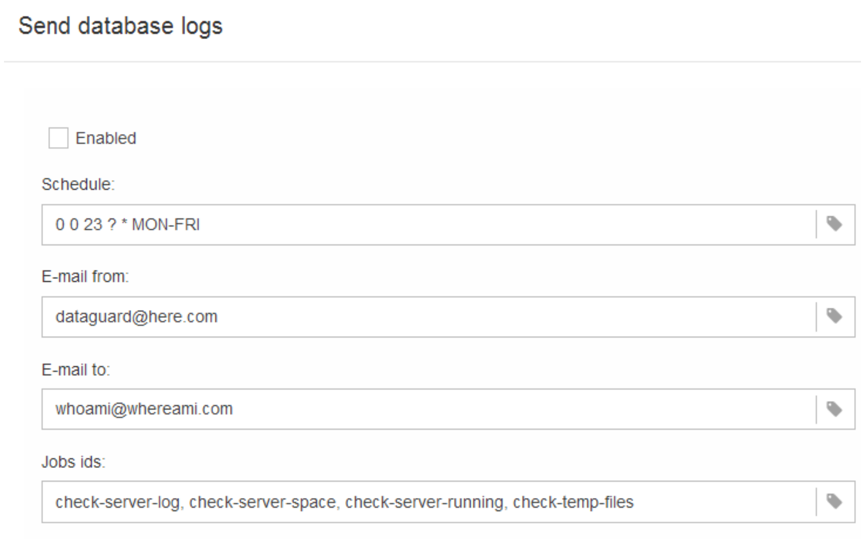
Note for Linux users: if you see red warning regarding the inconsistent space information, add locations with database and backups to Disk Space widget:


Space configuration

File System roots: /db, /backups

You can get idea where is your database and backup is actually located with command **df –h**.

Server: Send logs



Send database logs

- [ ] Enabled

Schedule: 0 0 23 ? * MON-FRI

E-mail from: dataguard@here.com

E-mail to: whoami@whereami.com

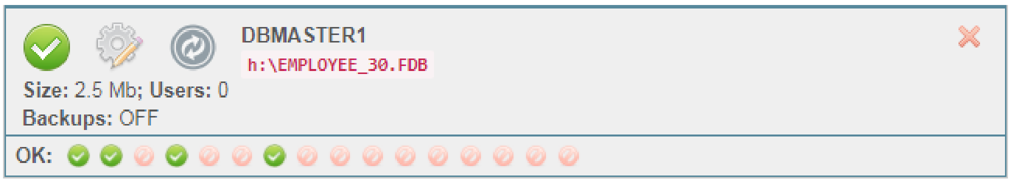
Jobs ids: check-server-log, check-server-space, check-server-running, check-temp-files

"Send logs" is an auxiliary job which sends logs from server-level jobs by email with desired frequency. This job is disabled by default. If monitored database is situated at remote location it's useful to schedule logs sending by email. Using configuration dialog you can schedule logs sending with CRON expression (for more details see ***Appendix: CRON Expressions***), specify from what email it will be sent and where. Setting from email alerts settings will be used (for details see Email alerts in HQbird FBDataGuard).

Also you can specify logs to be sent by specifying appropriate Jobs IDs.

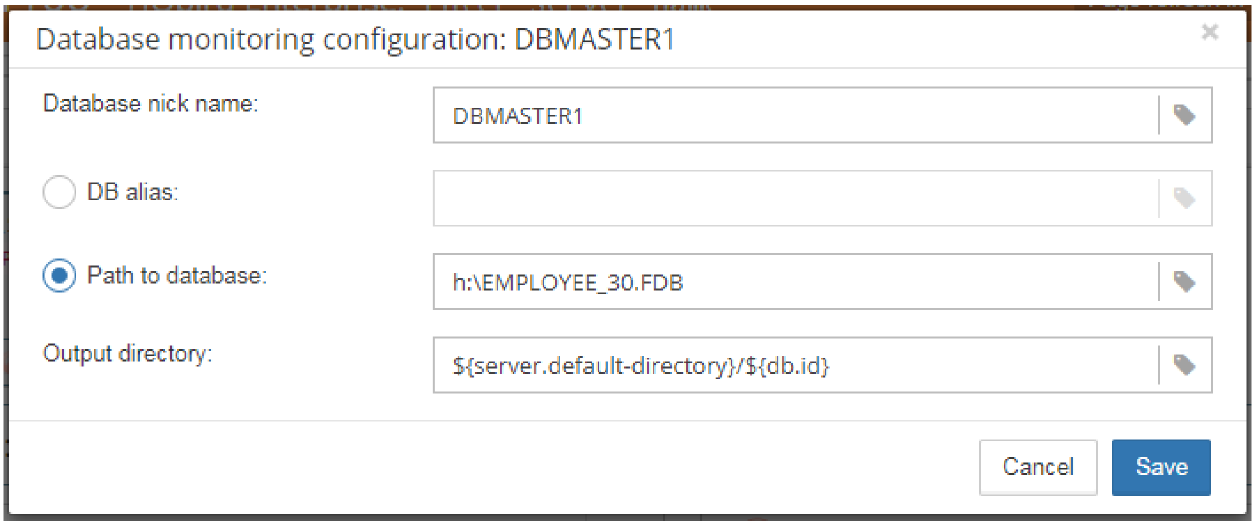
Database: General configuration

FBDataGuard can monitor several databases at the single server (up to 80 databases). For each database the separate widget is created. At the top widget database status is shown, database nickname (it's specified during database adding and can be changed). Also database widget shows the full path to the database, its size, status of backups and the number of currently connected users.



DBMASTER1
h:\EMPLOYEE_30.FDB
Size: 2.5 Mb; Users: 0
Backups: OFF
OK:

Using configuration dialog you can set database nickname, path to database and output folder for the database (to store logs and jobs results).



Database monitoring configuration: DBMASTER1

Database nick name:	DBMASTER1
DB alias:	
[✓ Path to database:]	h:\EMPLOYEE_30.FDB
Output directory:	${server.default-directory}/${db.id}

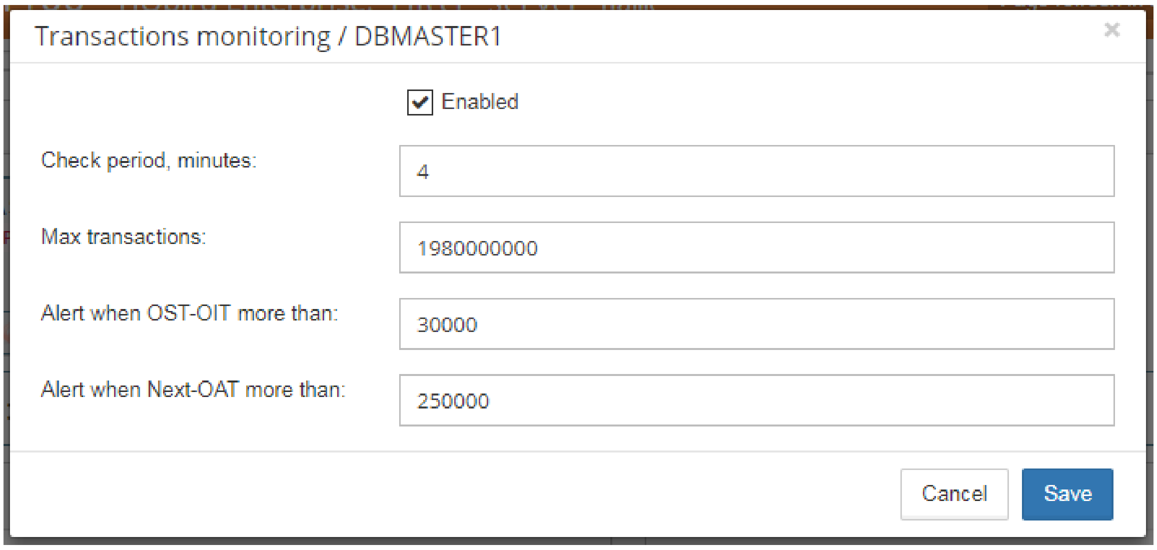
FBDataGuard checks the validity of path to database and it does not allow specifying the wrong path.

Also, for HQbird Enterprise, in the database widget you can see status of the replication and configure replication by clicking on the icon. Please read more details in the replication configuration section.

Database: Transactions

"Database: Transactions" job is intended to log transactions activity. It monitors 2 important intervals: difference between Oldest Active Transaction and Next transaction and gap between Oldest Snapshot and Oldest Interesting.

If these intervals are out of the frames of the specified threshold, it means problem with transactions management.

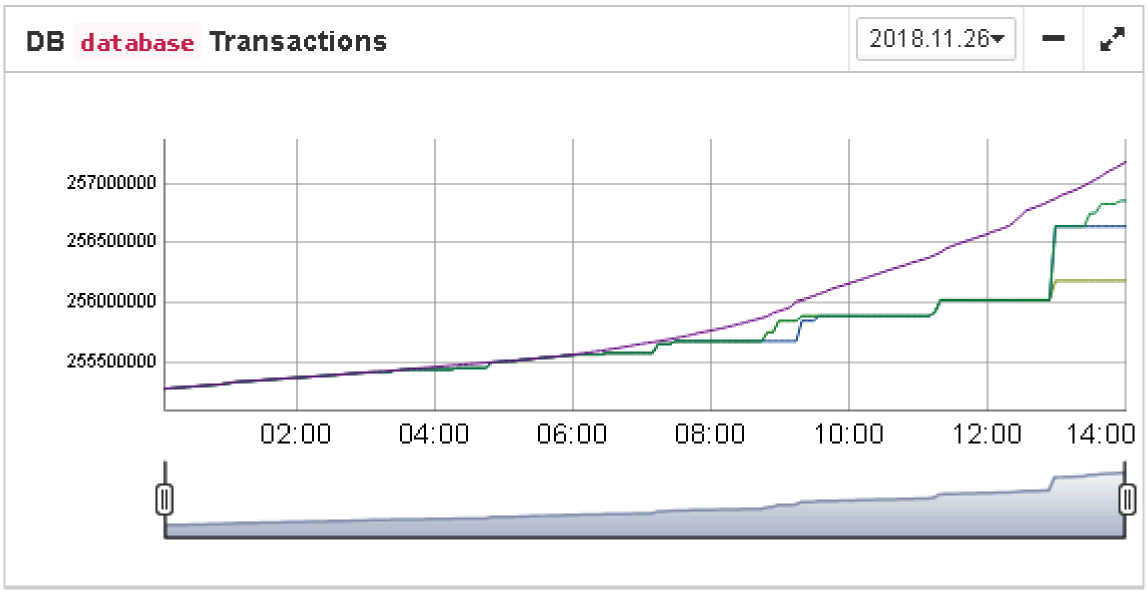

Transactions monitoring / DBMASTER1

☑ Enabled

Check period, minutes:	4
Max transactions:	1980000000
Alert when OST-OIT more than:	30000
Alert when Next-OAT more than:	250000

These logs can be analyzed to get helpful insight regarding database performance and application quality (see more information here http://ib-aid.com/en/articles/ibanalyst-what-you-can-see-at-summary-view/).

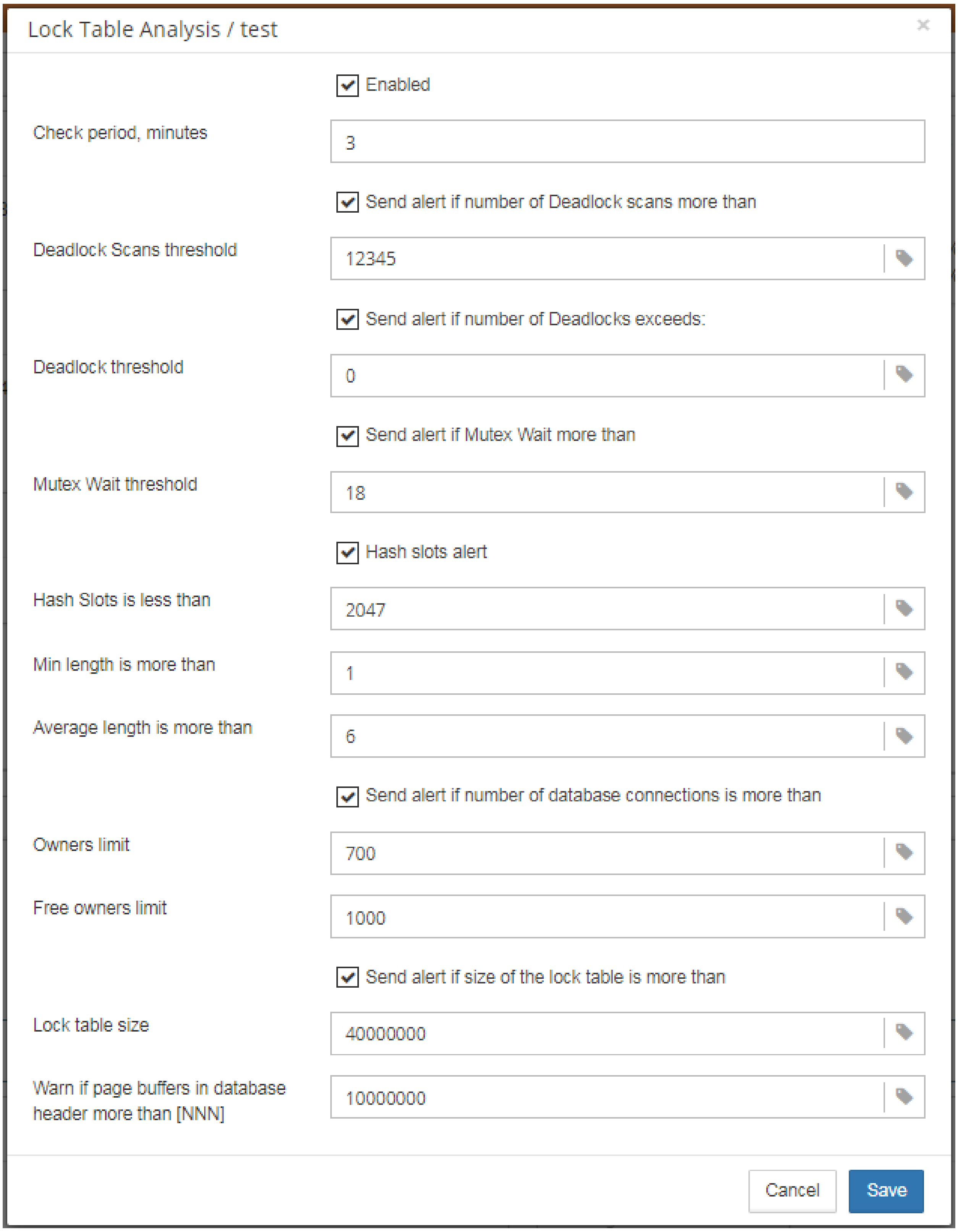
This job also monitors the implementation limit in Firebird: maximum transactions number in Firebird versions before 3.0 should be less than 2^32-1. Near this number database should be backup and restored. It will throw an alert if transaction number will be close to the restrictions. Also, the transaction dynamics is shown on the tab «Graphs gallery»:



Database: Lockprint

«Lockprint» job monitors the information from the lock table of Firebird. It is very important for architectures Classic/SuperClassic and useful for SuperServer.

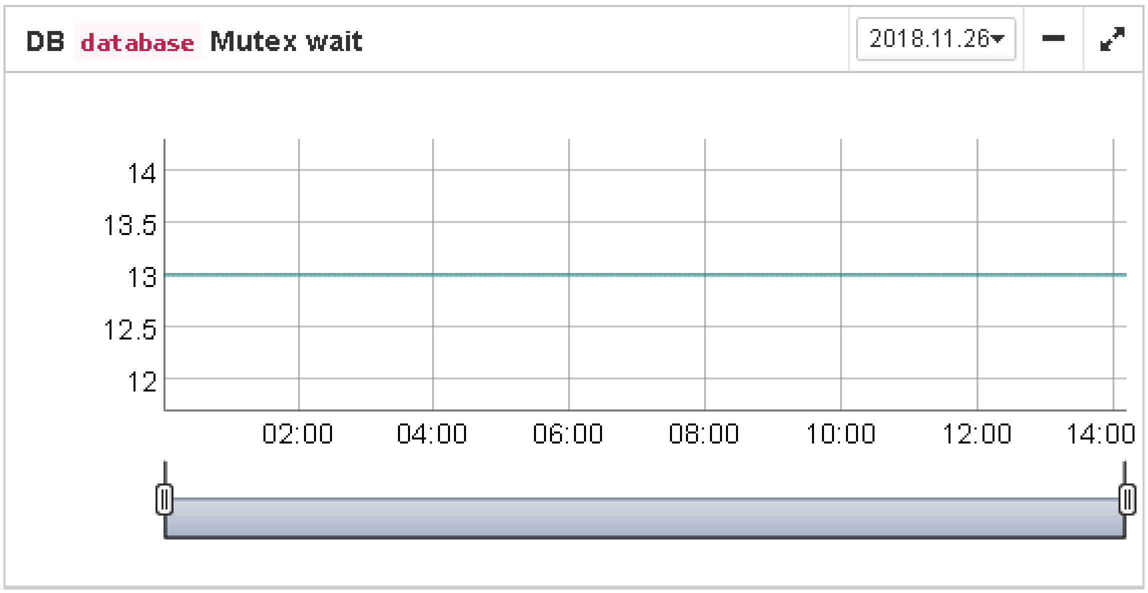
The lock table is an internal Firebird mechanism to organize access to the objects inside Firebird engine. HQbird monitors the important parameters of the lock table:



Lock Table Analysis / test

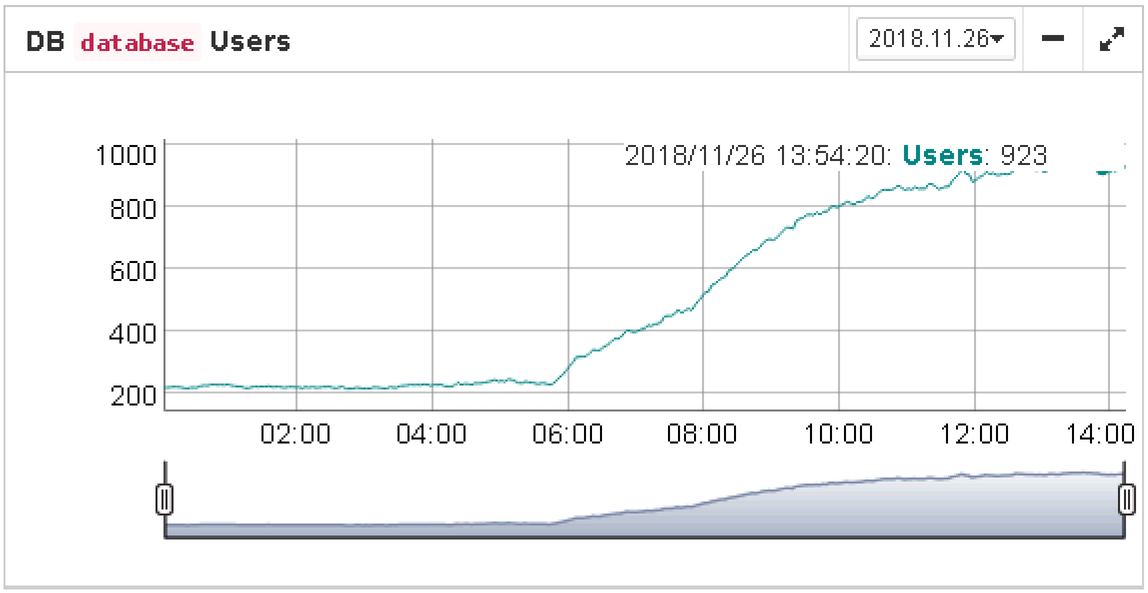
	✓ Enabled
Check period, minutes	3
	✓ Send alert if number of Deadlock scans more than
Deadlock Scans threshold	12345
	✓ Send alert if number of Deadlocks exceeds:
Deadlock threshold	0
	✓ Send alert if Mutex Wait more than
Mutex Wait threshold	18
	✓ Hash slots alert
Hash Slots is less than	2047
Min length is more than	1
Average length is more than	6
	✓ Send alert if number of database connections is more than
Owners limit	700
Free owners limit	1000
	✓ Send alert if size of the lock table is more than
Lock table size	40000000
Warn if page buffers in database header more than [NNN]	10000000

- **Check period**, minutes –how often HQbird analyses lock table. 3 minutes is an optimal interval
- **Deadlock Scans threshold** – deadlock scan is a process, started by Firebird engine, in case of a long response delay from the one of the threads. If a number of deadlock scans is high, it means that Firebird is heavily loaded. The value is accumulated since the Firebird engine start. The default value is pretty big – 12345, so if it is exceeded, it means that database performance is poor.
- **Deadlock threshold** – if Firebird engine finds the true deadlock during the deadlock scans, it increases this value. Please note: true deadlocks are very seldom. Don't confuse them with transactions' conflicts («deadlock. Lock conflict on nowait transaction» etc).
- **Mutex wait threshold**–Mutex Wait is a parameter of lock table which implicitly indicates the conflicts for the resources. The higher mutex wait, the more competition exists inside the engine for the resources. By default, the mutex wait threshold is set to 18%, but this value is not universal for all databases. The good approach is to watch for the mutex values during 1-2 weeks and then set the highest value seen during this period. Mutex wait graph is available in Mutex Wait gallery.

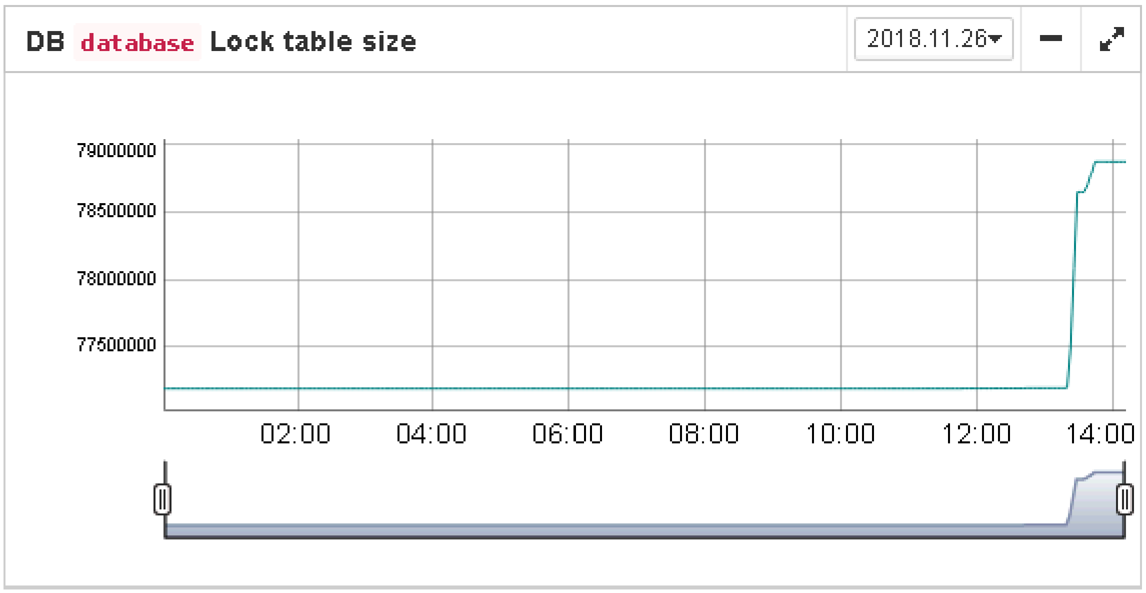


- **Hash slots alerts**. Lock table header has a parameter «Hash lengths (min/avg/max): 0/ 0/4», it shows the lengths in the lock table. It is important to keep these values as low as possible, so HQbird monitors them and suggest, how to improve the situation, if hash length is more than specified in this job.
- **Owners limit.**«Owners» is a number of connections established to the specified database. In fact, this is the fastest way to get the actual number of connections to the database with the minimum load to the database – other ways like request to MON$ATTACHMENTS or isc_tpb_database have various disadvantages. The limit here should be set according the actual peak number of connections. For example, if you are sure that peak number of the connections to your database is 500, set 550 as Owners

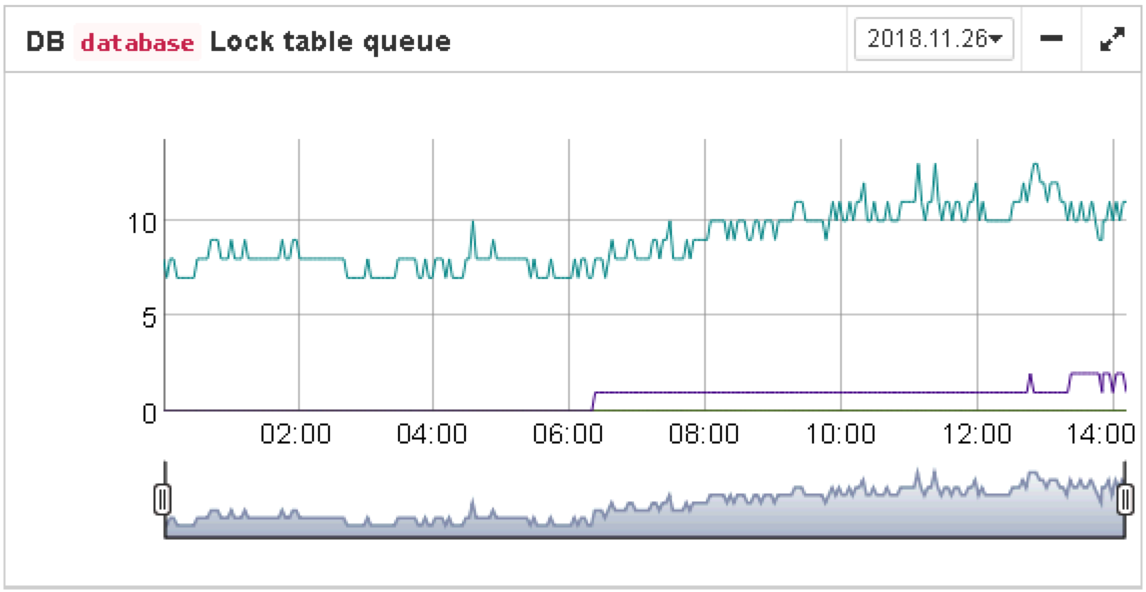
limit, and if at some moment the load will increase, you will not miss that moment.



- **Free owners**. «Free owners» is the value between the peak number of owners and current number of owners. If you see Free owners = 0, it means that number of connections grows steadily since the Firebird start. If you see high number of Free owners, it can be sign that many connections were disconnected recently.
- Lock table size. The lock table size is an implicit indicator of the load to the system. Normally, lock table size should be stable. Also, it is recommended to set the initial lock table size to the value it has after some active work period – though the lock table is enlarged on demand, the re-allocation process is a heavy operation and can lead to micro-freezes in database responses. Lock table graph is useful to determine the proper initial value.



- Lock table queue does not have the explicit threshold in Lockprint job, but its values are collected and shown in «Graphs gallery». Lock table queue is an indicator of a general load.



Database: Index statistics

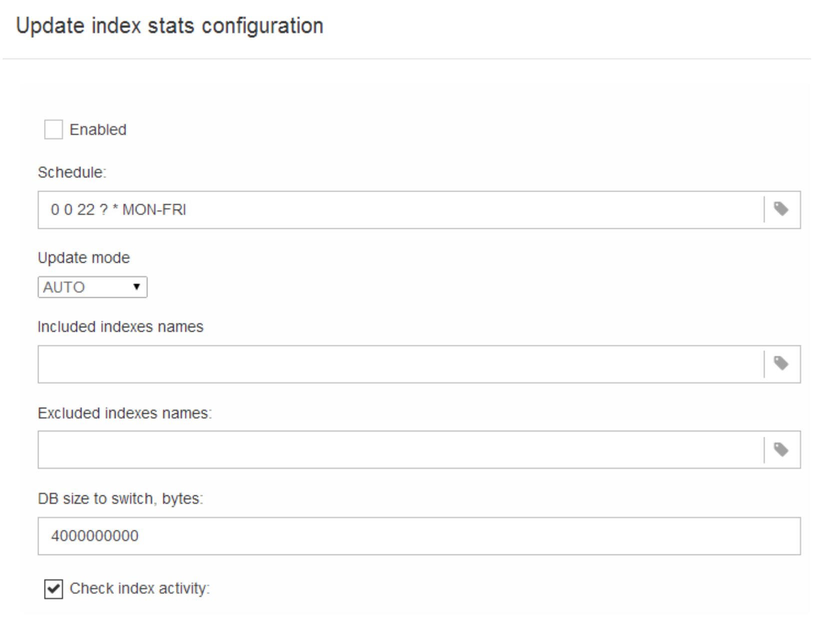
“Database: Index statistics” is an important job which helps to keep performance of indices at optimal level, and performs additional checking of a database health.

“Database: index statistics” allows to run re-computing of indices selectivity values. During this procedure Firebird quickly walks through leaf pages of indices, and renews statistics about selectivity. By visiting these pages Firebird also verifies their integrity and if index is corrupted, the warning will be thrown.

Also, this job verifies that all indices are active in database. Inactive or non-activated indices usually indicate corruption and lead to performance degradation.

By default this job is disabled, but we recommend enabling it after careful selecting of indices for the recalculation.

There are three modes in this job: AUTO, ALL, SELECTED. ALL is the mode where all indices will be checked. AUTO is the default mode. It is very similar to ALL, but it also checks the size of database and do not touch indices if database is bigger than 3.6Gb. SELECTED is the recommended mode. It allows choosing of indices which should be recomputed or those which should be avoided.



Update index stats configuration

☐ Enabled

Schedule: 0 0 22 ? * MON-FRI

Update mode: AUTO

Included indexes names:

Excluded indexes names:

DB size to switch, bytes: 4000000000

☑ Check index activity:

To include indices into the list of recomputed, you need to specify indices names (divided by comma), and to exclude – perform the same in the appropriate field.

As you can see at configuration dialog screenshot, there are fields to enable/disable job, to set update mode, and to include or exclude indices. “DB size to switch, bytes” is to set limit where AUTO mode is working. “Check index activity” switch should be always on, until you are not performing special manipulations with inactive indices.

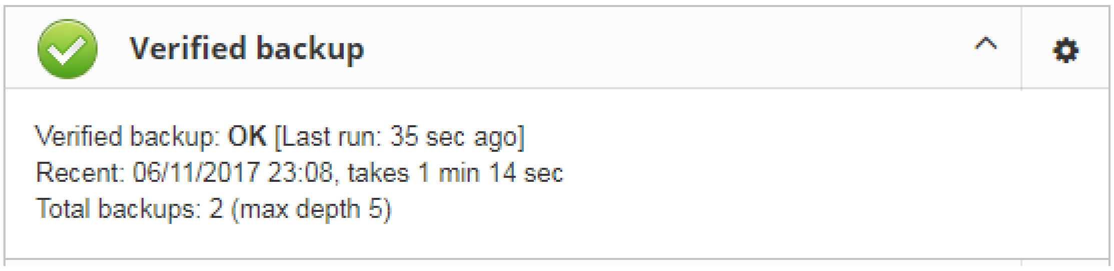
Database: Verified Backup

"Database: Verified Backup" is one of the key jobs to guarantee the safety of data stored in the protected database. During the development of HQbird we had in mind certain recovery scenario, and this scenario implies that the key goal of database protection is to minimize potential losses of data. If we have healthy backup, recovery can be concentrated on saving the most recent data (just entered into the database), and it greatly decreases the time of overall outage.

As you will see below, "Database: Verified Backup" is not just a wrapper for standard gbak functionality and scheduler, this is a smart job which has many built-in rules to prevent problems with backups and provide suitable interface for backups management.

"Database: Verified Backup" is disabled by default, but we strongly recommend reviewing of its settings immediately after HQbird setup.

Initially "Database: Verified Backup" job is shown as Ok, though backup was not tried. In this case OK means that backup at least scheduled.
Also this job recognizes files according the name pattern (see below information regarding configuration),and shows the totals number of backups.
After the backup will be done, the widget information will be changed: creation time of last successful backup will be shown, and also the time took to actually perform the backup (only 1 minute 14 seconds at the screenshot with example).



Verified backup: **OK** [Last run: 35 sec ago]
Recent: 06/11/2017 23:08, takes 1 min 14 sec
Total backups: 2 (max depth 5)

"Database: Verified Backup" checks the free space at the drive with backup destination, and if it detects that there is not enough free disk space, CRITICAL alert will be sent, and current backup will be canceled (if necessary).

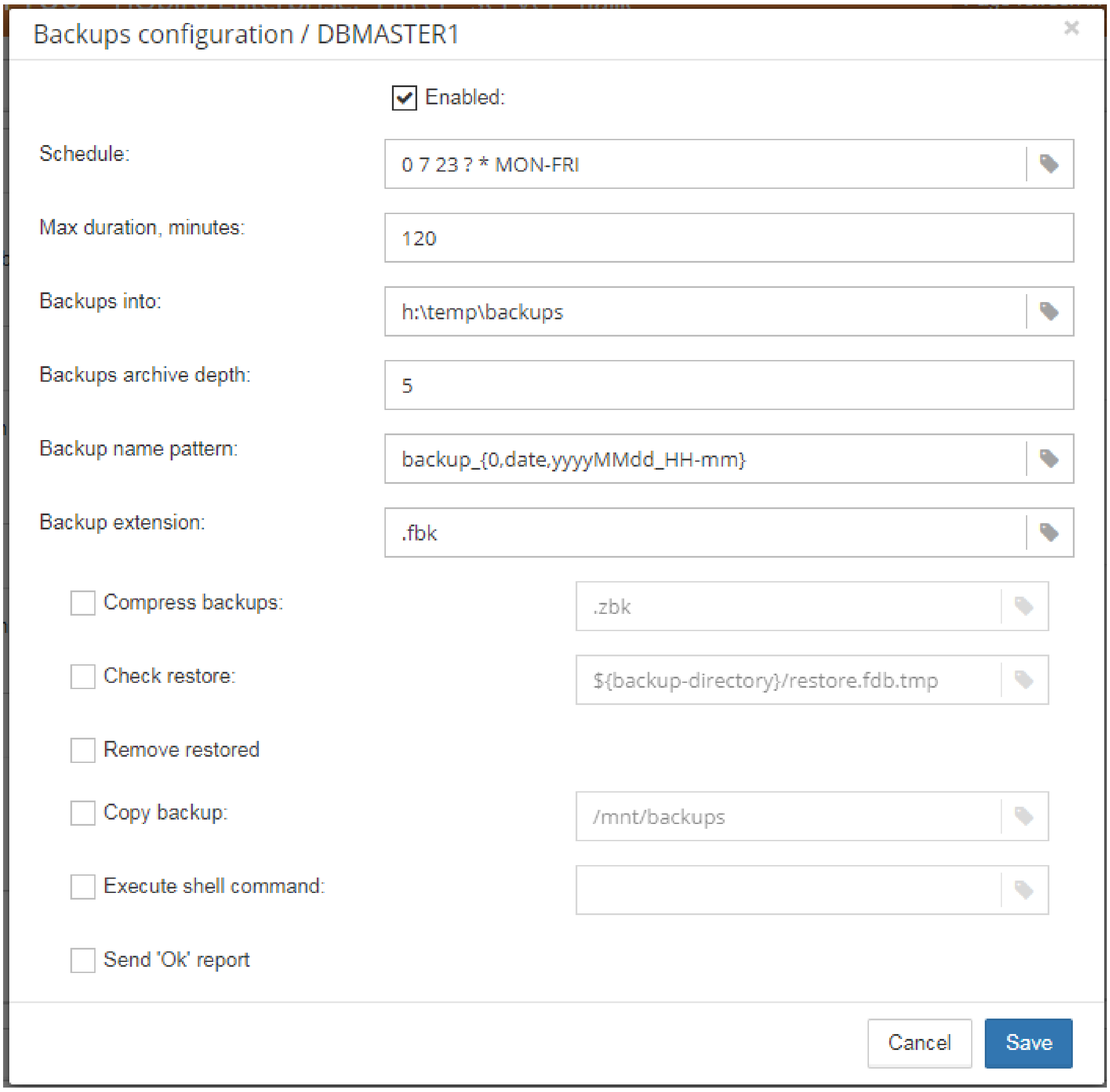
Be careful – by default backup time is set to 23-00 Monday-Friday.

By default database backups will be stored into the folder that you have specified during installation step!

It is very important to carefully review default database backups settings and adjust them according the local configuration!

- At configuration dialog (see below) in the "Schedule" field you can set the time when backup should be run. Scheduler uses CRON expression and this is a right place to apply all the power of CRON (see *Appendix: CRON Expressions*).
- "**Max duration**" time is a time frame to complete all backup steps. Sometimes backup process can hang up, so exceeding of the maximum time will immediately notify administrator of some problems.

Also, if backup took too much time, it can indicate problems with garbage collection or abnormal growth of database itself.



Backups configuration / DBMASTER1

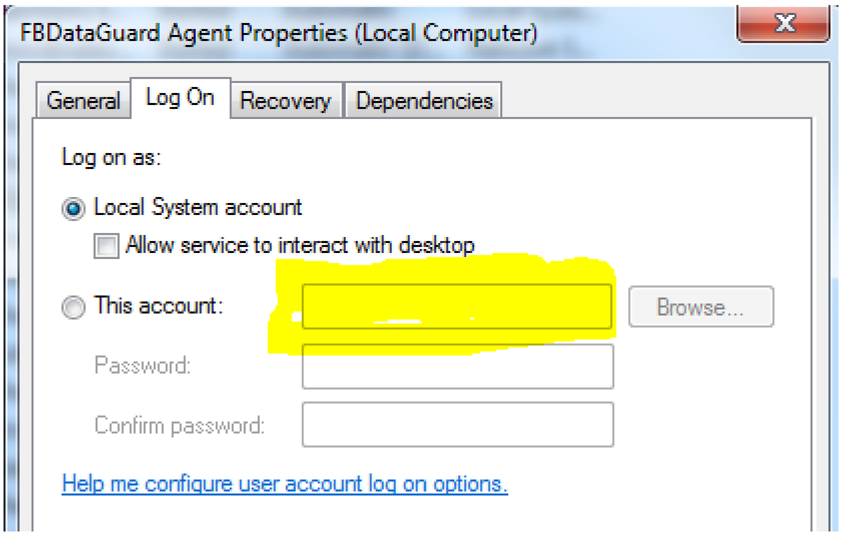
☑ Enabled:	
Schedule:	0 7 23 ? * MON-FRI
Max duration, minutes:	120
Backups into:	h:\temp\backups
Backups archive depth:	5
Backup name pattern:	backup_{0,date,yyyyMMdd_HH-mm}
Backup extension:	.fbk
☐ Compress backups:	.zbk
☐ Check restore:	${backup-directory}/restore.fdb.tmp
☐ Remove restored	
☐ Copy backup:	/mnt/backups
☐ Execute shell command:	
☐ Send 'Ok' report	

- “**Backups into**” specifies the folder to store backups. This folder should be at the same computer where database is. By default it is situated inside database default directory. Usually it’s a good idea to set the explicit path to the folders with backups.
- “**Backups archive depth**” specifies how many previous backups should be stored. FBDataGuard stores backups in revolver order: when the archive depth will be reached (i.e., 5 backups will be created), FBDataGuard will delete the oldest backup and create the new backup. In combination with CRON expressions it gives a powerful ability to create necessary history of backups.
- “**Backup name pattern**” specifies how backup files will be named. Also this name pattern allows FBDataGuard to recognize old backups with the same name pattern.
- “**Backup extension**” is fbk by default.

- "**Compress backups**" specifies should FBDataGuard archive backups after regular Firebird backup. By default this option is on, but you need to know that FBDataGuard will zip backups' files which are less than **100 Gb** in size. After that size, the backup compression will be automatically switched off. We recommend to turn this feature on for small databases only.
- "**Check restore**" is an important option. If it is on (by default), FBDataGuard will perform test restore of fresh backup, in order to test its validity. It guarantees the quality of created backup and notifies administrator in case of any problems with test restore.
- "**Restore to**" specifies the folder where to perform test restore. By default it's inside database output folder. It's a good idea to set the explicit path for test restore.
- "**Remove restored**" specifies should FBDataGuard delete restored database. ***By default it is OFF,*** so you might want to turn it ON, but you need carefully consider – do you really need to keep the copy of test restored database. With each test restore this copy will be overwritten.
- "**Copy backup**" switch and "**Copy backup to**" path. If you have network location or plugged USB drive to store database where you want to store copy of backup (in addition to usual backups), FBDataGuard can copy the latest backup there: just turn on "Copy backup" switch and specify "**Copy backup to**" path.
- "**Execute shell command**" switch and "**Shell command**" path. It is possible to specify custom script or executable after the general backup procedure will be complete. Shell command gets as the path to the fresh database backup as a parameter.
- "**Send "Ok" report**" – by default it is off, but it's strongly recommended to turn it ON and start to receive notifications about correct backups. This feature will use email settings from alerts system (see Email alerts in HQbird FBDataGuard).

Important Note: Backup to the network locations

Please be aware that for creating and copying backup to the network locations Firebird and FBDataGuard services must be started under the account with sufficient rights. By default, Firebird and FBDataGuard are started under LocalSystem account, which does not have rights to access network location.


FBDataGuard Agent Properties (Local Computer)

General | Log On | Recovery | Dependencies

Log on as:

(●) Local System account

[ ] Allow service to interact with desktop

( ) This account: Browse...

Password:

Confirm password:

Help me configure user account log on options.

So, to store Firebird backups to the network location on Windows, run Services applet (services.msc) and on the tab Log On change «Log on as» to the appropriate account (Domain Admin should be fine).

For Linux – add necessary rights for «firebird» user.

Database: Incremental Backup

Incremental backup is a job to schedule and manage incremental backups in Firebird.

Please note that we recommend to use incremental backups only in combination with verified backups, since incremental backup performs coping of database pages changed since the last backup (in case of multilevel incremental backup).

HQbird FBDataGuard implements 2 types of multilevel incremental backup: Simple and Advanced incremental backups, and also Dump backup (see Database: Dump backup).

Multilevel backup in Firebird must follow the following steps:

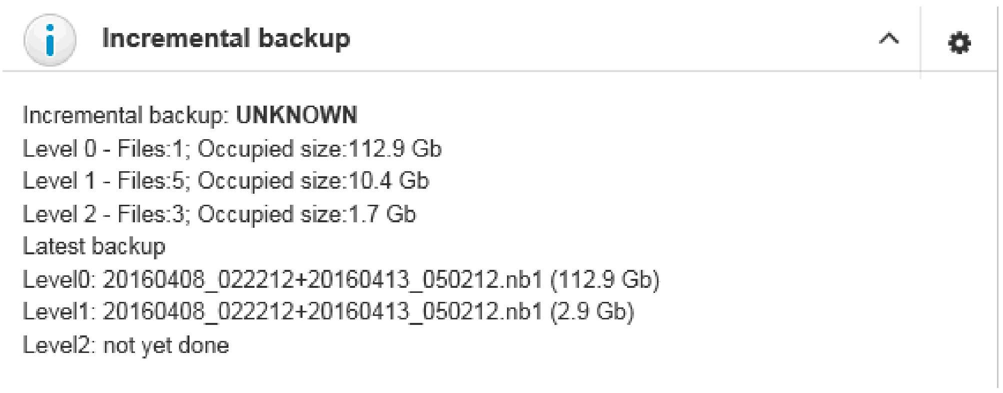
1) Create initial backup (level 0) which essentially is the copy of the database at the moment of backup start and mark it with backup GUID.

2) Since Firebird marks each data page with a certain identifier at every change, it is possible to find data pages, changed from the moment of previous backup and copy only them to form backup of level 1.

3) It is possible to create several level of the backups – for example, the initial backup (full copy, level 0) is being created every week, every day we create level 1 (differences from the level 0), and at every hour we create level 2 backups (differences from daily level 1).

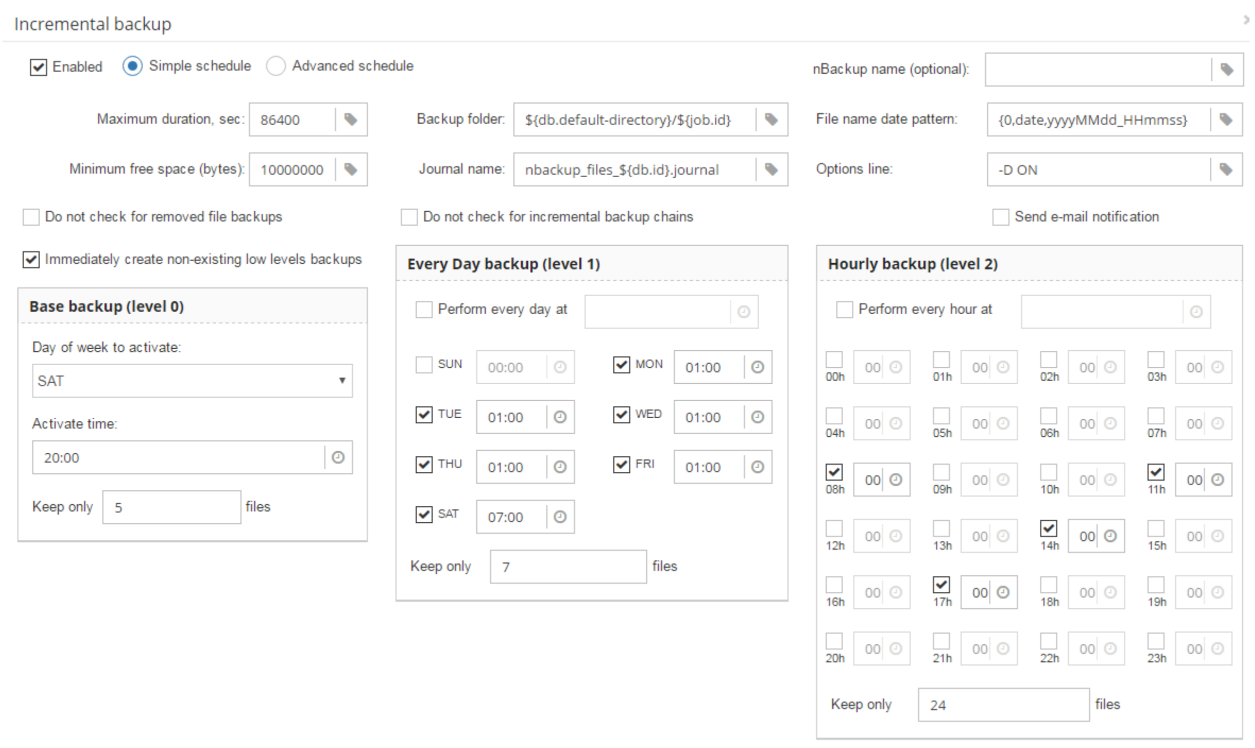
Incremental backup with simple schedule allows planning 3 levels of backups: weekly, daily and hourly.

You can see summary information for such incremental backup configuration at the following screenshot of its widget:



Incremental backup: **UNKNOWN**
Level 0 - Files:1; Occupied size:112.9 Gb
Level 1 - Files:5; Occupied size:10.4 Gb
Level 2 - Files:3; Occupied size:1.7 Gb
Latest backup
Level0: 20160408_022212+20160413_050212.nb1 (112.9 Gb)
Level1: 20160408_022212+20160413_050212.nb1 (2.9 Gb)
Level2: not yet done

In order to setup Simple incremental backup, click on Settings «gear» of the widget and select «Simple schedule» (selected by default). The following dialog will appear:


Incremental backup

☑ Enabled ● Simple schedule ○ Advanced schedule

nBackup name (optional):

Maximum duration, sec: 86400

Backup folder: ${db.default-directory}/${job.id}

File name date pattern: {0,date,yyyyMMdd_HHmmss}

Minimum free space (bytes): 10000000

Journal name: nbackup_files_${db.id}.journal

Options line: -D ON

☐ Do not check for removed file backups

☐ Do not check for incremental backup chains

☐ Send e-mail notification

☑ Immediately create non-existing low levels backups

Base backup (level 0)

Day of week to activate: SAT

Activate time: 20:00

Keep only 5 files

Every Day backup (level 1)

☐ Perform every day at

☐ SUN	00:00	☑ MON	01:00
☑ TUE	01:00	☑ WED	01:00
☑ THU	01:00	☑ FRI	01:00
☑ SAT	07:00		

Keep only 7 files

Hourly backup (level 2)

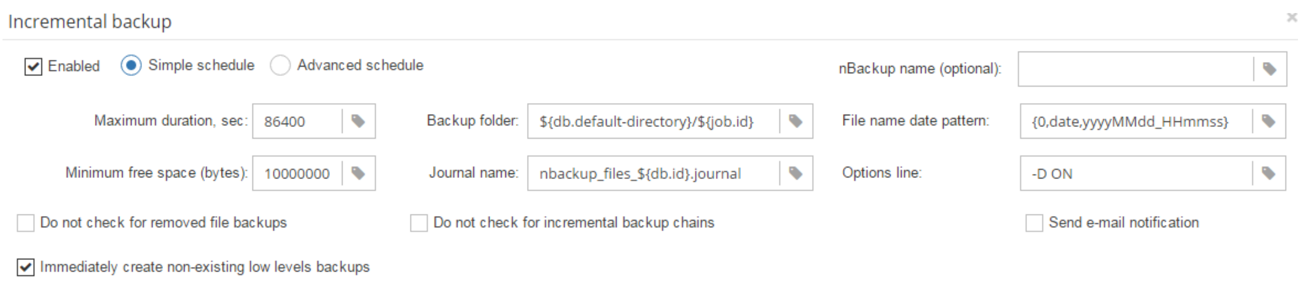
☐ Perform every hour at

☐ 00h	00	☐ 01h	00	☐ 02h	00	☐ 03h	00
☐ 04h	00	☐ 05h	00	☐ 06h	00	☐ 07h	00
☑ 08h	00	☐ 09h	00	☐ 10h	00	☑ 11h	00
☐ 12h	00	☐ 13h	00	☑ 14h	00	☐ 15h	00
☐ 16h	00	☑ 17h	00	☐ 18h	00	☐ 19h	00
☐ 20h	00	☐ 21h	00	☐ 22h	00	☐ 23h	00

Keep only 24 files

There are 4 main areas in this dialog, let's cover them one by one.

The top area is devoted for general settings of the incremental backup – they are the same for Simple and Advanced schedules:


Incremental backup

☑ Enabled ● Simple schedule ○ Advanced schedule

nBackup name (optional):

Maximum duration, sec: 86400

Backup folder: ${db.default-directory}/${job.id}

File name date pattern: {0,date,yyyyMMdd_HHmmss}

Minimum free space (bytes): 10000000

Journal name: nbackup_files_${db.id}.journal

Options line: -D ON

☐ Do not check for removed file backups

☐ Do not check for incremental backup chains

☐ Send e-mail notification

☑ Immediately create non-existing low levels backups

Max duration, sec – it limits the maximum duration of backup process, by default is 1 day (86400 seconds).

Minimum free disk space (bytes) – minimal size of free disk space to prevent backup to start, by default ~9Mb

Backup folder – where incremental backup for the selected database will be stored. It is necessary to store incremental backups for each database separately from backups of other databases: i.e., the separate folder for each database.

It is necessary to specify backup folder with enough free disk space to store all level of backups!

Journal name–file name details information about incremental backups files, for internal use only.

nBackup name – it is possible to specify other nbackup tool than standard nbackup.exe (not recommended).

File name date pattern – pattern for files of incremental backup (no need to change it).

Options line – additional options for nbackup command line tool (no need to change it).

Do not check for removed file backups – this option should be checked if you plan to delete or more incremental backups to another location.

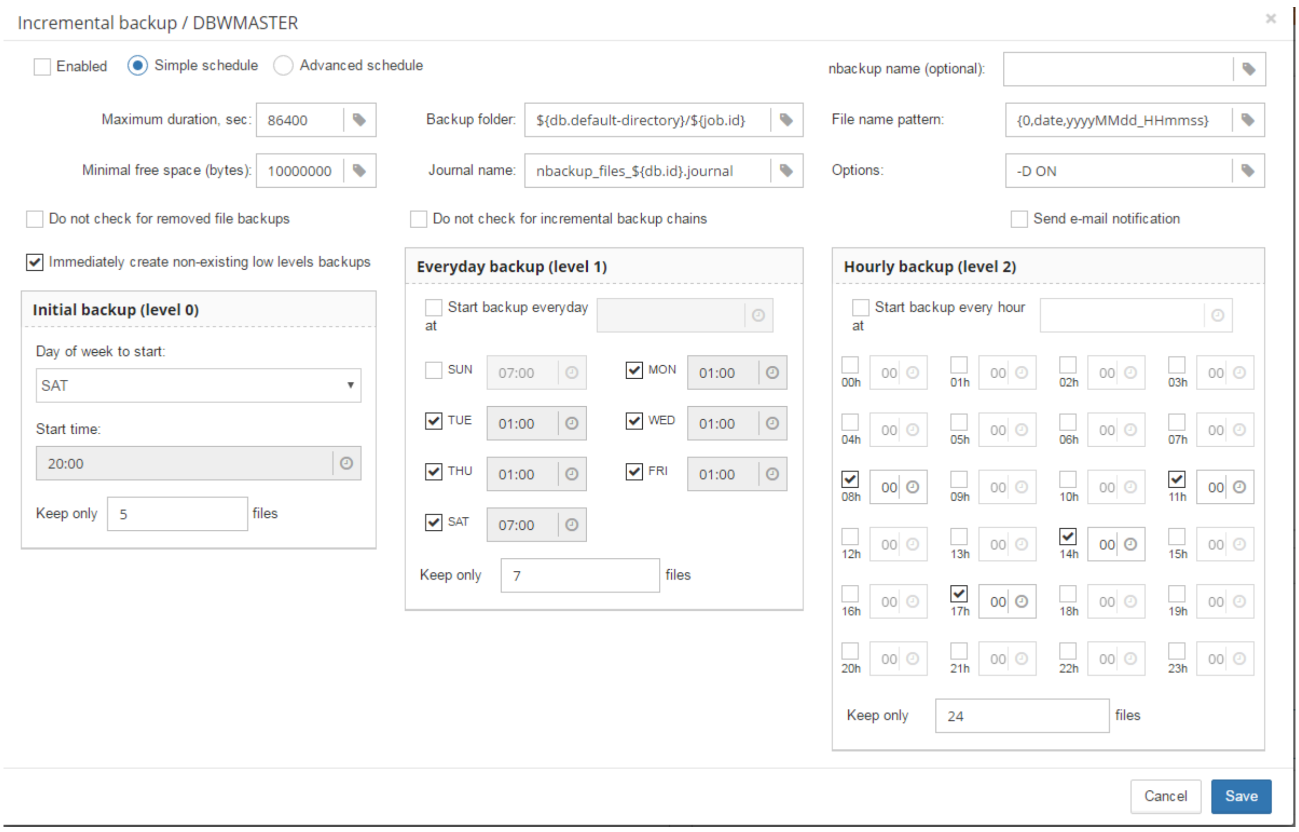
Do not check for incremental backup chains – this option should be checked if you want to skip existence check of previous levels of incremental backups.

Immediately create non-existing low-level backups – by default this option is On. It means that if you have scheduled the initial start moment of level 1 backup earlier than the initial start moment of level 0 backup, DataGuard will automatically fix it and create level 0 backup right before level 1. The following backups of level 0 will be fulfilled according the regular schedule.

Send email notifications – enable this option to receive notifications about incremental backups (highly recommended!)

After setting main set of parameters the schedule itself should be set. As you can see on the screenshot below, you need to specify day of the week and time to do level 0 (weekly) backup, days of week and time to start level 1 (daily) backups and hours and minutes of level 3 (hourly backups).

For each backup level you can specify how many files to keep in history.



Incremental backup / DBWMASTER

☐ Enabled [✓ Simple schedule] Advanced schedule

Maximum duration, sec:	86400
Minimal free space (bytes):	10000000
Backup folder:	${db.default-directory}/${job.id}
Journal name:	nbackup_files_${db.id}.journal
nbackup name (optional):	
File name pattern:	{0,date,yyyyMMdd_HHmmss}
Options:	-D ON

☐ Do not check for removed file backups
☐ Do not check for incremental backup chains
☐ Send e-mail notification
☑ Immediately create non-existing low levels backups

Initial backup (level 0)

Day of week to start: SAT
Start time: 20:00
Keep only 5 files

Everyday backup (level 1)

☐ Start backup everyday at

☐	SUN	07:00
☑	MON	01:00
☑	TUE	01:00
☑	WED	01:00
☑	THU	01:00
☑	FRI	01:00
☑	SAT	07:00

Keep only 7 files

Hourly backup (level 2)

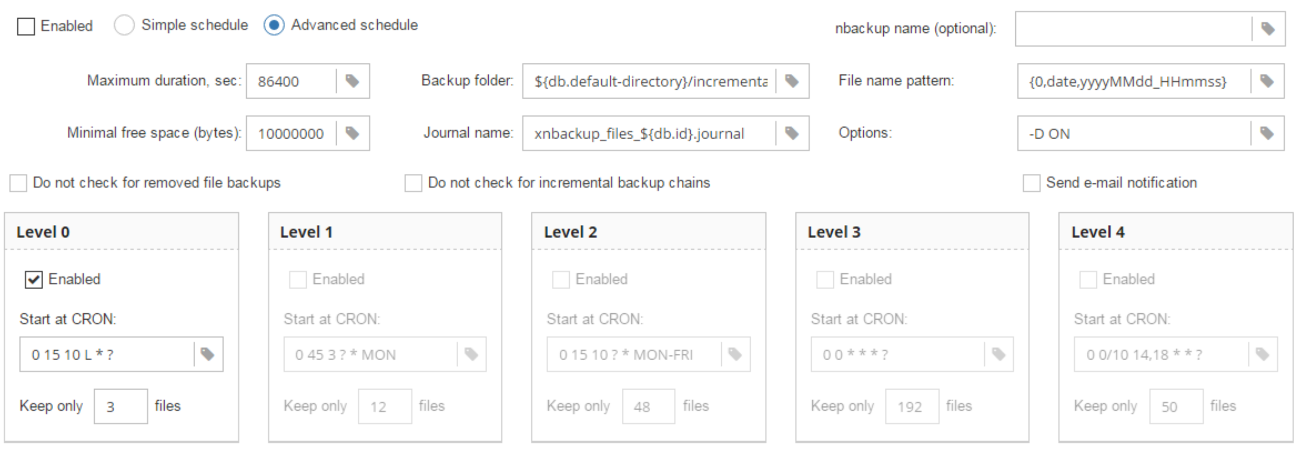
☐ Start backup every hour at

☐ 00h 00	☐ 01h 00	☐ 02h 00	☐ 03h 00
☐ 04h 00	☐ 05h 00	☐ 06h 00	☐ 07h 00
☑ 08h 00	☐ 09h 00	☐ 10h 00	☑ 11h 00
☐ 12h 00	☐ 13h 00	☑ 14h 00	☐ 15h 00
☐ 16h 00	☑ 17h 00	☐ 18h 00	☐ 19h 00
☐ 20h 00	☐ 21h 00	☐ 22h 00	☐ 23h 00

Keep only 24 files

By default it is set to keep 5 weekly backups, 7 daily and 24 hourly backups.

However, sometimes more flexible schedule is required, for this purpose Incremental Backup widget has Advanced schedule:



☐ Enabled ○ Simple schedule ● Advanced schedule

nbackup name (optional):

Maximum duration, sec: 86400

Backup folder: ${db.default-directory}/incrementa

File name pattern: {0,date,yyyyMMdd_HHmmss}

Minimal free space (bytes): 10000000

Journal name: xnbackup_files_${db.id}.journal

Options: -D ON

☐ Do not check for removed file backups

☐ Do not check for incremental backup chains

☐ Send e-mail notification

	Level 0	Level 1	Level 2	Level 3	Level 4
Enabled	✓	☐	☐	☐	☐
Start at CRON:	0 15 10 L * ?	0 45 3 ? * MON	0 15 10 ? * MON-FRI	0 0 * * * ?	0 0/10 14,18 * * ?
Keep only … files	3	12	48	192	50

As you can see, the upper part of the configuration screen is the same as in Simple schedule, and the difference is in the way how backup levels are scheduled.

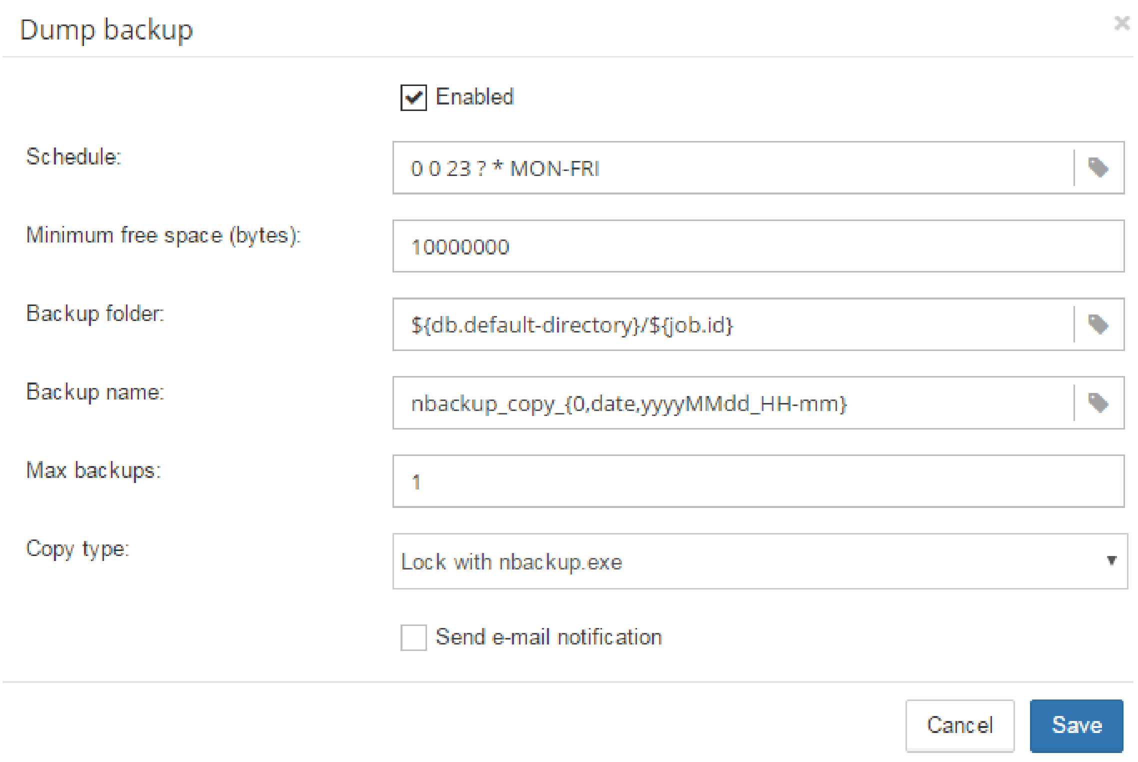
Advanced schedule allows to setup up to 5 levels of backup, and plan them with flexible CRON expressions.

For example, you can setup it to create full copy (level 0) backup every 3 months, level 1 copy every month, level 2 – every week, level 3 every day and level 4 – every hour.

Database: Dump Backup

This job also utilizes nbackup functionality in Firebird, but unlike multilevel backups, it always performs a full copy (level 0) of the database. Such job is useful to quickly create a copy of working database.

The configuration of Database: Dump backup is trivial:



Dump backup

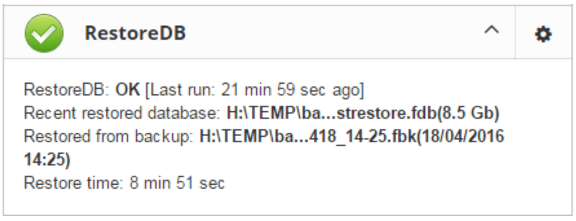
☑ Enabled

Schedule:	0 0 23 ? * MON-FRI
Minimum free space (bytes):	10000000
Backup folder:	${db.default-directory}/${job.id}
Backup name:	nbackup_copy_{0,date,yyyyMMdd_HH-mm}
Max backups:	1
Copy type:	Lock with nbackup.exe

☐ Send e-mail notification

You just need to setup when and where DataGuard should copy a full copy (level 0 incremental backup), and how many copies it should keep.

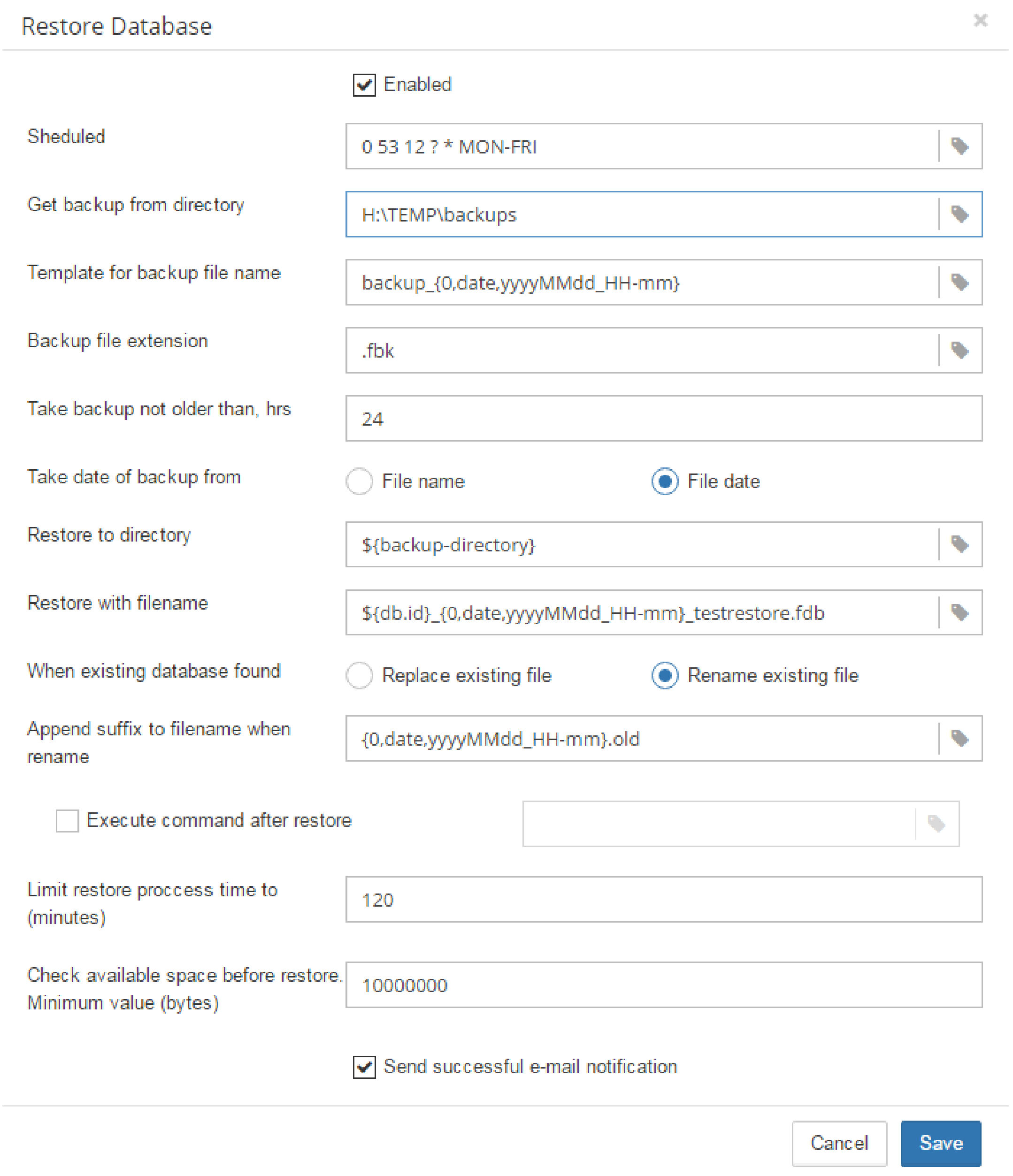
Database: RestoreDB


RestoreDB

RestoreDB: **OK** [Last run: 21 min 59 sec ago]
Recent restored database: **H:\TEMP\ba...strestore.fdb(8.5 Gb)**
Restored from backup: **H:\TEMP\ba...418_14-25.fbk(18/04/2016 14:25)**
Restore time: 8 min 51 sec

One of the often tasks of the database administrators is restoring database from the backup. There could be many reason to do restore, the most popular reasons are regular check of the stored backups and necessity to have fresh restored copy for quick rollback. HQbird FBDataGuard can automate restoring of backups (which were created with gbak or Database: Verified backup) with Database: RestoreDB job. Let's consider the options and parameters of this job.

By default restore is disabled – and, since restoring can be long and resource-consuming job, please plan when to restore carefully. Below you can see the configuration dialog for Database: RestoreDB:


Restore Database

	[✓] Enabled
Sheduled	0 53 12 ? * MON-FRI
Get backup from directory	H:\TEMP\backups
Template for backup file name	backup_{0,date,yyyyMMdd_HH-mm}
Backup file extension	.fbk
Take backup not older than, hrs	24
Take date of backup from	File name [✓ File date]
Restore to directory	${backup-directory}
Restore with filename	${db.id}_{0,date,yyyyMMdd_HH-mm}_testrestore.fdb
When existing database found	Replace existing file [✓ Rename existing file]
Append suffix to filename when rename	{0,date,yyyyMMdd_HH-mm}.old
[ ] Execute command after restore	
Limit restore proccess time to (minutes)	120
Check available space before restore. Minimum value (bytes)	10000000
	[✓] Send successful e-mail notification

«Scheduled» field contains CRON expression which defines when to run restore.

- «Get backup from directory» - here you need to specify the location of backup files. If you are restoring backups at the same computer where they have been created, specify the same folder as it is in Database: Verified backup job. If you are restoring backups from the another computer, specify the folder where those backups are located.
- «Template for backup file name»contains the template for backup names. It should be the same as for the backup.
- «Backup for file extension» - by default it is fbk
- «Take backup not older than, hrs» - this parameter specifies the maximum age of backup to be restored. If the latest backup file will be older than specified number of hours, RestoreDB job will send the alert with warning that backup is too old. This is useful for automatic checking of backups created on the remote computer.
- «Restore to directory» - folder where FBDataGuard will restore backups.
- «Restore with filename» - template for the restored database file. By default it contains the following pieces
    - ${db.id}_{0,date,yyyyMMdd_HH-mm}_testrestore.fdb
    - Db.id – internal identifier of the database (GUID)
    - 0,date,yyyyMMdd_HH-mm – timestamp
    - testrestore.fdb – description (You can set there any filename you need).
- «When existing database found» - if FBDataGuard will encounter a file with the same name as restored database in the destination folder, by default it will rename the existing file. If you want to replace old restored file with new one, choose «Replace existing file».
- «Append suffix to filename when rename» - if you have chosen «Rename existing file», this suffix will be used to rename it. If you have chosen «Replace existing file», this suffix also will be used to rename, but after that the old file will be deleted.
- «Execute command after restore» - in this field you can specify an optional path to the command file or another utility to be started after the restore. There will be 2 parameters passed: the first is the path to the backup which was just restored, and the second is the path to the restored file.
- «Limit restore process time, minutes» - here you can set the time limit for restore operation. If this limit will be exceeded, the warning will be sent, saying that restore takes too long.
- «Check available space before restore (bytes)» - here you can set the limit for the minimal free space in the restore destination – if there is less free space than specified, restore will not start, and associated warning will be sent.
- «Send successful email notification» - send email about successful restore (by default it is off, only alerts about problems will be sent).

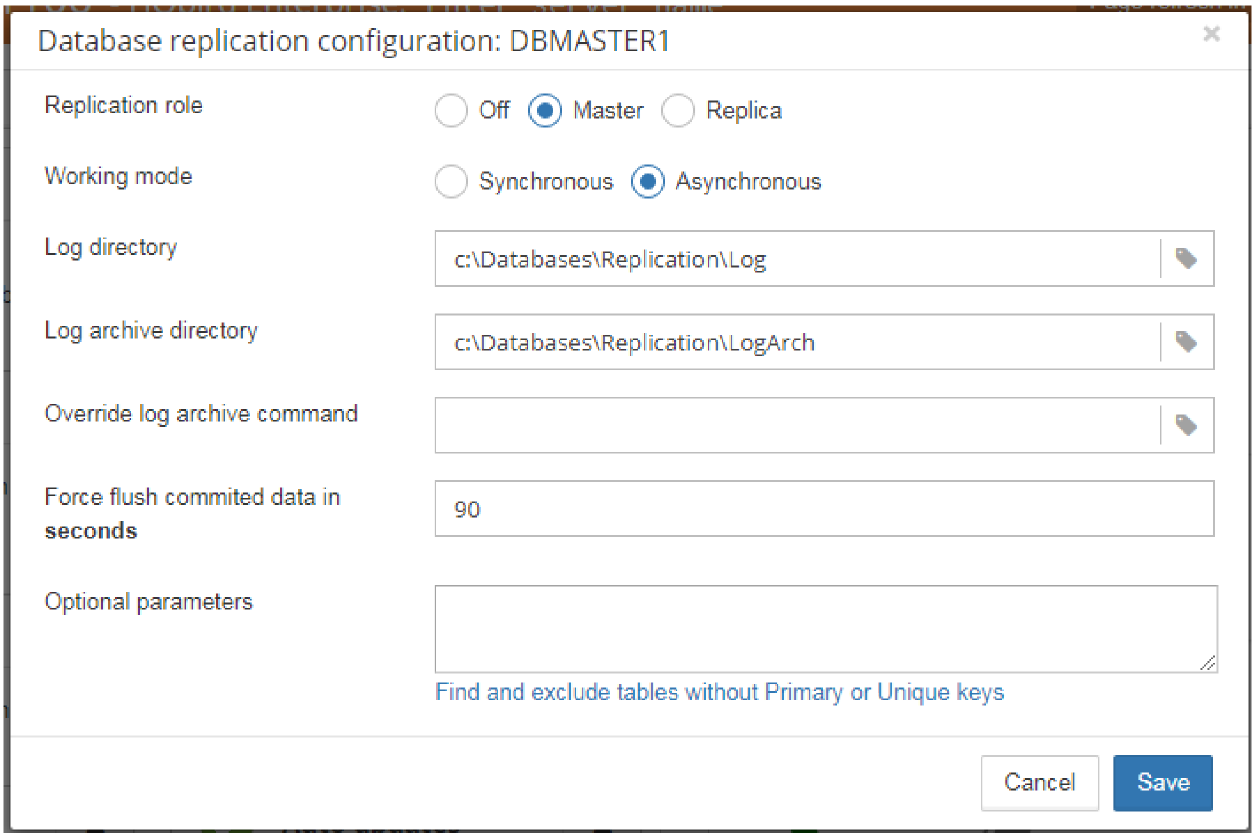
Database: Cloud Backup

In general, Cloud Backup can be used to send any files to FTP/FTPS/etc. For example, you can setup Cloud Backup to look for FBK files, produces by Verified Backup Job, and schedule to upload to the remote FTP server.

The most popular scenario is to use Cloud Backup with archived segments, in order to send them to replica server. Below we will consider how to setup Cloud Backup for this task.

In the case of distributed environment of the asynchronous replication, when the network connection between master and replica server is unstable, or with high latency, or when servers are in the different geographical regions, the best way to transfer replication segments will be through FTP or FTP over SSH. HQbird FBDataGuard has ability to transfer replication segments files from the master server to remote server.

First, the asynchronous master should be configured to save replication logs into the local folder, for example, into C:\Databases\Replication\LogArch in the example below:



Database replication configuration: DBMASTER1

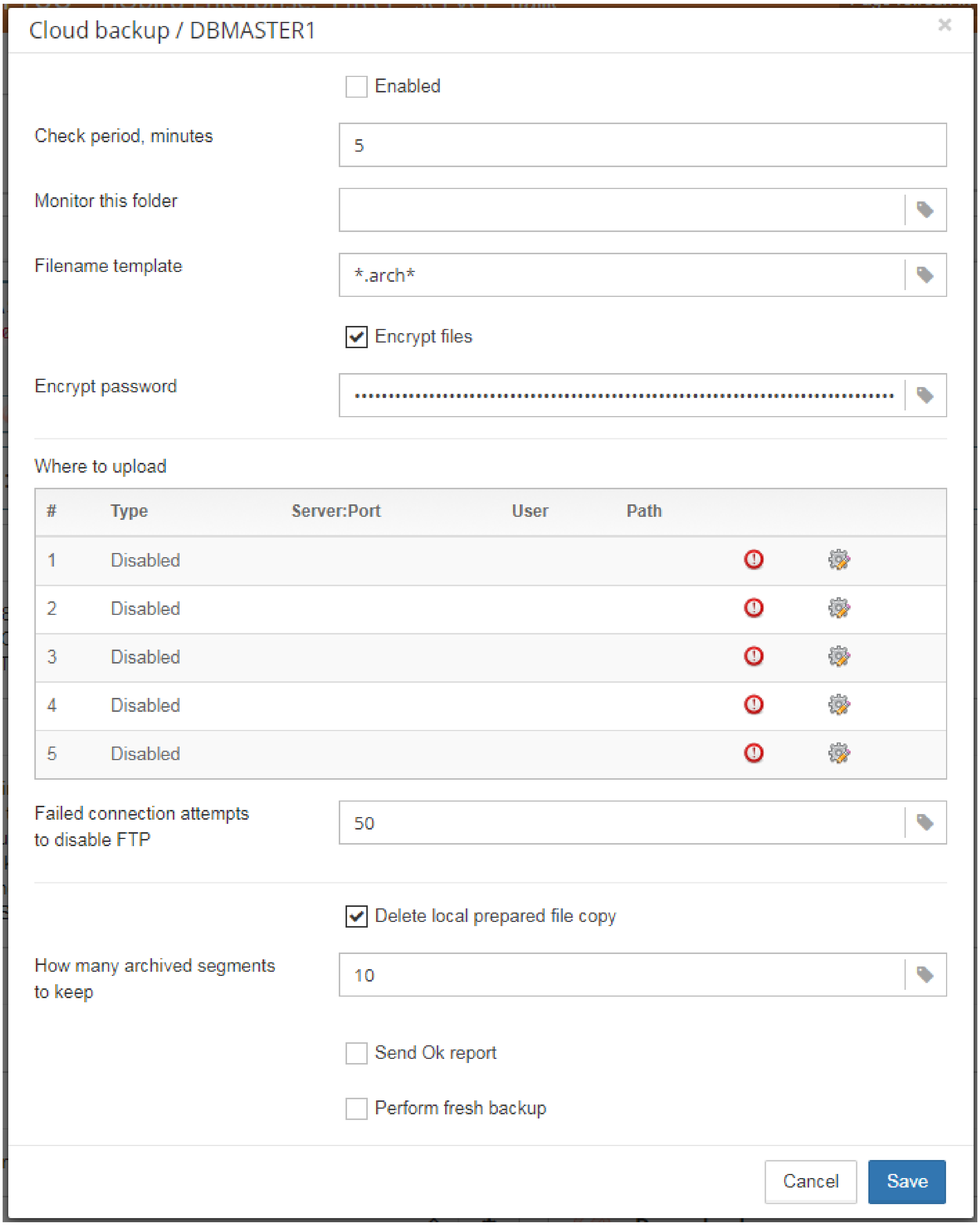
Replication role	Off [✓ Master] Replica
Working mode	Synchronous [✓ Asynchronous]
Log directory	c:\Databases\Replication\Log
Log archive directory	c:\Databases\Replication\LogArch
Override log archive command	
Force flush commited data in **seconds**	90
Optional parameters	

Then we can setup **Cloud Backup** job to monitor the folder for the new replication segments and upload them to the remote FTP server.

As you can see at the screenshot below, Cloud backup job checks folder, specified in «**Monitor this folder**» with an interval, specified in «Check period, minutes».

It watches for the new files with filenames according the mask in «Filename template». By default, it looks for archived replication segments which have names like «dbwmaster.fdb.**arch**-000000001».

Please note – Cloud Backup sends files in the order of their names, not dates.



Cloud backup / DBMASTER1

	☐ Enabled
Check period, minutes	5
Monitor this folder	
Filename template	*.arch*
	☑ Encrypt files
Encrypt password	••

Where to upload

#	Type	Server:Port	User	Path
1	Disabled			
2	Disabled			
3	Disabled			
4	Disabled			
5	Disabled			

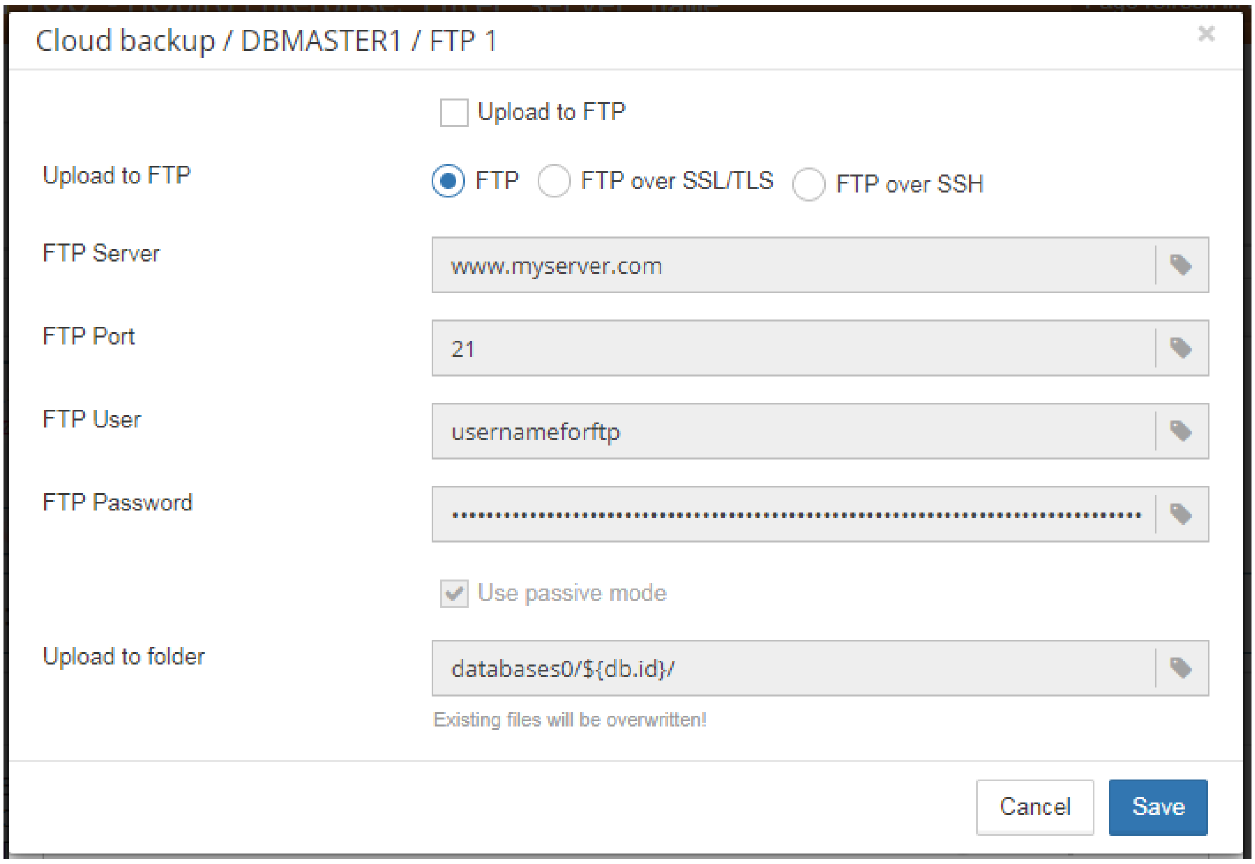
Failed connection attempts to disable FTP	50
	☑ Delete local prepared file copy
How many archived segments to keep	10
	☐ Send Ok report
	☐ Perform fresh backup

By default, Cloud backup compresses and encrypts replication segments before they sent. FBDataGuard creates the compressed and encrypted copy of the replication segment and upload it to the specified target server.

The default password is «zipmasterkey» (without quotes).

There are several types of target servers: FTP, FTP over SSL/TLS, FTP over SSH. When you select the necessary type, dialog shows mandatory fields to be completed.

You can select up to 5 simultaneous remote servers to upload backups. Below you can see the configuration dialog for FTP.


Cloud backup / DBMASTER1 / FTP 1

	☐ Upload to FTP
Upload to FTP	[● FTP] ○ FTP over SSL/TLS ○ FTP over SSH
FTP Server	www.myserver.com
FTP Port	21
FTP User	usernameforftp
FTP Password	•••
	✓ Use passive mode
Upload to folder	databases0/${db.id}/
	Existing files will be overwritten!

***Note**: if you don't have FTP installed on the target server with Windows, install Filezilla – it is very popular fast and lightweight FTP-server for Windows.*

The last part of parameters allows controlling the behavior of Cloud backup.

- **Delete local, prepared copy** – by default it is On. This parameter specify that Cloud backup job deletes compressed copy of the replication segment after the successful upload to the target server. If you don't want to keep these copies on the master server, keep the parameter enabled.
- **Remove original files after successful backup upload** – by default is Off. It means status means that replication segment will be not deleted by FBDataGuard after uploading. It can be useful if you want to keep the full history of changes in replication segments, but,

be careful; in case of an intensive write activity replication segments can occupy a lot of space (Terabytes).

- **Send Ok report** – send email to the specified in Alerts address every time when replication segment is uploaded. By default it is off.
- **Perform fresh backup** – disabled by default. Cloud backup remembers the last number of replication segment it sends. If you need to start again from scratch, from segment 1 (for example, after re-initialization of replication), enable this parameter. Please note that it will automatically become disabled after the resetting of the counter.

As a result, FBDataGuard will upload encrypted and compressed replication segments to the remote server. To decompress and decrypt them into the regular replication segments, another instance of HQbird FBDataGuard should be installed on the replica server, and Cloud Backup Receiver job should be configured – see more details in the section Database: Cloud Backup Receiver.

Fresh backup option

Sometimes it is necessary to re-send files, stored on the master server, to the replica server (for example, re-send archived segments in case of transport error).

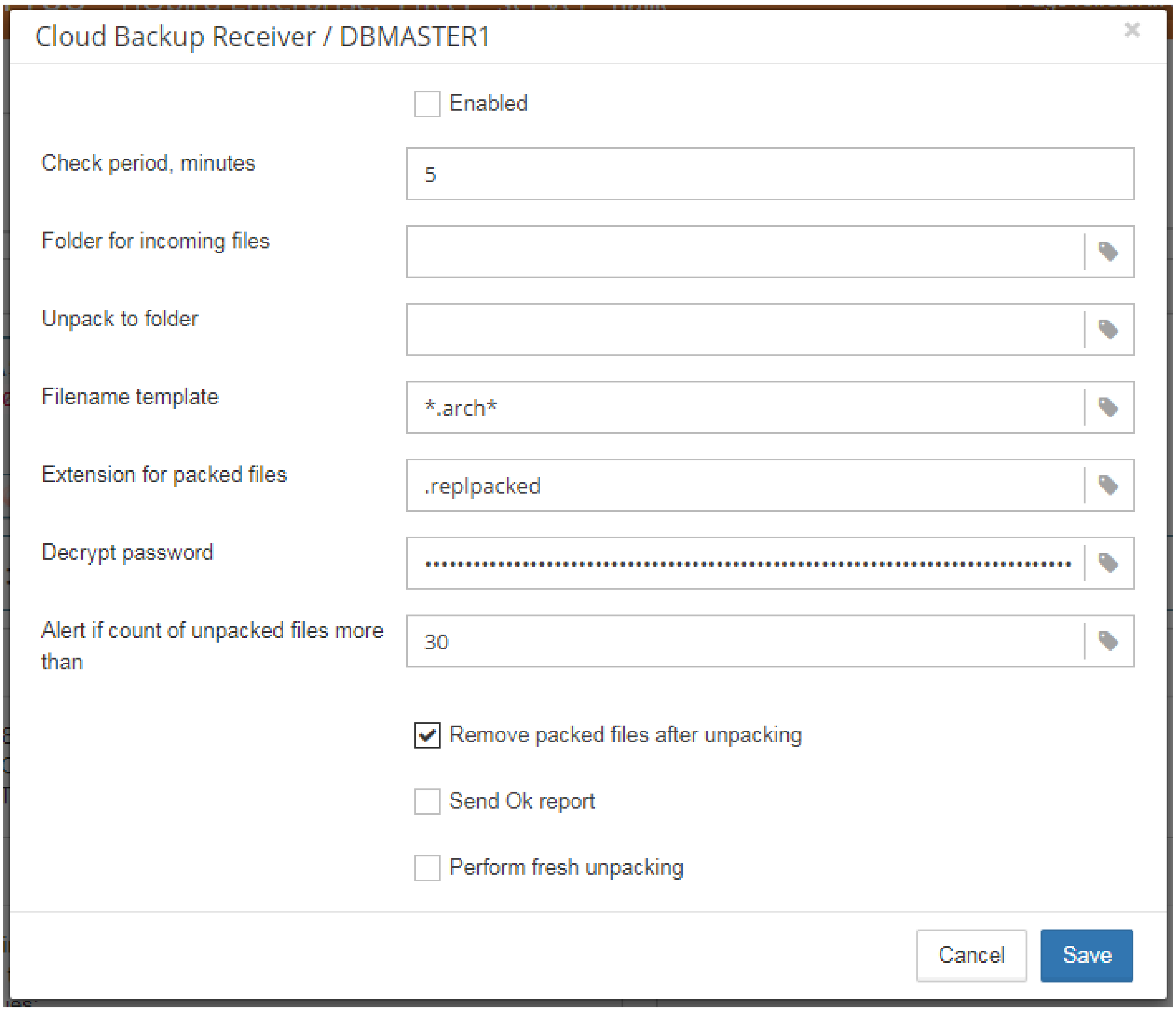
In this case Cloud Backup will resend all files with the appropriate file mask, stored in «Monitor this folder».

However, Cloud Backup Receiver will recognize such files as duplicates and will just delete them. To avoid it, and actually unpack resent segments, select «Perform fresh »

Database: Cloud Backup Receiver

In general, Cloud Backup Receiver is designed to decompress files from zip archives, and the most often is used in the pair with Cloud Backup to transfer archived replication segments.

Cloud Backup Receiver checks files in the folder specified in «Monitor directory», with interval equal to «Check periods, minutes». Its checks only files with specified mask (*arch* by default) and specified extension (.replpacked by default), and if it encounters such files, it decompresses and decrypts them with the password, specified in «Decrypt password», and copies to the folder, specified in «Unpack to directory».


Cloud Backup Receiver / DBMASTER1

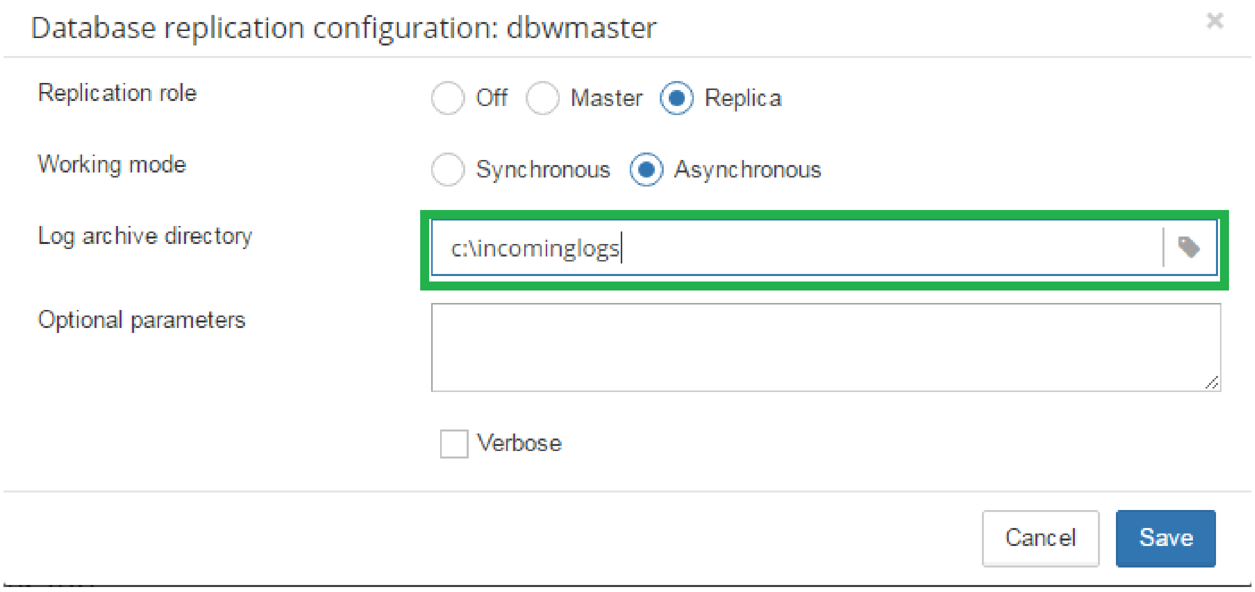
	☐ Enabled
Check period, minutes	5
Folder for incoming files	
Unpack to folder	
Filename template	*.arch*
Extension for packed files	.replpacked
Decrypt password	••
Alert if count of unpacked files more than	30
	☑ Remove packed files after unpacking
	☐ Send Ok report
	☐ Perform fresh unpacking

There are the following additional parameters:

- **Remove packed files after unpack** – by default is On. It means that FBDataGuard will delete received compressed files after successful unpacking.
- **Send Ok report** – by default is Off. If it is On, FBDataGuard sends an email about each successful unpacking of the segment.
- **Perform fresh unpack** – disabled by default. Cloud Backup Receiver remembers the last number of replication segment it unpacked. If you need to start unpacking from scratch, from segment 1 (for example, after re-initialization of replication), enable this

parameter. Please note that it will automatically become disabled after the resetting of the counter.

After setup of Cloud Backup Receiver, configure the replica to look for replication segments: set in the «Log archive directory» the same path as in «Cloud Backup Receiver» -> «Unpack to directory».


Database replication configuration: dbwmaster

Replication role: Off Master [✓ Replica]

Working mode: Synchronous [✓ Asynchronous]

Log archive directory: c:\incominglogs

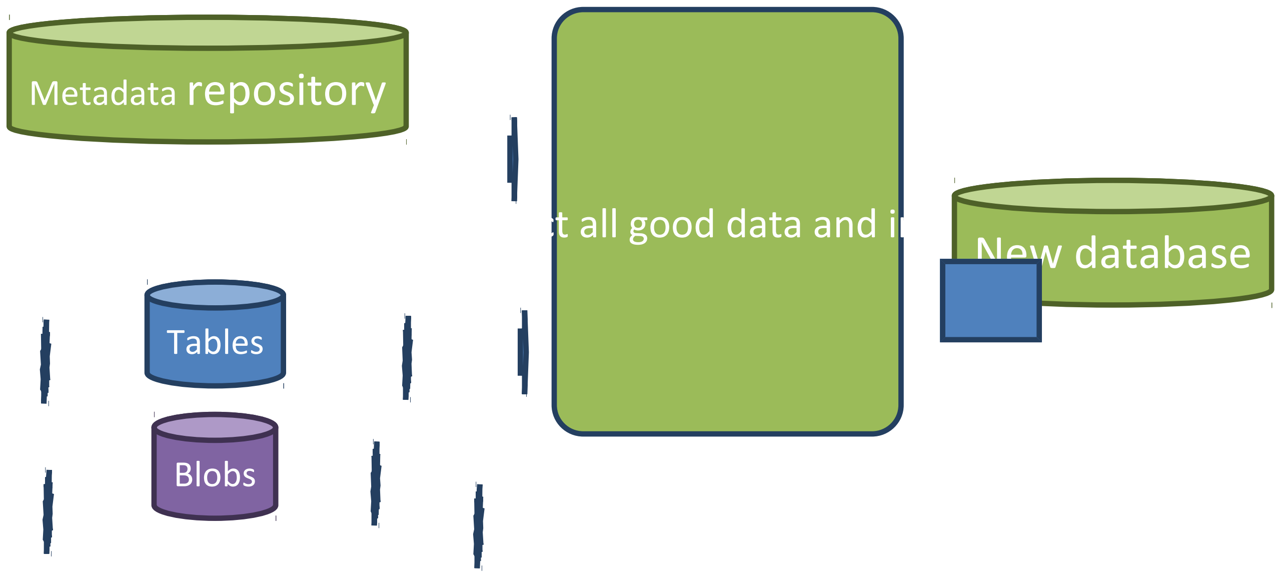
Optional parameters:

☐ Verbose

Database: Store metadata

"Database: Store metadata" is one of the key jobs of DataGuard, it ensures database protection at low level.

First of all, this job stores raw metadata in special repository, so in case of heavy corruption (due to hardware failure, for example) of database it is possible to use this repository to recover database.

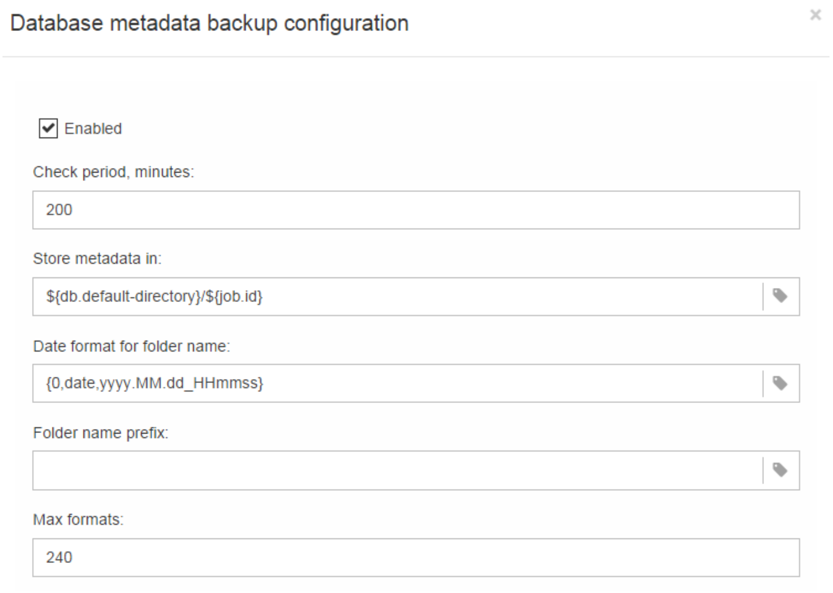


FBDataGuard Extractor can extract data from heavily corrupted databases

The second purpose of this job is to constantly check all important system tables for consistency. Every 20 minutes it walks through all important system tables in the database and ensures that there are no errors at metadata level.

The third purpose is to warn administrator about too many formats for each tables.

There is an implementation limit in Firebird to have 256 formats per table, however even several formats can greatly increase a chance of hard corruption and can slow down the performance. It is recommended do not change tables structure at production database and keep only one format per each table. If it's not possible, administrator should try to perform backup/restore more often to transform all formats into the single one.



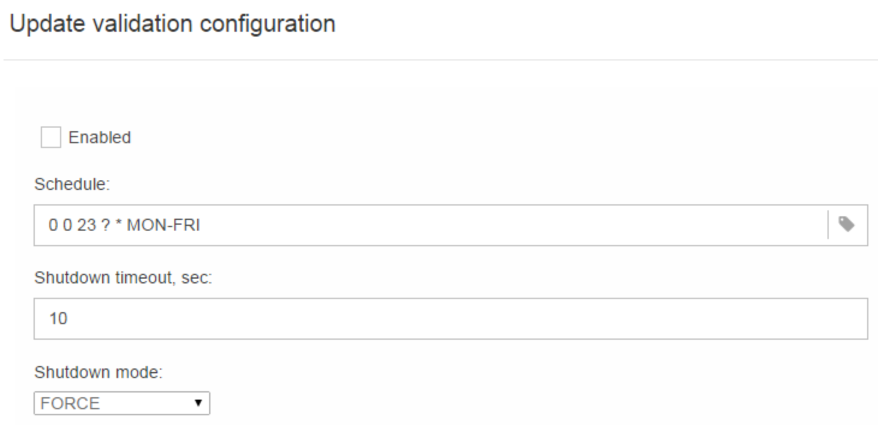
Database metadata backup configuration

☑ Enabled

Check period, minutes:	200
Store metadata in:	${db.default-directory}/${job.id}
Date format for folder name:	{0,date,yyyy.MM.dd_HHmmss}
Folder name prefix:	
Max formats:	240

Database: Validate DB

Validation of Firebird database requires exclusive access: i.e., no users should be connected during validation. "Database: Validate DB" job shuts down the database and performs validation of database, and then turns it on.


Update validation configuration

☐ Enabled

Schedule: 0 0 23 ? * MON-FRI

Shutdown timeout, sec: 10

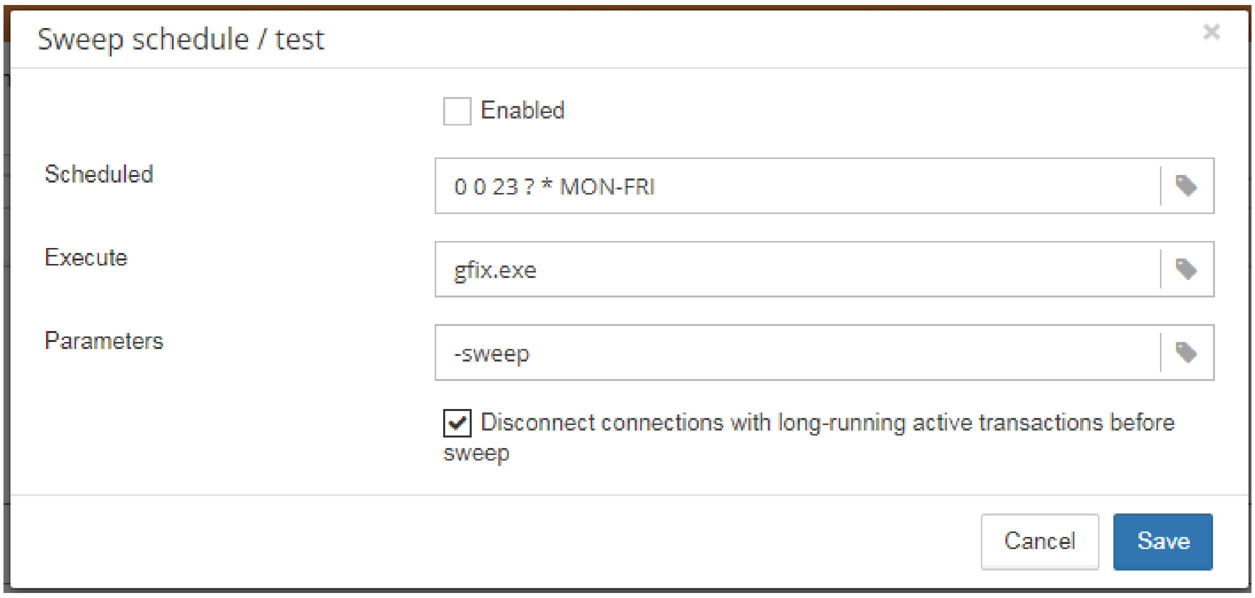
Shutdown mode: FORCE

By default this job is OFF. Please consider carefully, is it possible to provide exclusive access for database. Validation can also take significant time. Using configuration dialog you can enable/disable this job, set time to run, set the shutdown timeout (time to wait before launch validation), and also shutdown mode (FORCE, ATTACH, TRANSNATIONAL). If you have no deep knowledge n what you are doing, it's better to keep default parameters.

"Database: Validate DB" will send alert and put database to critical mode if there will be any errors. Also it is possible that these errors will be written into the firebird.log, so job will act accordingly.

Database: Sweep Schedule

FBDataGuard includes special job to run an explicit sweep, in case if automatic sweep was disabled.



Sweep schedule / test

	☐ Enabled
Scheduled	0 0 23 ? * MON-FRI
Execute	gfix.exe
Parameters	-sweep
	☑ Disconnect connections with long-running active transactions before sweep

Specify CRON expression to run gfix –sweep at the necessary moments.

Please note: by default, check mark «Disconnect all connections with long-running active transactions before» is enabled. It means that HQbird will find and disconnect long-running transactions (more than 30 minutes) before sweep – in order to make sweep efficient.

If long-running active transactions will be not disconnected, sweep cannot clean old records versions.

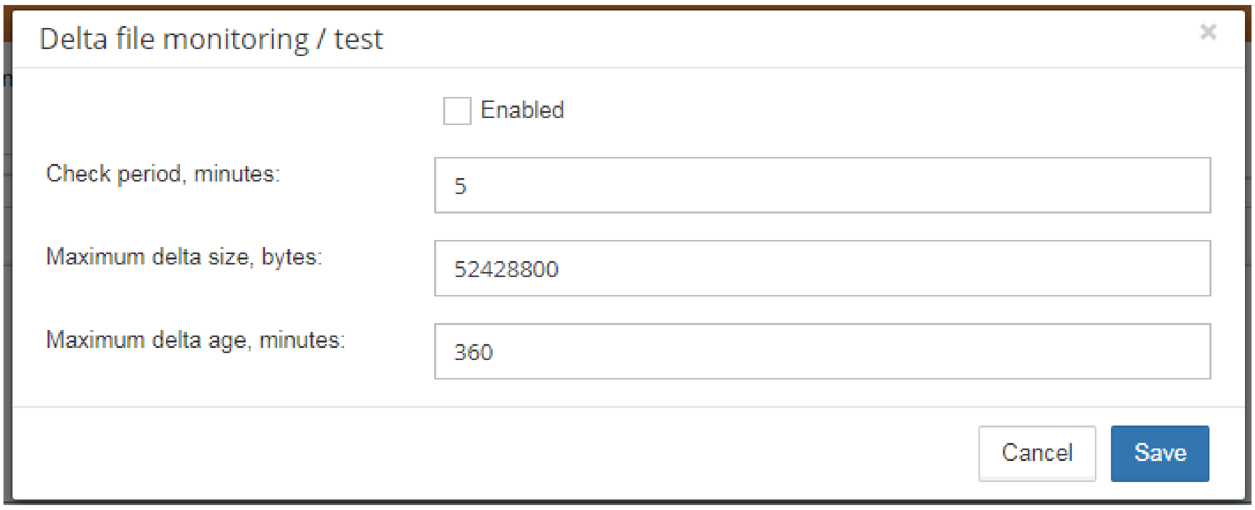
The recommendation is to schedule sweep with disconnection of long-running transactions for all databases where such transactions are detected once per day (usually during the night, after backup's completing).

Database: Delta-files monitoring

If you are using incremental backups (or Dump backup), this job is critically important. It watches for delta-files lifetime and size, and warns if something goes wrong. Forgotten delta-files are the often reason of corruptions and significant losses of data.

This jobs finds all delta files associated with database and check their age and size. If one of these parameters exceeds thresholds “Maximum delta size” or “Maximum delta age”, administrator will receive the alert and database status will be set to CRITICAL.

***Note**: If delta file of the protected database was corrupted, it is possible to extract data from it using metadata from the original database file or repository from Store Metadata.*


Delta file monitoring / test

☐ Enabled

Check period, minutes:	5
Maximum delta size, bytes:	52428800
Maximum delta age, minutes:	360

Database: Disk space

This job watches for all objects related with database: database files (including volumes of multi-volume database), delta-files, backup files and so on.

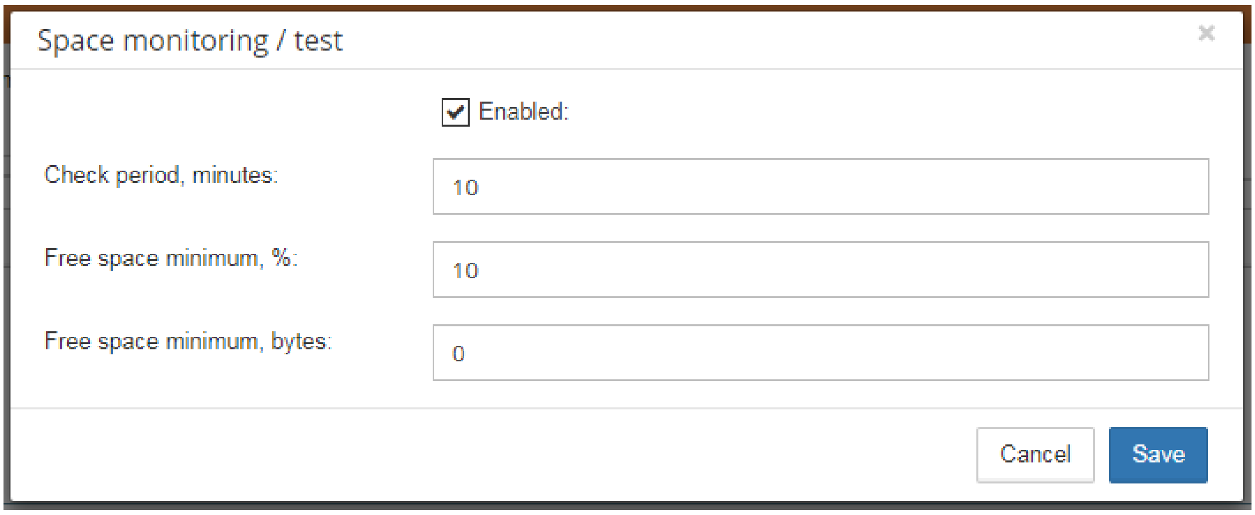
“Database: Disk space” job analyzes the growth of database and estimate will there be enough free space for the next operation like backup (including test restore) on the specific hard drive.

It generates several types of alerts. Problems with disk space are in the top list of corruption reasons, so please pay attention to the alerts from this job.

This job also contributes data to the server space analysis graph (Server: Server space).

By default this job is enabled.

Using configuration dialog you can specify check period and thresholds for free space. The first reached threshold will be alerted. To set threshold only in % of disk space, you need to set explicit space in bytes to “0”.


Space monitoring / test

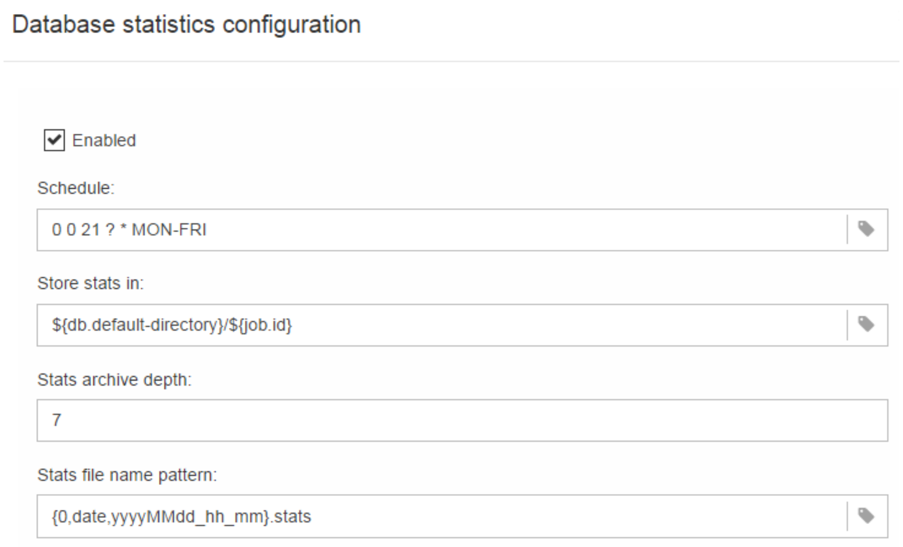
☑ Enabled:

Check period, minutes:	10
Free space minimum, %:	10
Free space minimum, bytes:	0

Database: Database statistics

This job is very useful to capture performance problems and perform overall check of database at low-level without making backup.

We recommend running this job everyday and storing a history of statistics report.
Using HQbird Database IBAnalyst it is possible to discover deep problems with database performance.
As a useful side effect, statistics visits all database pages for tables and indices, and ensures that all of them are correct.


Database statistics configuration

☑ Enabled

Schedule: 0 0 21 ? * MON-FRI

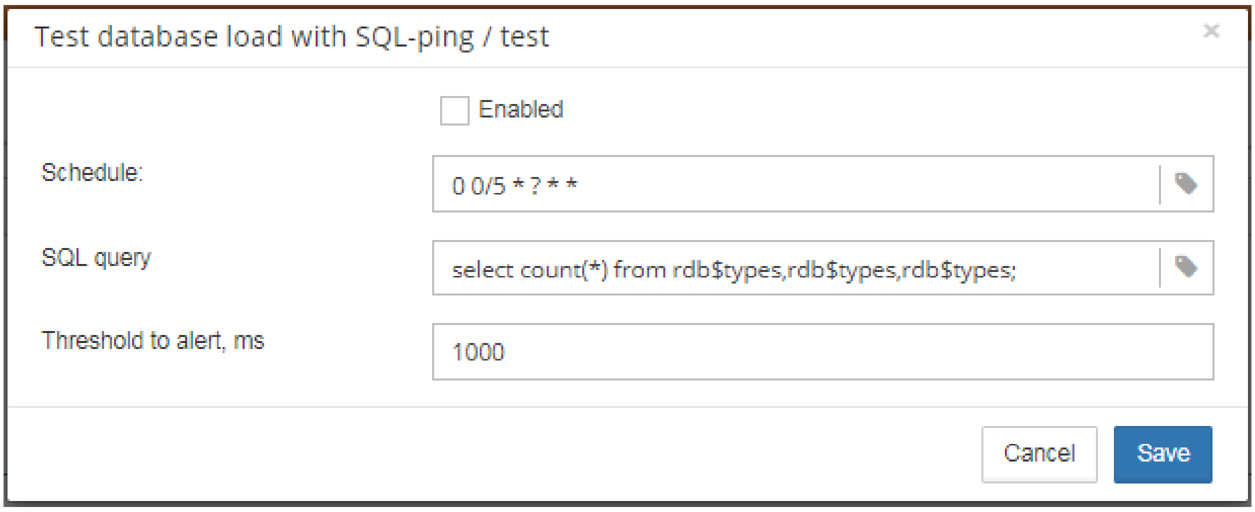
Store stats in: ${db.default-directory}/${job.id}

Stats archive depth: 7

Stats file name pattern: {0,date,yyyyMMdd_hh_mm}.stats

Database: SQL Ping

SQL Ping widget is an easy yet powerful diagnostic tool: it runs an SQL query with a specified schedule (by default CRON expression is set to run every 5 minutes):

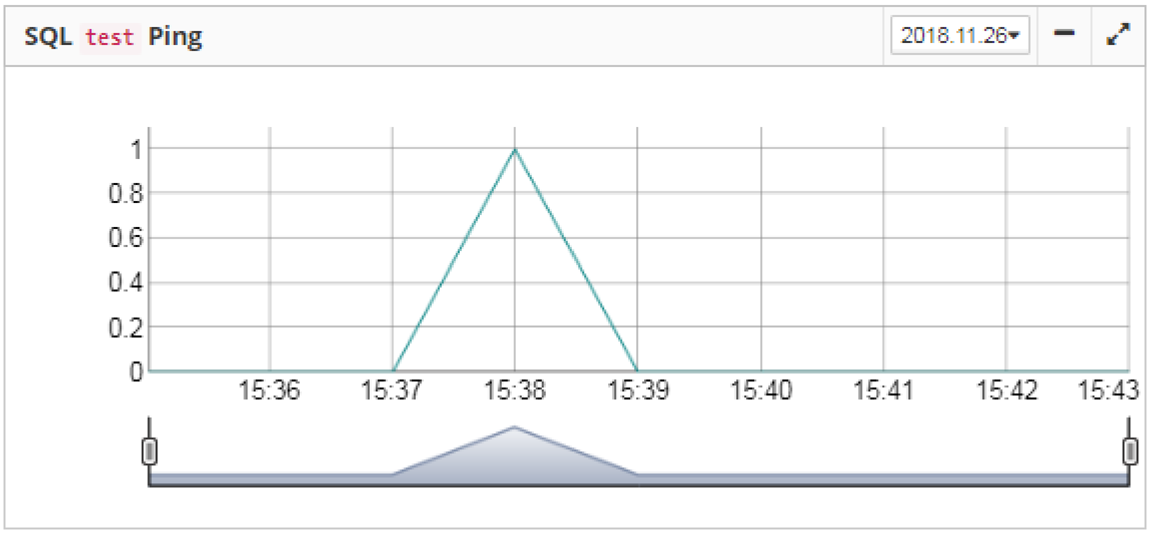


Schedule:	0 0/5 * ? * *
SQL query	select count(*) from rdb$types,rdb$types,rdb$types;
Threshold to alert, ms	1000

The default SQL query is «select count(*) from rdb$types, rdb$types, rdb$types;» and threshold for the alert is 1000 ms.

To make this job more useful, it is recommended to use the SQL query which is relevant for the database: it should be some read-only query which is frequently used by the client applications; in this case the slowness of its execution will mean that database is less responsive than it should be.

There are 2 possible usage of this job: 1) it sends alert if the ping SQL exceeds the threshold, 2) HQbird draws the graph with the response time, so you can see the history of database response.



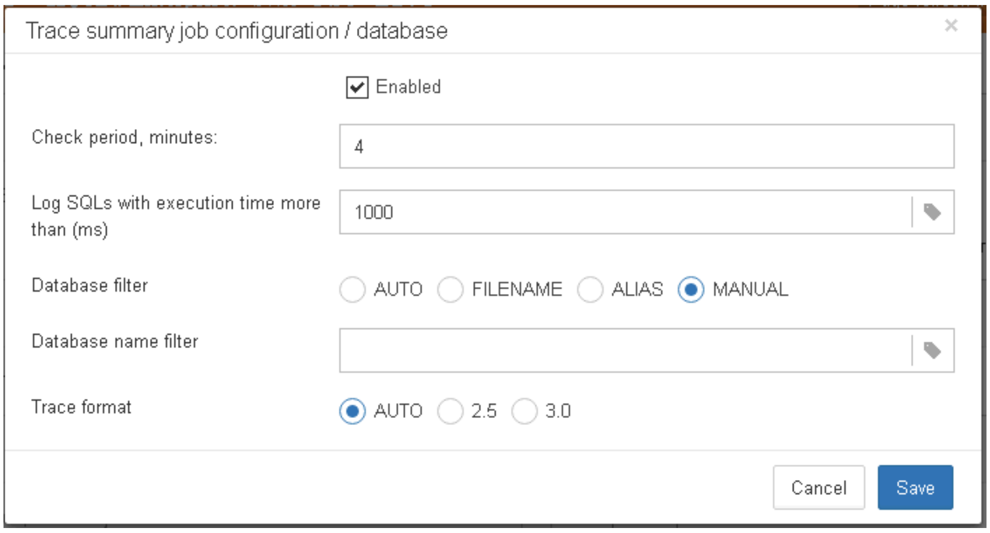
Database: Performance Summary Logging

«Performance Summary» is a new job in HQbird 2018R3, it calculates the average minute amount of database load parameters: reads, writes, fetches and marks, using Trace API mechanism.

- Reads and writes means the actual IO reads and writes from/to the disk system of your server.
- Time is a summary time of all executed SQLs during the minute
- Fetches mean requests to the database cache, and correspond to the CPU load.
- Marks mean how often database pages are changed in the database cache.

To enable performance summary logging, click enabled. We recommend setting AUTO.

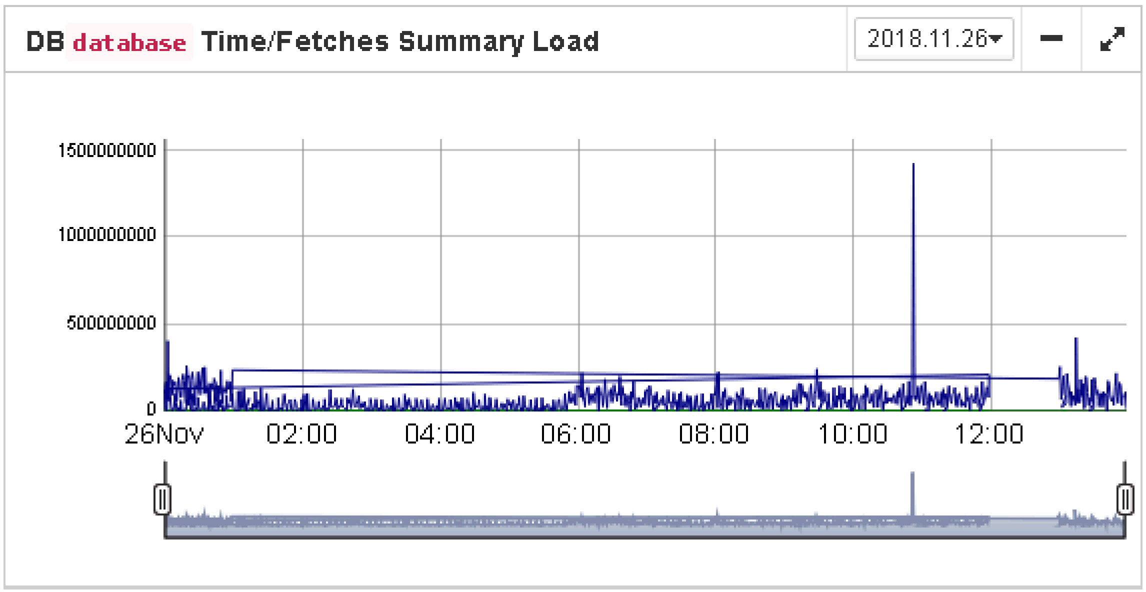
Please note, if there will be set MANUAL with empty string in «Database name filter», it will calculate load for all databases on the server.

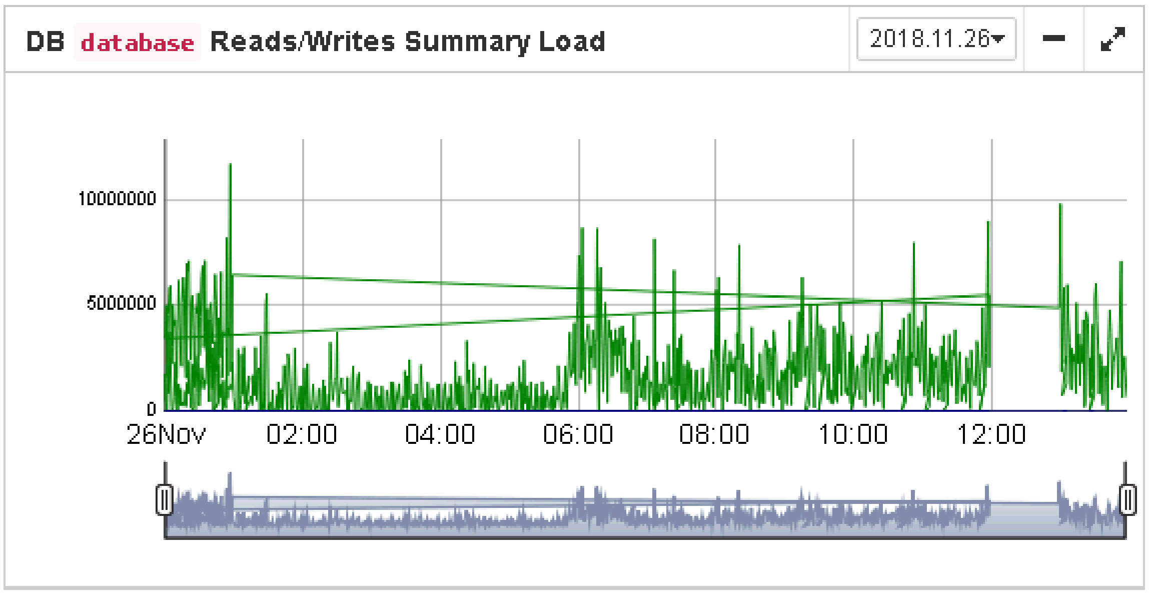

Trace summary job configuration / database

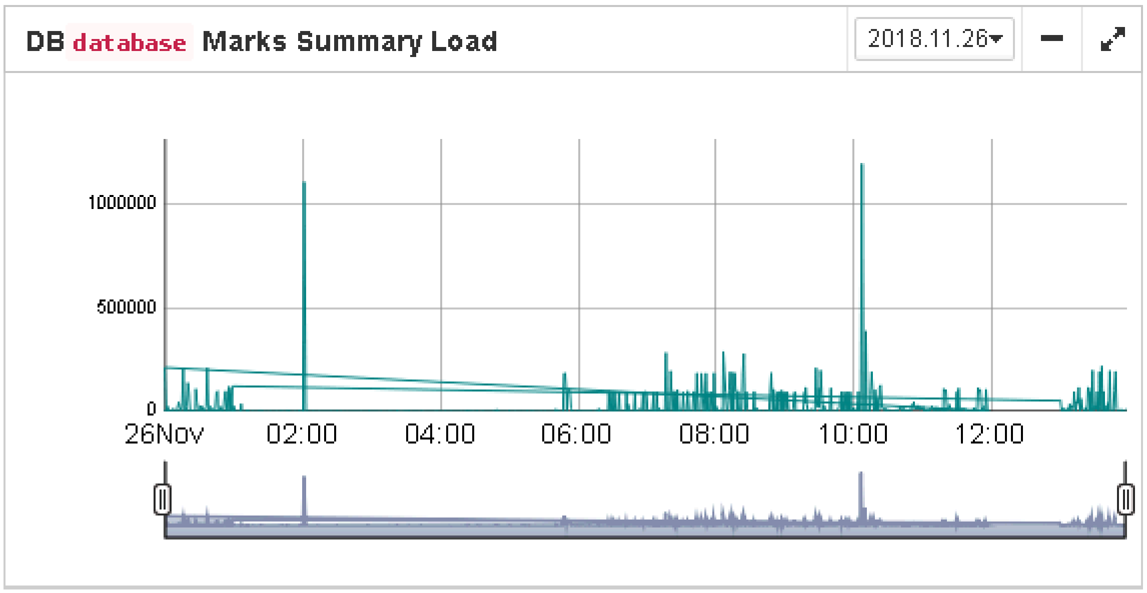
[✓] Enabled

Check period, minutes:	4
Log SQLs with execution time more than (ms)	1000
Database filter	AUTO FILENAME ALIAS [✓ MANUAL]
Database name filter	
Trace format	[✓ AUTO] 2.5 3.0

The result can be seen on the «Graphs» tab of HQbird web-console:

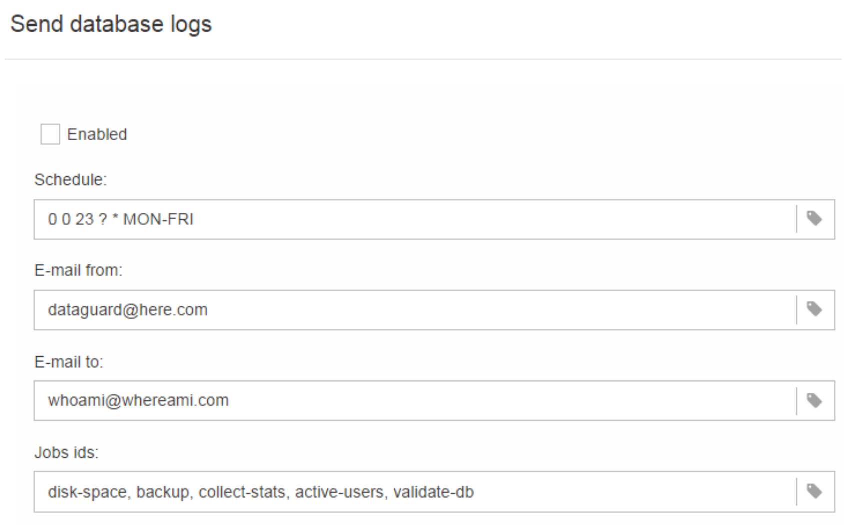






As you can see, it is easy to see peak of the load

Database: Send logs


Send database logs

☐ Enabled

Schedule: 0 0 23 ? * MON-FRI

E-mail from: dataguard@here.com

E-mail to: whoami@whereami.com

Jobs ids: disk-space, backup, collect-stats, active-users, validate-db

"Database: Send logs" is an auxiliary job which sends logs from database-level jobs by email with desired frequency.

This job is disabled by default.

If monitored database is situated at remote location it's useful to schedule logs sending by email. Using configuration dialog you can schedule logs sending with CRON expression (for more details see ***Appendix: CRON Expressions***), specify from what email it will be sent and where. Setting from email alerts settings will be used (for details see Email alerts in HQbird FBDataGuard).

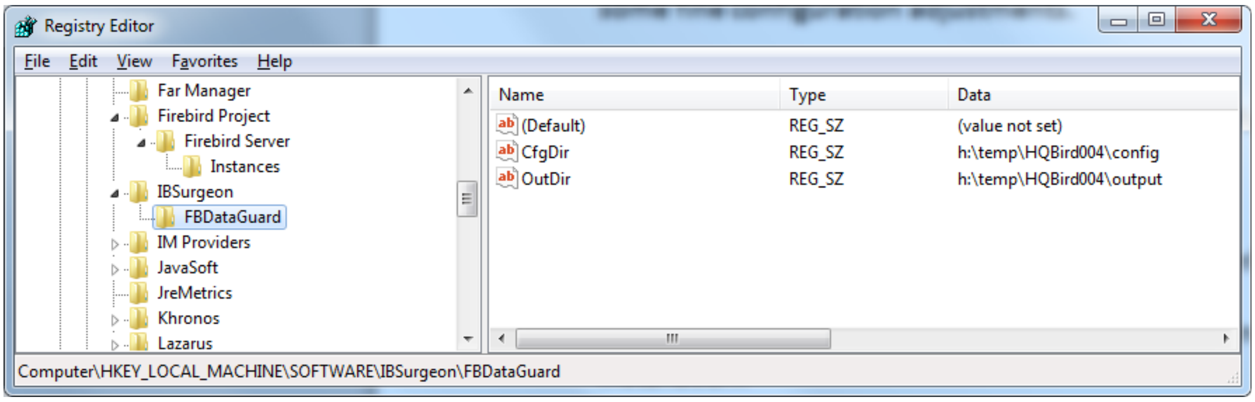
Also you can specify logs from what jobs you need to send by specifying Jobs IDs.

FBDataGuard tips&tricks

FBDataGuard allows changing its setting not only through web-console, but also using direct modification of configuration files. This can be useful when you need to install FBDataGuard in silent mode (no interaction with user), to bundle it with third-party software, or to perform some fine configuration adjustments.

Path to FBDataGuard configuration

During the start FBDataGuard looks for in registry for configuration and output paths:



Name	Type	Data
(Default)	REG_SZ	(value not set)
CfgDir	REG_SZ	h:\temp\HQBird004\config
OutDir	REG_SZ	h:\temp\HQBird004\output

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBSurgeon\FBDataGuard

These values specify the paths to FBDataGuard configuration and output folder – these values are chosen during installation.

Adjusting web-console port

One of the most frequently asked questions is how to adjust port for web-console application (by default it is 8082), It can be done by changing port setting in file

%config%\agent\agent.properties

server.port = 8082 #change it

%config% - folder to store configuration information, it is specified in ***Path to FBDataGuard configuration***

Appendix: CRON Expressions

All jobs in FBDataGuard have time settings in CRON format. CRON is very easy and powerful format to schedule execution times.

CRON Format

A CRON expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. The fields are as follows:

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

So cron expressions can be as simple as this: `* * * * ? *`
or more complex, like this: `0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2010`

Special characters

***** (*"all values"*) - used to select all values within a field. For example, "*" in the minute field means *"every minute"*.

? (*"no specific value"*) - useful when you need to specify something in one of the two fields in which the character is allowed, but not the other. For example, if I want my trigger to fire on a particular day of the month (say, the 10th), but don't care what day of the week that happens to be, I would put "10" in the day-of-month field, and "?" in the day-of-week field. See the examples below for clarification.

- - used to specify ranges. For example, "10-12" in the hour field means *"the hours 10, 11 and 12"*.

, - used to specify additional values. For example, "MON,WED,FRI" in the day-of-week field means *"the days Monday, Wednesday, and Friday"*.

`/` - used to specify increments. For example, "0/15" in the seconds field means *"the seconds 0, 15, 30, and 45"*. And "5/15" in the seconds field means *"the seconds 5, 20, 35, and 50"*. You can also specify '/' after the '' **character – in this case** '' is equivalent to having '0' before the '/'. '1/3' in the day-of-month field means *"fire every 3 days starting on the first day of the month"*.

`L` (*"last"*) - has different meaning in each of the two fields in which it is allowed. For example, the value "L" in the day-of-month field means *"the last day of the month"* - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But if used in the day-of-week field after another value, it means *"the last xxx day of the month"* - for example "6L" means *"the last Friday of the month"*. When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing results.

`W` (*"weekday"*) - used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: *"the nearest weekday to the 15th of the month"*. So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The 'W' character can only be specified when the day-of-month is a single day, not a range or list of days.

The 'L' and 'W' characters can also be combined in the day-of-month field to yield 'LW', which translates to *"last weekday of the month"*.

`#` - used to specify "the nth" XXX day of the month. For example, the value of "6#3" in the day-of-week field means *"the third Friday of the month"* (day 6 = Friday and "#3" = the 3rd one in the month). Other examples: "2#1" = the first Monday of the month and "4#5" = the fifth Wednesday of the month. Note that if you specify "#5" and there is not 5 of the given day-of-week in the month, then no firing will occur that month.

The legal characters and the names of months and days of the week are not case sensitive. `MON` is the same as `mon`.

CRON Examples

Here are some full examples:

Expression	Meaning
`0 0 12 * * ?`	Fire at 12pm (noon) every day
`0 15 10 ? * *`	Fire at 10:15am every day
`0 15 10 * * ?`	Fire at 10:15am every day
`0 15 10 * * ? *`	Fire at 10:15am every day

`0 15 10 * * ? 2005`	Fire at 10:15am every day during the year 2005
`0 * 14 * * ?`	Fire every minute starting at 2pm and ending at 2:59pm, every day
`0 0/5 14 * * ?`	Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day
`0 0/5 14,18 * * ?`	Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day
`0 0-5 14 * * ?`	Fire every minute starting at 2pm and ending at 2:05pm, every day
`0 10,44 14 ? 3 WED`	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
`0 15 10 ? * MON-FRI`	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday
`0 15 10 15 * ?`	Fire at 10:15am on the 15th day of every month
`0 15 10 L * ?`	Fire at 10:15am on the last day of every month
`0 15 10 ? * 6L`	Fire at 10:15am on the last Friday of every month
`0 15 10 ? * 6L`	Fire at 10:15am on the last Friday of every month
`0 15 10 ? * 6L 2002-2005`	Fire at 10:15am on every last Friday of every month during the years 2002, 2003, 2004 and 2005
`0 15 10 ? * 6#3`	Fire at 10:15am on the third Friday of every month
`0 0 12 1/5 * ?`	Fire at 12pm (noon) every 5 days every month, starting on the first day of the month.
`0 11 11 11 11 ?`	Fire every November 11th at 11:11am.

Pay attention to the effects of '?' and '*' in the day-of-week and day-of-month fields!

Notes

Support for specifying both a day-of-week and a day-of-month value is not complete (you must currently use the '?' character in one of these fields).

Be careful when setting fire times between mid-night and 1:00 AM - "daylight savings" can cause a skip or a repeat depending on whether the time moves back or jumps forward.

More information is here

http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html

3.2. Configuring firebird.conf for the best performance

HQbird includes set of optimized configuration files for all Firebird versions from 1.5 to 2.5 – they are located in <Program Files>\IBSurgeon\HQBird Server Side\Configurations.

If you did not perform a justified tuning of firebird.conf or you are using default firebird.conf, consider to use one of the optimized files from this collection.

There are three variants of Firebird configuration files for every Firebird architecture: balanced, read-intensive and write intensive. We always recommend to start with balanced firebird.conf. Then we recommend to measure actual ratio between reads and writes using HQbird MonLogger tool (tab «Aggregated Performance Statistics»). In 90% of cases there are much more reads than writes, so the next step is to try read-optimized firebird configuration file.

Firebird configuration greatly depends on the hardware, so if you want to tune Firebird properly, please also read «Firebird Hardware Guide», it will help you to understand what parameters must be tuned.

For the deep tuning of high-load Firebird databases IBSurgeon offers Firebird Database Optimization Service:

https://ib-aid.com/en/firebird-interbase-performance-optimization-service/

Also, HQbird FBDataGuard analyses the database health and sends alerts with intelligent suggestions to increase specific parameters in firebird.conf, like TempCacheLimit or LockHashSlots.

Attention! If you have specified many page buffers in the header of your database and installed SuperClassic or Classic, it can affect Firebird performance. To avoid the potential problem, set page buffers in the header of your database to 0, it will ensure that the value from firebird.conf will be used:

```
gfix -buff 0 -user SYSDBA -pass masterkey disk:\path\database.fdb
```

4. Monitoring

4.1. Monitoring with HQbird FBDataGuard

Overview

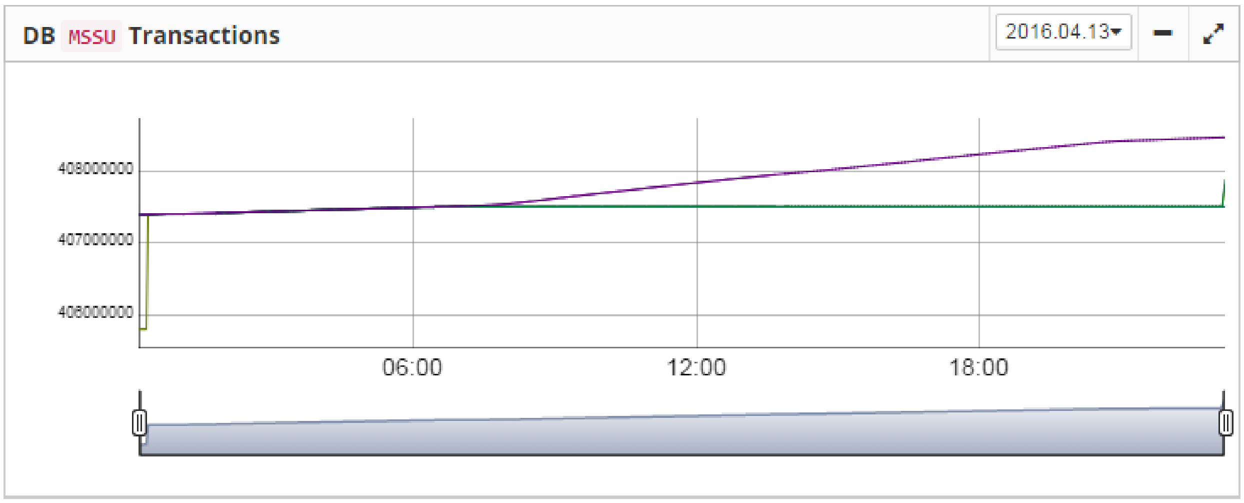
HQbird monitors all aspects of Firebird server and database functioning, and includes continuous monitoring and detailed monitoring.

The continuous monitoring is performed by HQbird FBDataGuard. It is low-invasive, but very effective monitoring which can help to find and resolve the majority of issues with databases performance and stability.

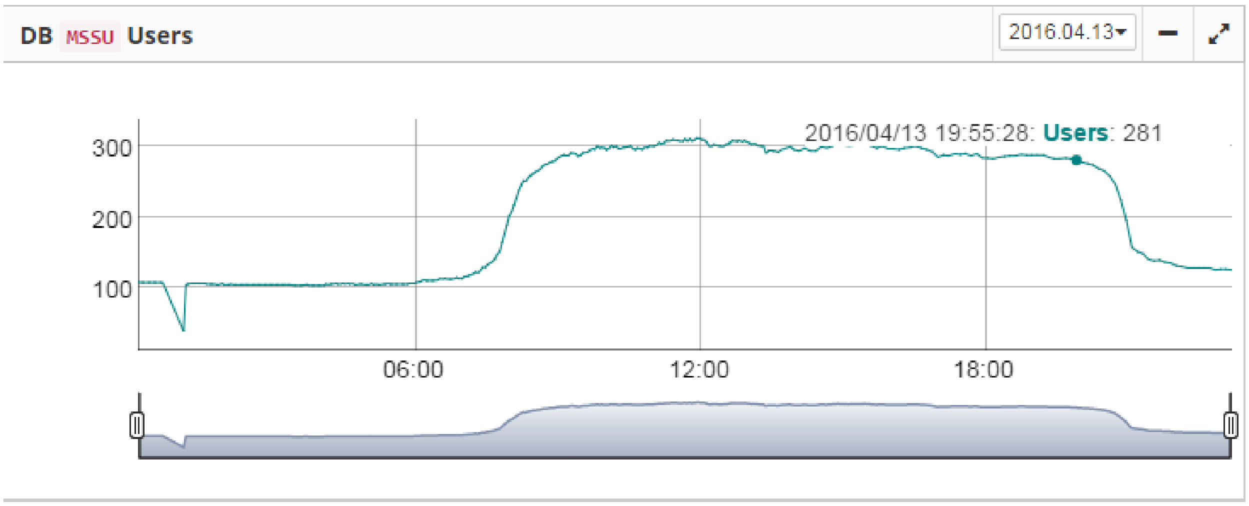
FBDataGuard performs the following monitoring activities:

- Optional performance monitoring with TraceAPI and MON$
- Monitoring of transactions markers dynamics (Next, OAT, OIT, OST, active transactions)
- Monitoring of lock table activity (queues, deadlocks, mutexes)
- Monitoring of Firebird log (errors, warnings, messages)
- Number of connected users
- Free space monitoring for server, databases and their backups
- Health monitoring through analysis of database metadata and general availability of server and databases
- Number and size of Firebird temporary files (sorting, etc)
- Indices monitoring: health and activity check
- General validation of Firebird database
- Backups statuses monitoring

FBDataGuard can graphically represent information gathered during the monitoring, for example, transactions and number of users:



Transaction dynamics in FBDataGuard



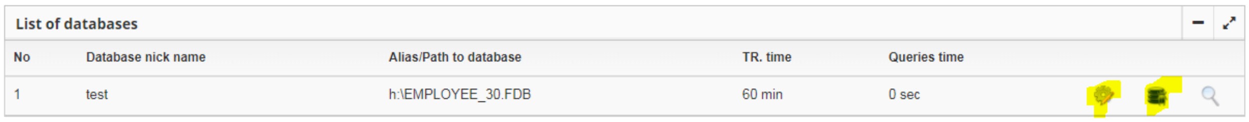
Number of users

Automatic monitoring with FBDataGuard (Trace API and MON$)

In the version 2018, HQBird introduces the new approach for the automatic performance monitoring with Firebird MON$ tables and TraceAPI.

Now it is possible to schedule the regular check of database performance for every HQBird (Standard, Professional, Enterprise) in less than 1 minute.

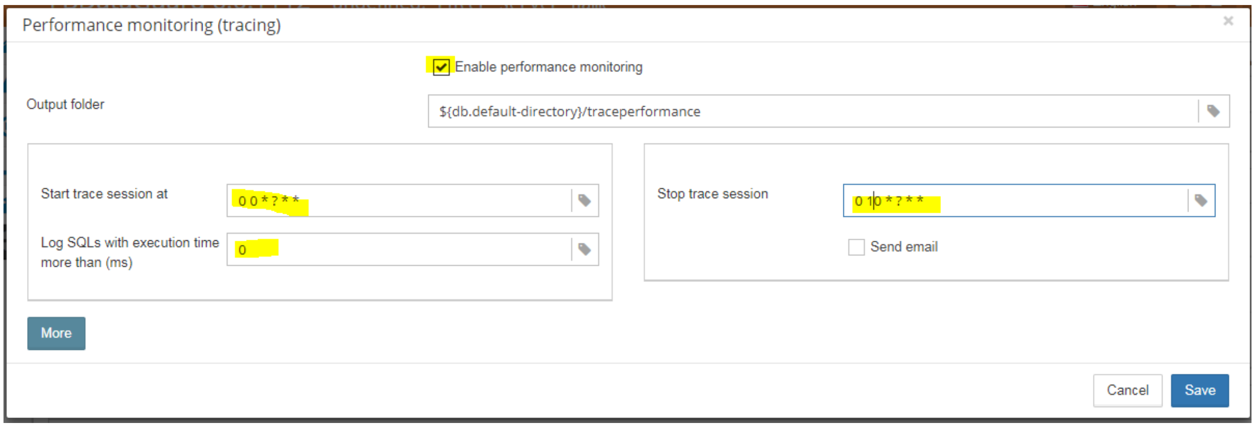
For this just open tab Performance and setup monitoring for transactions and queries:



List of databases

No	Database nick name	Alias/Path to database	TR. time	Queries time
1	test	h:\EMPLOYEE_30.FDB	60 min	0 sec

In order to setup Performance monitoring, specify its mandatory parameters in the dialog:



Performance monitoring (tracing)

[✓] Enable performance monitoring

Output folder: ${db.default-directory}/traceperformance

Start trace session at: 0 0 * ? * *

Log SQLs with execution time more than (ms): 0

Stop trace session: 0 10 * ? * *

[ ] Send email

The first mandatory parameter is «**Enable performance monitoring**» - it must be enabled to run traces by schedule.

The next important parameters are «Start trace session at» and «Stop trace session». They contain CRON expressions which specify when tracing starts and stops.

By default, trace is set to start at 10-30 and to end at 11-00. It is recommended to adopt tracing schedule for your needs. Below you can see the table with some popular options.

CRON expression for		Description
Start	End	
0 0 * ? * *	0 10 * ?* *	Run trace every hour from 0 to 10 minutes
0 0 8 ? * *	0 0 17 ?* *	Run trace every day from 8-00 to 17-00
0 30 10,13,15 ? * *	0 0 11,14,16 ? * *	Run trace sessions every day from 10-30 to 11-00, 13-30 to 14-00 and from 15-30 to 16-00

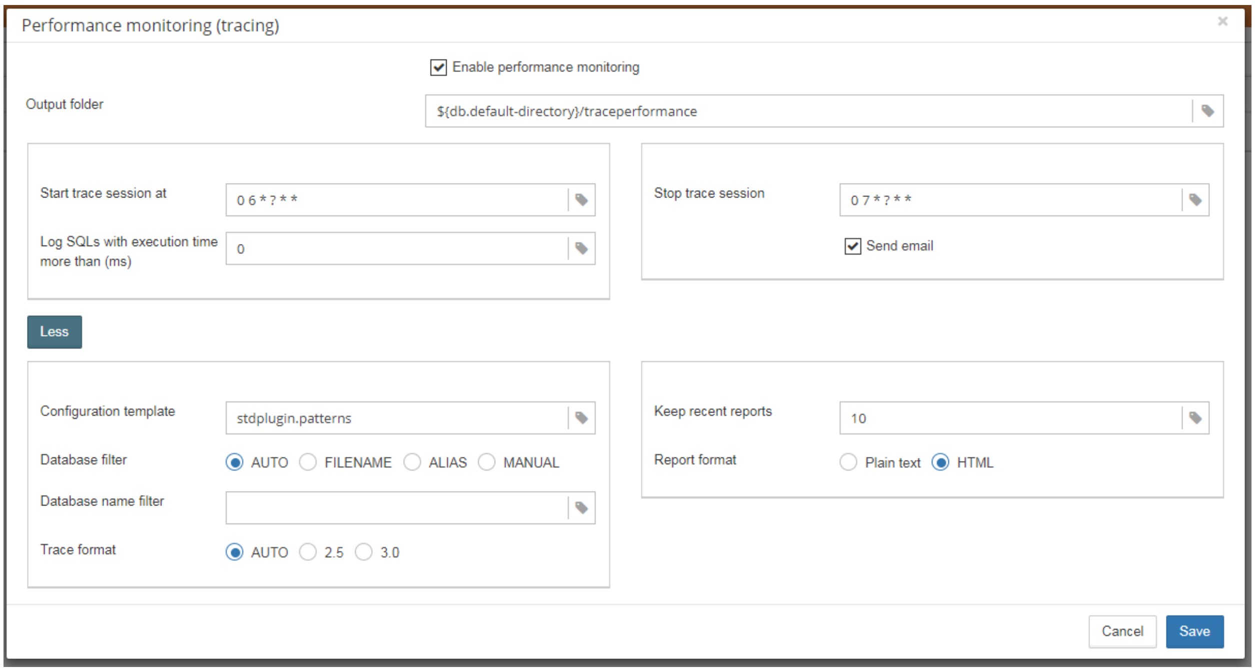
The next important parameter is time threshold for the slow queries, it is set in the field «Log SQLs with execution time more than». In this field you need to set time threshold (in milliseconds), after exceeding it logs will be stored and analyzed.

By default, the time is set to 1000 milliseconds, or 1 second. It means, that only queries which take more than 1 seconds, will be logged and analyzed.

We recommend keeping 1000 ms as a basic value, until your database is very slow: in this case 3000-5000 ms can be a good start.

«Send email» check mark indicates if there necessity to send the performance report. The email settings from Alerts configuration will be used to send performance report.

For more advanced settings, «Performance Monitoring» dialog has additional parameters (normally, you don' t need to adjust them).


Performance monitoring (tracing)

[✓] Enable performance monitoring

Output folder: ${db.default-directory}/traceperformance

Start trace session at: 0 6 * ? * *

Log SQLs with execution time more than (ms): 0

Stop trace session: 0 7 * ? * *

[✓] Send email

Less

Configuration template: stdplugin.patterns

Database filter: [✓ AUTO] FILENAME ALIAS MANUAL

Database name filter:

Trace format: [✓ AUTO] 2.5 3.0

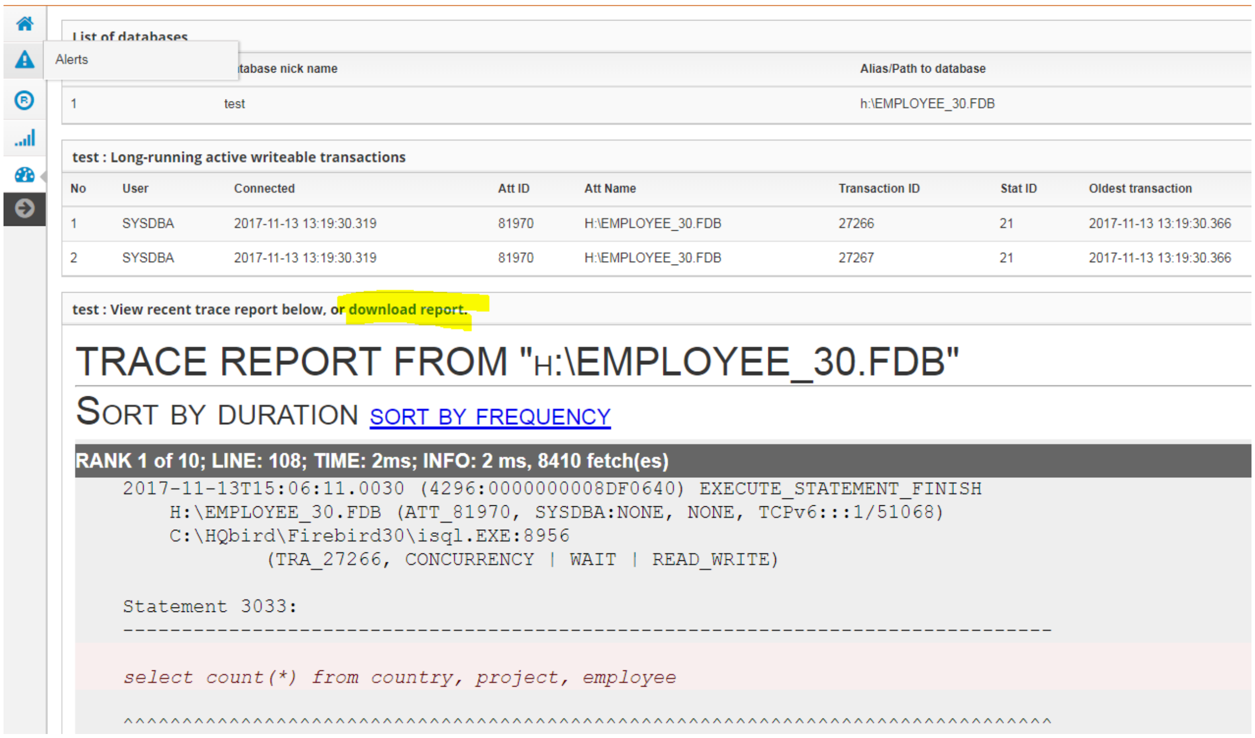
Keep recent reports: 10

Report format: Plain text [✓ HTML]

Cancel Save

- «Configuration template» - name of the configuration template file which should be used for trace settings
- «Database filter» - how the database should be identified. Usually AUTO is enough, it will trace specified database. In case of Filename or Alias it will use filename or alias to filter database events. «Manual» provides an ability to set any regular expression, to trace several databases, for example, or more than one alias for the single database.
- «Database name filter» it is used in case of «MANUAL» selection.
- «Trace format» - AUTO means automatic selection, 2.5 or 3.0 will force format for 2.5 or 3.0. Usually there is no need to change it.
- «Keep recent reports» - it specifies how many reports should be kept in the «Output folder» for possible retrospective usage.

As a result of this job, HQbird will generate the performance report, which will be stored in the Output folder as a file with the extension **html**, and it will be sent by email (in case if «Send email» is enabled). Also the most recent performance is available for review and download in the HQBird interface:



List of databases

	tabase nick name	Alias/Path to database
1	test	h:\EMPLOYEE_30.FDB

test : Long-running active writeable transactions

No	User	Connected	Att ID	Att Name	Transaction ID	Stat ID	Oldest transaction
1	SYSDBA	2017-11-13 13:19:30.319	81970	H:\EMPLOYEE_30.FDB	27266	21	2017-11-13 13:19:30.366
2	SYSDBA	2017-11-13 13:19:30.319	81970	H:\EMPLOYEE_30.FDB	27267	21	2017-11-13 13:19:30.366

test : View recent trace report below, or download report.

TRACE REPORT FROM "H:\EMPLOYEE_30.FDB"

SORT BY DURATION SORT BY FREQUENCY

RANK 1 of 10; LINE: 108; TIME: 2ms; INFO: 2 ms, 8410 fetch(es)

```
    2017-11-13T15:06:11.0030 (4296:0000000008DF0640) EXECUTE_STATEMENT_FINISH
        H:\EMPLOYEE_30.FDB (ATT_81970, SYSDBA:NONE, NONE, TCPv6:::1/51068)
        C:\HQbird\Firebird30\isql.EXE:8956
                (TRA_27266, CONCURRENCY | WAIT | READ_WRITE)

    Statement 3033:
    -------------------------------------------------------------------------------

    select count(*) from country, project, employee

    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

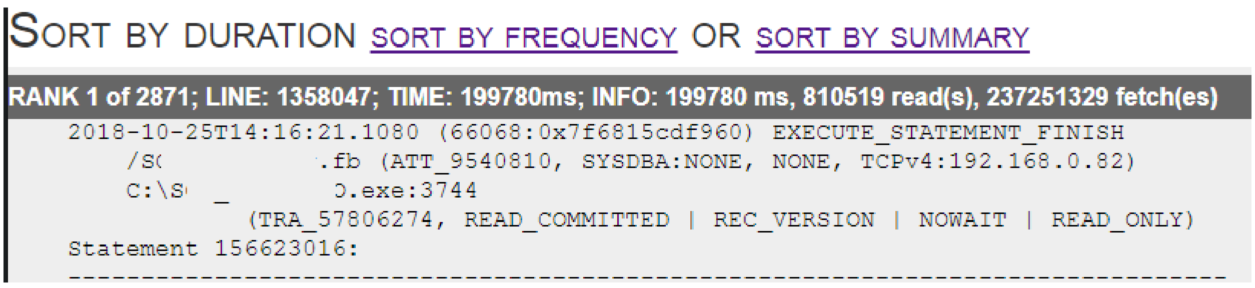
What can we see in the performance report?

The HQbird FBDataGuard performance analysis provides 3 types of reports:

1. list of queries sorted by their time – «Sort by duration»,
2. list of queries sorted by its frequency – «Sort by frequency»,
3. list of queries sorted by the total time (i.e.,summary execution time for queries with the same text and various parameters) – «Sort by summary».

When you click «Sort by duration» (it is a default option), you will see SQL queries and stored procedures which took the longest time to execute first.

Normally there will be long-running reports and other big SQLs.

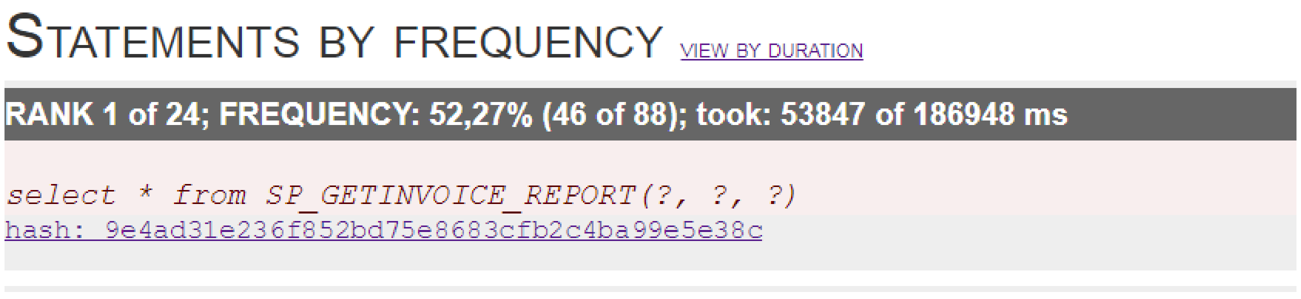


SORT BY DURATION SORT BY FREQUENCY OR SORT BY SUMMARY

RANK 1 of 2871; LINE: 1358047; TIME: 199780ms; INFO: 199780 ms, 810519 read(s), 237251329 fetch(es)

```
    2018-10-25T14:16:21.1080 (66068:0x7f6815cdf960) EXECUTE_STATEMENT_FINISH
        /S(            .fb (ATT_9540810, SYSDBA:NONE, NONE, TCPv4:192.168.0.82)
        C:\S  _          ).exe:3744
                (TRA_57806274, READ_COMMITTED | REC_VERSION | NOWAIT | READ_ONLY)
    Statement 156623016:
    -------------------------------------------------------------------------------
```

When you click on «Sort by frequency» link in the header of the report, you will see most frequent queries: i.e., those queries which started frequently (among logged queries).



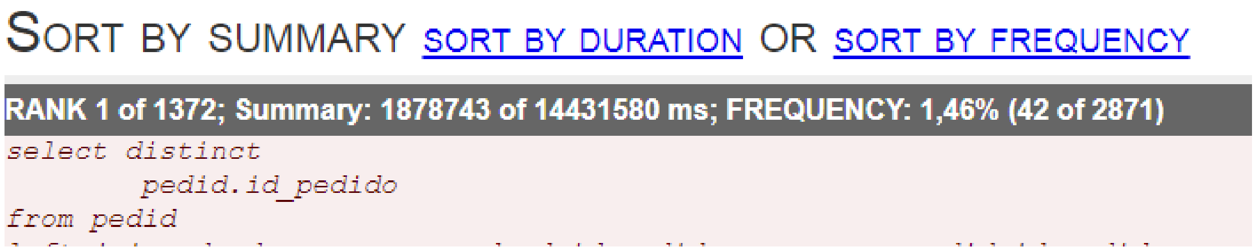
STATEMENTS BY FREQUENCY VIEW BY DURATION

RANK 1 of 24; FREQUENCY: 52,27% (46 of 88); took: 53847 of 186948 ms

```
select * from SP_GETINVOICE_REPORT(?, ?, ?)
hash: 9e4ad31e236f852bd75e8683cfb2c4ba99e5e38c
```

For example, in this case the statement SP_GETINVOICE_REPORT was run 46 times. It means that this query heavily affects the overall performance, and it should be optimized first.

When you click on Sort by summary, you will see the queries which took the most part of the time (among logged queries). These queries usually are the best candidates for the optimization.



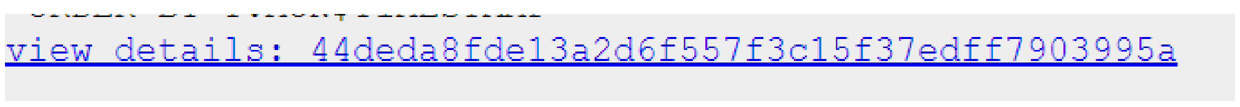
SORT BY SUMMARY SORT BY DURATION OR SORT BY FREQUENCY

RANK 1 of 1372; Summary: 1878743 of 14431580 ms; FREQUENCY: 1,46% (42 of 2871)

```
select distinct
        pedid.id_pedido
from pedid
```

Detailed information for the problematic SQL queries

To see details of the most frequent query, click in the link «View details» in the bottom of the query text:



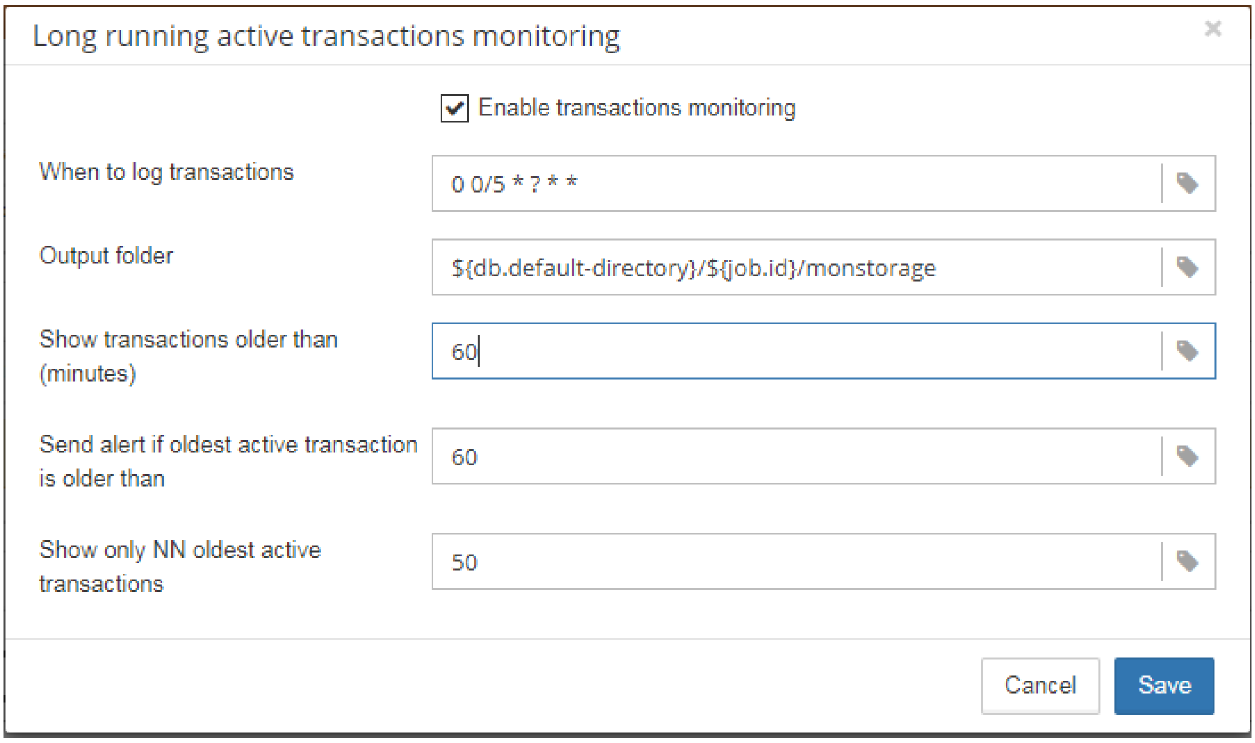
```
view details: 44deda8fde13a2d6f557f3c15f37edff7903995a
```

As a result, you will see the longest query among the queries with the same SQL text, with its execution plan, execution statistics and input parameters.

 This information is enough to analyze and optimize SQL query in Firebird SQL Studio or other developer IDE.

Automatic monitoring of long-running active transactions

On the «Performance» tab you can find the option to enable automatic monitoring of long active transactions:


Long running active transactions monitoring

☑ Enable transactions monitoring

When to log transactions	0 0/5 * ? * *
Output folder	${db.default-directory}/${job.id}/monstorage
Show transactions older than (minutes)	60
Send alert if oldest active transaction is older than	60
Show only NN oldest active transactions	50

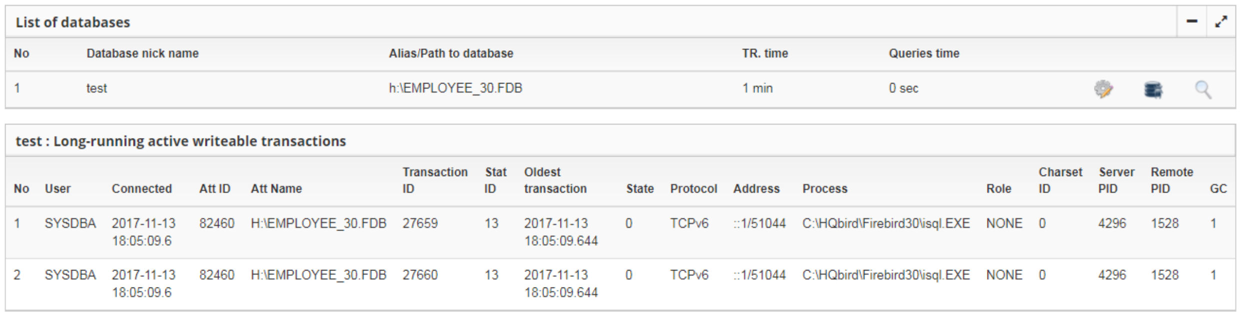
By default, this monitoring is off. To enable it, click on «Enable transactions monitoring». In general, it is enough, this monitoring does not require further setup.

Let's consider it's settings:

- When to log transactions: This parameter defines when to check MON$ tables for long-running active transactions. By default, it is set to run every 5 minutes (see CRON statement). You can make less often on heavy loaded databases, up to once per hour.
- Output folder: it is a service parameter.
- Show transactions older then (minutes): it specifies the time threshold to show the transaction (and associated connection) in the list of long-running-transactions. By default, this threshold is 60 minutes: it means that writeable transactions which started more than 1 hour ago, will be considered as long-running.
- Send alert if oldest active transaction is older then (minutes): the same, but it triggers alert and, if email notifications are enabled, the automatic email with the details of long running active transaction. The text of the alert looks like the following:

  ```
  There is a long running active transactions: it was started at 11/13/17 1:19 PM (and run at least 107 minutes}) from ::1/51068 by C:\HQbird\Firebird30\isql.exe. Such transactions block garbage collection, please perform transactions analysis with HQbird MonLogger.
  ```
- Show only NN oldest active transactions: it specifies how many records will be shown in the list with long-running transactions.

The list of long-running active transactions is shown on the screenshot below:



List of databases

No	Database nick name	Alias/Path to database	TR. time	Queries time
1	test	h:\EMPLOYEE_30.FDB	1 min	0 sec

test : Long-running active writeable transactions

No	User	Connected	Att ID	Att Name	Transaction ID	Stat ID	Oldest transaction	State	Protocol	Address	Process	Role	Charset ID	Server PID	Remote PID	GC
1	SYSDBA	2017-11-13 18:05:09.6	82460	H:\EMPLOYEE_30.FDB	27659	13	2017-11-13 18:05:09.644	0	TCPv6	::1/51044	C:\HQbird\Firebird30\isql.EXE	NONE	0	4296	1528	1
2	SYSDBA	2017-11-13 18:05:09.6	82460	H:\EMPLOYEE_30.FDB	27660	13	2017-11-13 18:05:09.644	0	TCPv6	::1/51044	C:\HQbird\Firebird30\isql.EXE	NONE	0	4296	1528	1

Here you can see that isql.exe started 2 long-running active transactions to the database h:\employee_30.fdb at November 13, 18-05.

How to select a tool for detailed monitoring

FBDataGuard is the first line of a defense for Firebird database; once FBDataGuard encounters something suspicious inside the monitored areas, it sends an alert with description of the issue.

Important! If you have several Firebird servers, we offer HQbird Control Center application which gathers alerts data from the Firebird servers and databases and shows them at the single screen. Contact our support@ib-aid.com for more details.

After receiving such alert from FBDataGuard the database administrator should proceed with detailed investigation of the problem.

The choice of tool for detailed monitoring depends on the type of detected problem.

If FBDataGuard reports long-running active transaction (Next-OAT), it is necessary to use **HQbird MonLogger** to detect the source of currently running active transaction.

If stuck of oldest interesting transaction is reported, database administrator must plan an explicit sweep to clean uncollected garbage with FBDataGuard sweep job (if it is necessary) and then plan tracking of forced rollbacks with HQbird Firebird Performance Monitor (**FBPerfMon**) or, if it is more relevant, with **FBScanner**.

If users report slowness problem with some queries, Perfusion or FBScanner should be used.

If there is unusual spikes in transaction behavior, **IBTransactionMonitor** can be a good addition to FBPerfMon or FBScanner to clarify the situation.

The problems with general database performance and occasional or periodic slowness require an analysis of database structure, which can be done only with HQbird Database Analyst.

Below we will consider how to work with HQbird monitoring tools in more details.

4.2. Monitoring with MON$ tables: HQbird MonLogger

HQbird MonLogger is a tool to analyze monitoring tables output in Firebird and find problems with slow SQL queries, wrongly designed transactions (long-running transactions, transactions with incorrect isolation level, etc) and identify problematic applications.

MonLogger can connect to Firebird database with performance problems and identify what is the reason of slowness: is it some user attachment, slow SQL query or long-running transaction?

MonLogger supports Firebird 2.1, 2.5 and 3.0 – for older Firebird versions or InterBase please use FBScanner.

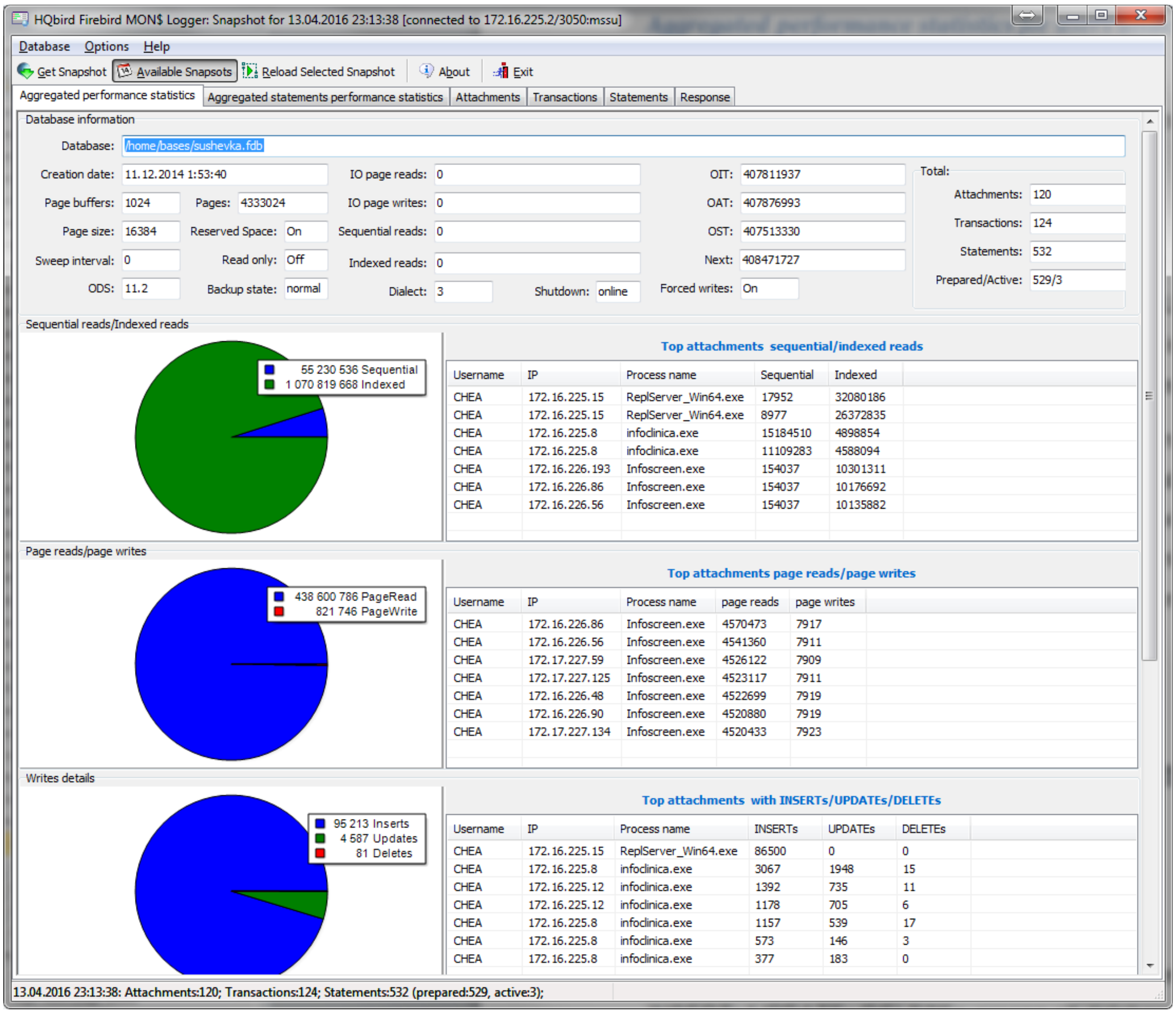
MonLogger can show you:

- Top attachments with highest number of IO operations, non-indexed and indexed reads
- Top SQL statements with highest number of IO operations, non-indexed and indexed reads
- Problematic transactions: long-running transactions, transactions with erroneous isolation level, read/write transactions, and related information: when they started, what applications started these transactions, from what IP address, etc
- Attachments and statements with the most intensive garbage collection actions
- Read/write ratio, INSERTS/UPDATE/DELETE ratio, and more.

After connection to the database where you want to find performance problems, several snapshots of monitoring tables should be done – click on «Get Snapshot» to take snapshot.

Aggregated performance statistics for users attachments

At the first screen we can see aggregated statistics for database connections, and identify connections with the biggest problems:



Database information

Database:	/home/bases/sushevka.fdb						
Creation date:	11.12.2014 1:53:40	IO page reads:	0	OIT:	407811937	Total:	
Page buffers:	1024	Pages:	4333024	IO page writes:	0	OAT:	407876993
Page size:	16384	Reserved Space:	On	Sequential reads:	0	OST:	407513330
Sweep interval:	0	Read only:	Off	Indexed reads:	0	Next:	408471727
ODS:	11.2	Backup state:	normal	Dialect:	3	Shutdown:	online
Forced writes:	On						

Total:

Attachments:	120
Transactions:	124
Statements:	532
Prepared/Active:	529/3

Sequential reads/Indexed reads

Top attachments sequential/indexed reads

Username	IP	Process name	Sequential	Indexed
CHEA	172.16.225.15	ReplServer_Win64.exe	17952	32080186
CHEA	172.16.225.15	ReplServer_Win64.exe	8977	26372835
CHEA	172.16.225.8	infoclinica.exe	15184510	4898854
CHEA	172.16.225.8	infoclinica.exe	11109283	4588094
CHEA	172.16.226.193	Infoscreen.exe	154037	10301311
CHEA	172.16.226.86	Infoscreen.exe	154037	10176692
CHEA	172.16.226.56	Infoscreen.exe	154037	10135882

Page reads/page writes

Top attachments page reads/page writes

Username	IP	Process name	page reads	page writes
CHEA	172.16.226.86	Infoscreen.exe	4570473	7917
CHEA	172.16.226.56	Infoscreen.exe	4541360	7911
CHEA	172.17.227.59	Infoscreen.exe	4526122	7909
CHEA	172.17.227.125	Infoscreen.exe	4523117	7911
CHEA	172.16.226.48	Infoscreen.exe	4522699	7919
CHEA	172.16.226.90	Infoscreen.exe	4520880	7919
CHEA	172.17.227.134	Infoscreen.exe	4520433	7923

Writes details

Top attachments with INSERTs/UPDATEs/DELETEs

Username	IP	Process name	INSERTs	UPDATEs	DELETEs
CHEA	172.16.225.15	ReplServer_Win64.exe	86500	0	0
CHEA	172.16.225.8	infoclinica.exe	3067	1948	15
CHEA	172.16.225.12	infoclinica.exe	1392	735	11
CHEA	172.16.225.12	infoclinica.exe	1178	705	6
CHEA	172.16.225.8	infoclinica.exe	1157	539	17
CHEA	172.16.225.8	infoclinica.exe	573	146	3
CHEA	172.16.225.8	infoclinica.exe	377	183	0

13.04.2016 23:13:38: Attachments:120; Transactions:124; Statements:532 (prepared:529, active:3);

Sequential reads / Indexed reads

"Sequential reads / Indexed reads" shows us total ratio between sequential (non-indexed) reads and indexed reads in application. Usually number of non-indexed reads should be low, so big percent of sequential reads is sign that many SQL queries have NATURAL execution plans, and they could be a reason of slow response time.

Click on record in «TOP attachments: sequential/indexed reads» will bring you to tab «Attachments», where you can see more details about Attachment, and then jump to tab «Transactions» or «Statements», where you will see transactions and attachments linked with selected attachment (if checkmark «Link to selected attachment» is on, otherwise all transactions/statements for all attachments will be shown).

Write details

«Write details» gives you an overview of write operations: ratio between INSERTs/UPDATEs/DELETEs among all database attachments. In the table of top writers you can see attachments with the biggest number of write operations. It is useful to identify applications or software modules which performs excessive number of update or deletes (which are the most dangerous operations in terms of garbage collections).

Garbage collection details

What garbage collection operations mean?

- Purge – engine removes back-versions, only primary version is in database.
- Expunge – both primary version and all back-versions were deleted.
- Back-out – remove only primary version (due to rollback).

Usually we can associate purge with UPDATE operation, Expunge with DELETE, and Backout with rollback of INSERT or UPDATE. Many backouts could mean that there is a problem with transaction management in the application.

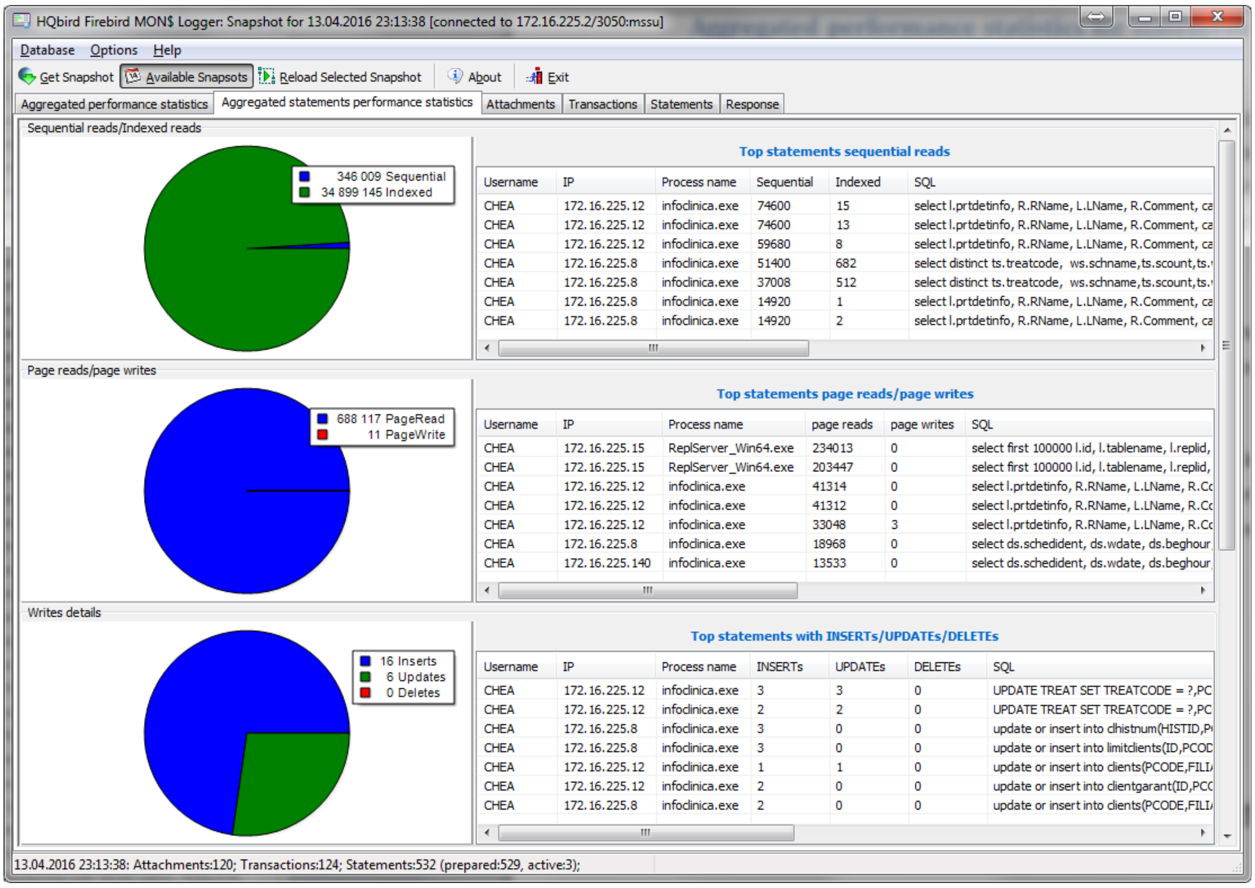
Memory usage

«Memory usage» graph shows us total memory used by all active attachments now, and peak of allocated memory for them in the past.

List of top attachments by memory usage shows us the biggest memory consumers among your attachments. It is useful to find applications or software modules with excessive memory usage.

Aggregated performance statistics for statements

At the second tab you can find aggregated performance statistics for statements.



Top statements sequential reads

Username	IP	Process name	Sequential	Indexed	SQL
CHEA	172.16.225.12	infoclinica.exe	74600	15	select l.prtdetinfo, R.RName, L.LName, R.Comment, ca
CHEA	172.16.225.12	infoclinica.exe	74600	13	select l.prtdetinfo, R.RName, L.LName, R.Comment, ca
CHEA	172.16.225.12	infoclinica.exe	59680	8	select l.prtdetinfo, R.RName, L.LName, R.Comment, ca
CHEA	172.16.225.8	infoclinica.exe	51400	682	select distinct ts.treatcode, ws.schname,ts.scount,ts.
CHEA	172.16.225.8	infoclinica.exe	37008	512	select distinct ts.treatcode, ws.schname,ts.scount,ts.
CHEA	172.16.225.8	infoclinica.exe	14920	1	select l.prtdetinfo, R.RName, L.LName, R.Comment, ca
CHEA	172.16.225.8	infoclinica.exe	14920	2	select l.prtdetinfo, R.RName, L.LName, R.Comment, ca

Top statements page reads/page writes

Username	IP	Process name	page reads	page writes	SQL
CHEA	172.16.225.15	ReplServer_Win64.exe	234013	0	select first 100000 l.id, l.tablename, l.replid,
CHEA	172.16.225.15	ReplServer_Win64.exe	203447	0	select first 100000 l.id, l.tablename, l.replid,
CHEA	172.16.225.12	infoclinica.exe	41314	0	select l.prtdetinfo, R.RName, L.LName, R.Co
CHEA	172.16.225.12	infoclinica.exe	41312	0	select l.prtdetinfo, R.RName, L.LName, R.Co
CHEA	172.16.225.12	infoclinica.exe	33048	3	select l.prtdetinfo, R.RName, L.LName, R.Co
CHEA	172.16.225.8	infoclinica.exe	18968	0	select ds.schedident, ds.wdate, ds.beghour
CHEA	172.16.225.140	infoclinica.exe	13533	0	select ds.schedident, ds.wdate, ds.beghour

Top statements with INSERTs/UPDATEs/DELETEs

Username	IP	Process name	INSERTs	UPDATEs	DELETEs	SQL
CHEA	172.16.225.12	infoclinica.exe	3	3	0	UPDATE TREAT SET TREATCODE = ?,PC
CHEA	172.16.225.12	infoclinica.exe	2	2	0	UPDATE TREAT SET TREATCODE = ?,PC
CHEA	172.16.225.8	infoclinica.exe	3	0	0	update or insert into clhistnum(HISTID,P
CHEA	172.16.225.8	infoclinica.exe	3	0	0	update or insert into limitclients(ID,PCOD
CHEA	172.16.225.12	infoclinica.exe	1	1	0	update or insert into clients(PCODE,FILI
CHEA	172.16.225.12	infoclinica.exe	2	0	0	update or insert into clientgarant(ID,PCC
CHEA	172.16.225.8	infoclinica.exe	2	0	0	update or insert into clients(PCODE,FILI

13.04.2016 23:13:38: Attachments:120; Transactions:124; Statements:532 (prepared:529, active:3);

This statistics better reflects the momentary situation in the database – since monitoring tables collect information since the beginning of each object life, statements you can see here are those which were running during the moment when snapshot was taken.

Sequential reads / Indexed reads

In this list we can see top statements which perform many sequential reads from the database. Usually such statements require SQL tuning – either through indices tuning, or through SQL query redesign.

To tune the query, check its execution plan: usually it is possible to improve query speed by eliminating NATURAL in plans with new indices or query redesign. Click on the statement in this list to open tab «Statements», where you can find more details about selected statement, and jump to associated transaction or attachment.

Page reads/page writes

This graphs and list shows brief information about top statements which perform many reads – it means that they consume significant IO and can affect performance of other queries. SQL statements with peak values should be carefully checked for optimal performance.

Write details for statements

At this graph you can see what writing SQL statements were doing at the moment when snapshot of monitoring tables was taken, and identify UPDATES and DELETEs which made many changes in the database

Garbage collection details for statements

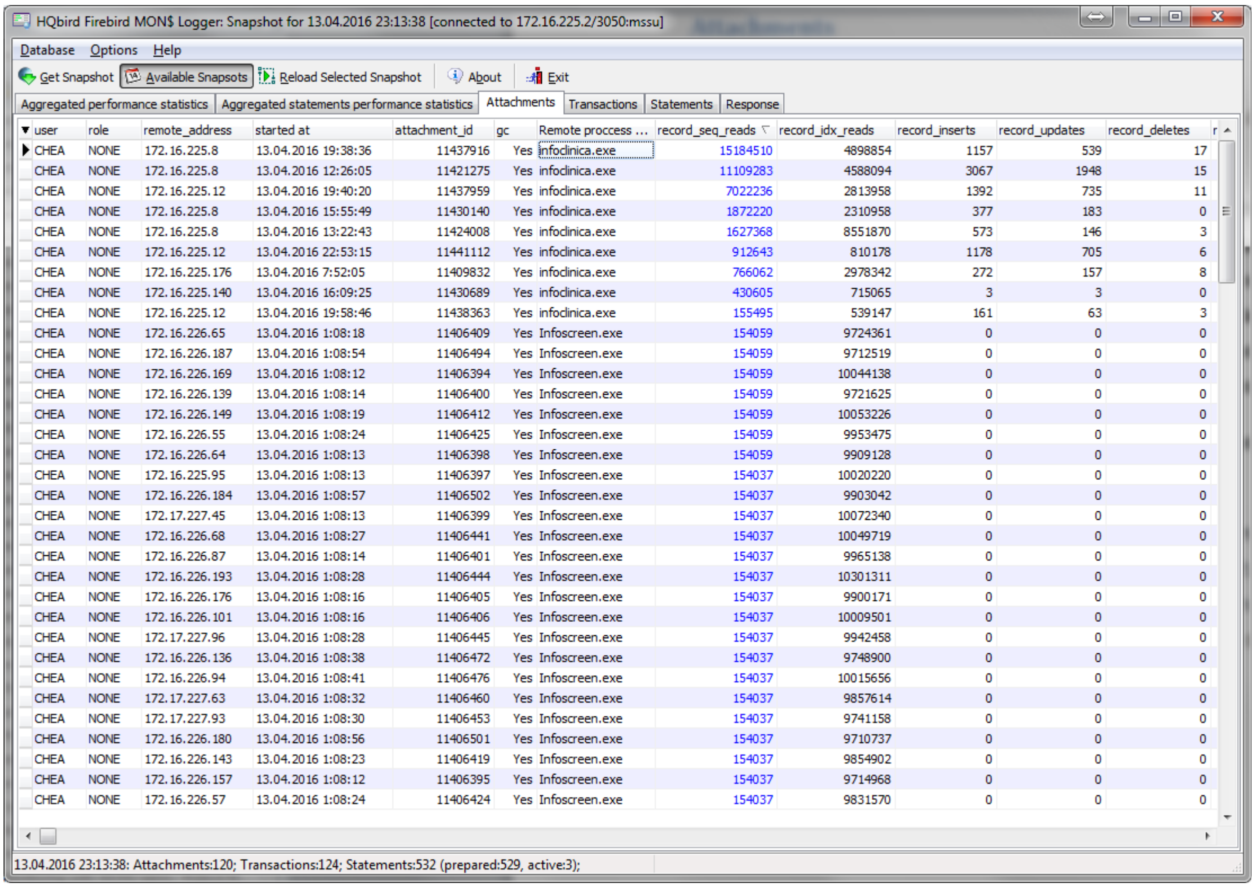
At this graph we can see how many garbage collection operations were done by statements running at the moment of snapshot.

Memory usage for statements

Unlike aggregated memory usage statistics for attachment, statements' memory usage can show us list of exact statements which consume a lot of memory at the moment.

Attachments

The third tab is «Attachments». You can open this tab directly to jump there by clicking one of the records at «Aggregated performance statistics».



HQbird Firebird MON$ Logger: Snapshot for 13.04.2016 23:13:38 [connected to 172.16.225.2/3050:mssu]
Database
Options
Help
Get Snapshot
Available Snapsots
Reload Selected Snapshot
About
Exit
Aggregated performance statistics
Aggregated statements performance statistics
Attachments
Transactions
Statements
Response

user	role	remote_address	started at	attachment_id	gc	Remote proccess ...	record_seq_reads	record_idx_reads	record_inserts	record_updates	record_deletes	r
CHEA	NONE	172.16.225.8	13.04.2016 19:38:36	11437916	Yes	infoclinica.exe	15184510	4898854	1157	539	17	
CHEA	NONE	172.16.225.8	13.04.2016 12:26:05	11421275	Yes	infoclinica.exe	11109283	4588094	3067	1948	15	
CHEA	NONE	172.16.225.12	13.04.2016 19:40:20	11437959	Yes	infoclinica.exe	7022236	2813958	1392	735	11	
CHEA	NONE	172.16.225.8	13.04.2016 15:55:49	11430140	Yes	infoclinica.exe	1872220	2310958	377	183	0	
CHEA	NONE	172.16.225.8	13.04.2016 13:22:43	11424008	Yes	infoclinica.exe	1627368	8551870	573	146	3	
CHEA	NONE	172.16.225.12	13.04.2016 22:53:15	11441112	Yes	infoclinica.exe	912643	810178	1178	705	6	
CHEA	NONE	172.16.225.176	13.04.2016 7:52:05	11409832	Yes	infoclinica.exe	766062	2978342	272	157	8	
CHEA	NONE	172.16.225.140	13.04.2016 16:09:25	11430689	Yes	infoclinica.exe	430605	715065	3	3	0	
CHEA	NONE	172.16.225.12	13.04.2016 19:58:46	11438363	Yes	infoclinica.exe	155495	539147	161	63	3	
CHEA	NONE	172.16.226.65	13.04.2016 1:08:18	11406409	Yes	Infoscreen.exe	154059	9724361	0	0	0	
CHEA	NONE	172.16.226.187	13.04.2016 1:08:54	11406494	Yes	Infoscreen.exe	154059	9712519	0	0	0	
CHEA	NONE	172.16.226.169	13.04.2016 1:08:12	11406394	Yes	Infoscreen.exe	154059	10044138	0	0	0	
CHEA	NONE	172.16.226.139	13.04.2016 1:08:14	11406400	Yes	Infoscreen.exe	154059	9721625	0	0	0	
CHEA	NONE	172.16.226.149	13.04.2016 1:08:19	11406412	Yes	Infoscreen.exe	154059	10053226	0	0	0	
CHEA	NONE	172.16.226.55	13.04.2016 1:08:24	11406425	Yes	Infoscreen.exe	154059	9953475	0	0	0	
CHEA	NONE	172.16.226.64	13.04.2016 1:08:13	11406398	Yes	Infoscreen.exe	154059	9909128	0	0	0	
CHEA	NONE	172.16.225.95	13.04.2016 1:08:13	11406397	Yes	Infoscreen.exe	154037	10020220	0	0	0	
CHEA	NONE	172.16.226.184	13.04.2016 1:08:57	11406502	Yes	Infoscreen.exe	154037	9903042	0	0	0	
CHEA	NONE	172.17.227.45	13.04.2016 1:08:13	11406399	Yes	Infoscreen.exe	154037	10072340	0	0	0	
CHEA	NONE	172.16.226.68	13.04.2016 1:08:27	11406441	Yes	Infoscreen.exe	154037	10049719	0	0	0	
CHEA	NONE	172.16.226.87	13.04.2016 1:08:14	11406401	Yes	Infoscreen.exe	154037	9965138	0	0	0	
CHEA	NONE	172.16.226.193	13.04.2016 1:08:28	11406444	Yes	Infoscreen.exe	154037	10301311	0	0	0	
CHEA	NONE	172.16.226.176	13.04.2016 1:08:16	11406405	Yes	Infoscreen.exe	154037	9900171	0	0	0	
CHEA	NONE	172.16.226.101	13.04.2016 1:08:16	11406406	Yes	Infoscreen.exe	154037	10009501	0	0	0	
CHEA	NONE	172.17.227.96	13.04.2016 1:08:28	11406445	Yes	Infoscreen.exe	154037	9942458	0	0	0	
CHEA	NONE	172.16.226.136	13.04.2016 1:08:38	11406472	Yes	Infoscreen.exe	154037	9748900	0	0	0	
CHEA	NONE	172.16.226.94	13.04.2016 1:08:41	11406476	Yes	Infoscreen.exe	154037	10015656	0	0	0	
CHEA	NONE	172.17.227.63	13.04.2016 1:08:32	11406460	Yes	Infoscreen.exe	154037	9857614	0	0	0	
CHEA	NONE	172.17.227.93	13.04.2016 1:08:30	11406453	Yes	Infoscreen.exe	154037	9741158	0	0	0	
CHEA	NONE	172.16.226.180	13.04.2016 1:08:56	11406501	Yes	Infoscreen.exe	154037	9710737	0	0	0	
CHEA	NONE	172.16.226.143	13.04.2016 1:08:23	11406419	Yes	Infoscreen.exe	154037	9854902	0	0	0	
CHEA	NONE	172.16.226.157	13.04.2016 1:08:12	11406395	Yes	Infoscreen.exe	154037	9714968	0	0	0	
CHEA	NONE	172.16.226.57	13.04.2016 1:08:24	11406424	Yes	Infoscreen.exe	154037	9831570	0	0	0	

13.04.2016 23:13:38: Attachments:120; Transactions:124; Statements:532 (prepared:529, active:3);

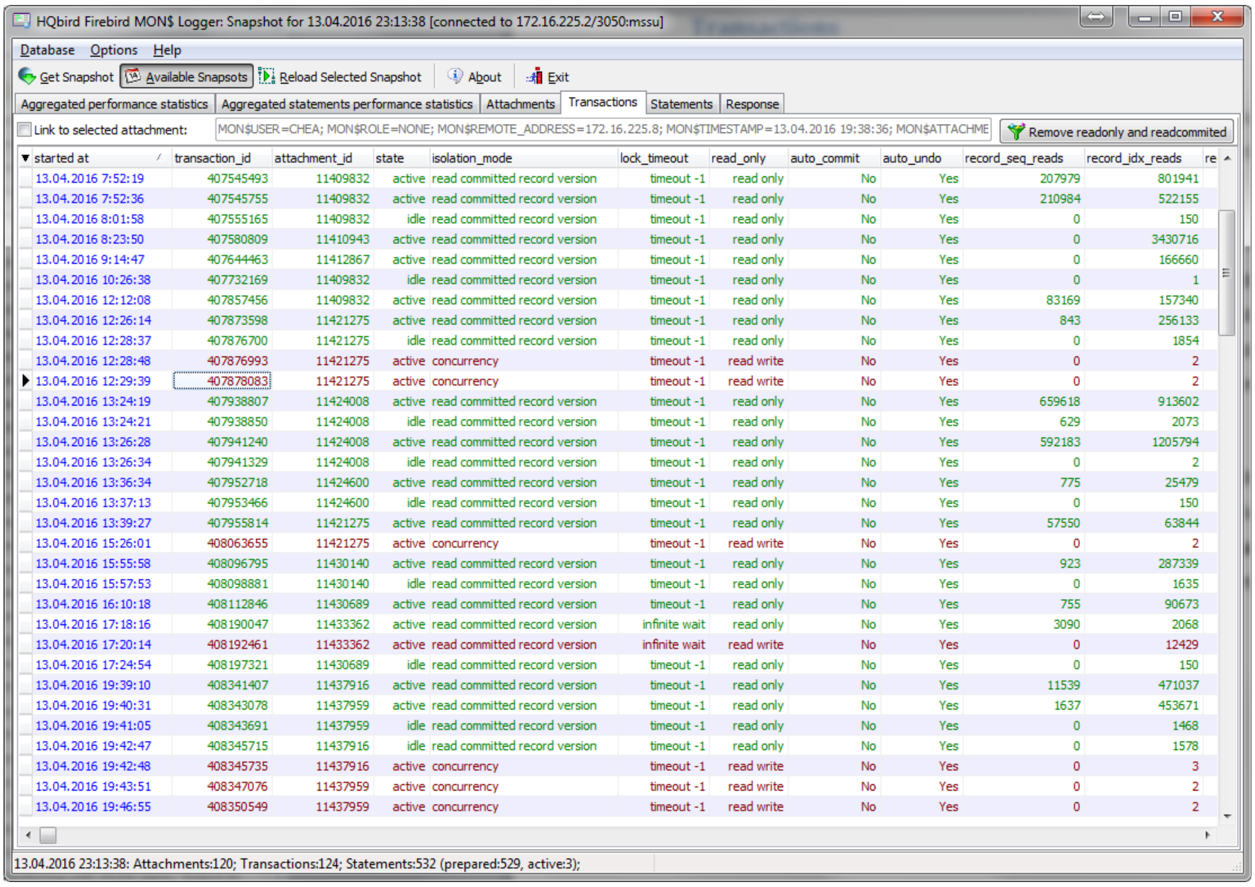
«Attachments» shows the list of users connected to the Firebird database, with many useful details: USER and ROLE of attachment, start time and ID of attachment, is there garbage collection enabled for the attachment, name of remote process which established an attachment, and several accumulated performance counters for the attachment: number of sequential reads [done by attachment since its start], number of indexed reads, number of inserts, updates and deletes, as well records backouts, purges and expunges.

By default some of columns of attachment are switched off, to show only most important information.

Of course, every time you click on attachment, you can jump to transactions running inside it, and then to statements. There is a checkbox in the left upper corner of Transactions and Statements tabs, which controls the behavior – when checked, only transactions and statements marked by selected attachment ID, will be shown.

Transactions

Tab «Transactions» shows active transactions at the moment when snapshot was taken.


HQbird Firebird MON$ Logger: Snapshot for 13.04.2016 23:13:38 [connected to 172.16.225.2/3050:mssu]
Database
Options
Help
Get Snapshot
Available Snapsots
Reload Selected Snapshot
About
Exit
Aggregated performance statistics
Aggregated statements performance statistics
Attachments
Transactions
Statements
Response
Link to selected attachment:
Remove readonly and readcommited

MON$USER=CHEA; MON$ROLE=NONE; MON$REMOTE_ADDRESS=172.16.225.8; MON$TIMESTAMP=13.04.2016 19:38:36; MON$ATTACHME

started at	transaction_id	attachment_id	state	isolation_mode	lock_timeout	read_only	auto_commit	auto_undo	record_seq_reads	record_idx_reads	re
13.04.2016 7:52:19	407545493	11409832	active	read committed record version	timeout -1	read only	No	Yes	207979	801941	
13.04.2016 7:52:36	407545755	11409832	active	read committed record version	timeout -1	read only	No	Yes	210984	522155	
13.04.2016 8:01:58	407555165	11409832	idle	read committed record version	timeout -1	read only	No	Yes	0	150	
13.04.2016 8:23:50	407580809	11410943	active	read committed record version	timeout -1	read only	No	Yes	0	3430716	
13.04.2016 9:14:47	407644463	11412867	active	read committed record version	timeout -1	read only	No	Yes	0	166660	
13.04.2016 10:26:38	407732169	11409832	idle	read committed record version	timeout -1	read only	No	Yes	0	1	
13.04.2016 12:12:08	407857456	11409832	active	read committed record version	timeout -1	read only	No	Yes	83169	157340	
13.04.2016 12:26:14	407873598	11421275	active	read committed record version	timeout -1	read only	No	Yes	843	256133	
13.04.2016 12:28:37	407876700	11421275	idle	read committed record version	timeout -1	read only	No	Yes	0	1854	
13.04.2016 12:28:48	407876993	11421275	active	concurrency	timeout -1	read write	No	Yes	0	2	
13.04.2016 12:29:39	407878083	11421275	active	concurrency	timeout -1	read write	No	Yes	0	2	
13.04.2016 13:24:19	407938807	11424008	active	read committed record version	timeout -1	read only	No	Yes	659618	913602	
13.04.2016 13:24:21	407938850	11424008	idle	read committed record version	timeout -1	read only	No	Yes	629	2073	
13.04.2016 13:26:28	407941240	11424008	active	read committed record version	timeout -1	read only	No	Yes	592183	1205794	
13.04.2016 13:26:34	407941329	11424008	idle	read committed record version	timeout -1	read only	No	Yes	0	2	
13.04.2016 13:36:34	407952718	11424600	active	read committed record version	timeout -1	read only	No	Yes	775	25479	
13.04.2016 13:37:13	407953466	11424600	idle	read committed record version	timeout -1	read only	No	Yes	0	150	
13.04.2016 13:39:27	407955814	11421275	active	read committed record version	timeout -1	read only	No	Yes	57550	63844	
13.04.2016 15:26:01	408063655	11421275	active	concurrency	timeout -1	read write	No	Yes	0	2	
13.04.2016 15:55:58	408096795	11430140	active	read committed record version	timeout -1	read only	No	Yes	923	287339	
13.04.2016 15:57:53	408098881	11430140	idle	read committed record version	timeout -1	read only	No	Yes	0	1635	
13.04.2016 16:10:18	408112846	11430689	active	read committed record version	timeout -1	read only	No	Yes	755	90673	
13.04.2016 17:18:16	408190047	11433362	active	read committed record version	infinite wait	read only	No	Yes	3090	2068	
13.04.2016 17:20:14	408192461	11433362	active	read committed record version	infinite wait	read write	No	Yes	0	12429	
13.04.2016 17:24:54	408197321	11430689	idle	read committed record version	timeout -1	read only	No	Yes	0	150	
13.04.2016 19:39:10	408341407	11437916	active	read committed record version	timeout -1	read only	No	Yes	11539	471037	
13.04.2016 19:40:31	408343078	11437959	active	read committed record version	timeout -1	read only	No	Yes	1637	453671	
13.04.2016 19:41:05	408343691	11437959	idle	read committed record version	timeout -1	read only	No	Yes	0	1468	
13.04.2016 19:42:47	408345715	11437916	idle	read committed record version	timeout -1	read only	No	Yes	0	1578	
13.04.2016 19:42:48	408345735	11437916	active	concurrency	timeout -1	read write	No	Yes	0	3	
13.04.2016 19:43:51	408347076	11437959	active	concurrency	timeout -1	read write	No	Yes	0	2	
13.04.2016 19:46:55	408350549	11437959	active	concurrency	timeout -1	read write	No	Yes	0	2	

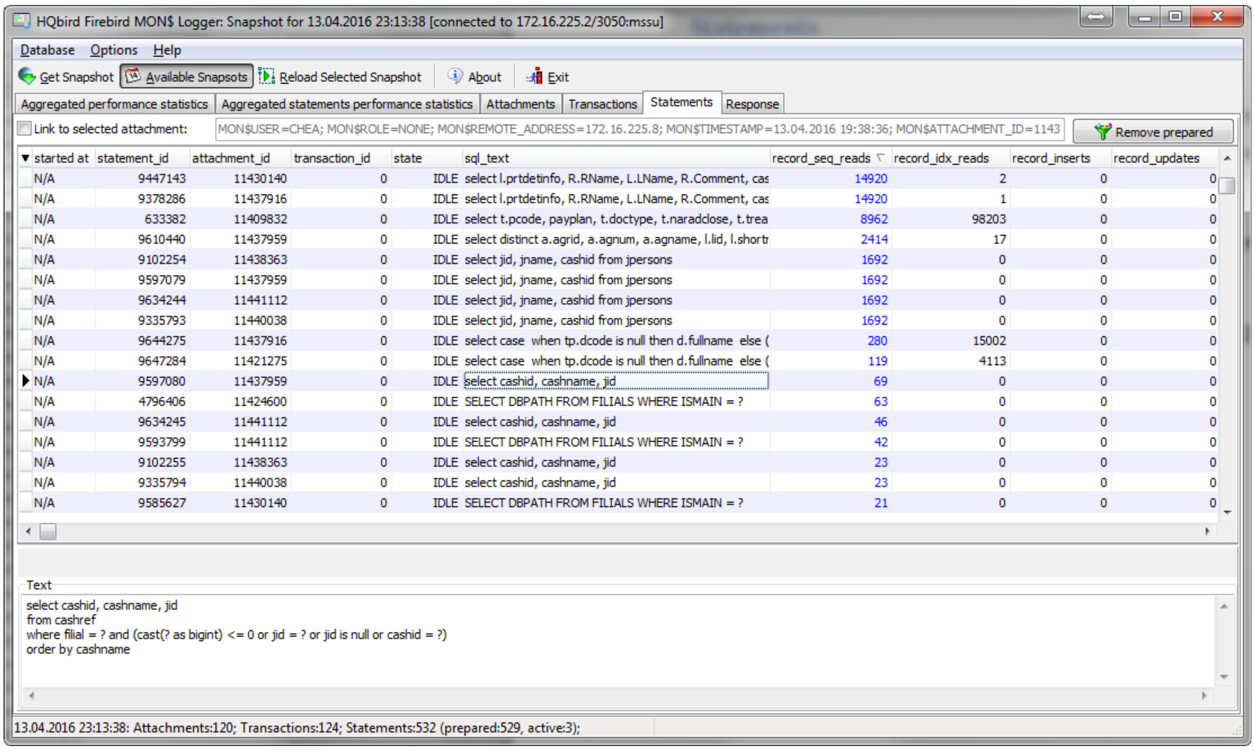
13.04.2016 23:13:38: Attachments:120; Transactions:124; Statements:532 (prepared:529, active:3);

If checkbox «Link to selected attachment» is enabled, only transactions for selected attachment will be shown, otherwise all transactions are shown.

One of the most important characteristics is a lifetime of transactions: since Firebird is designed to work with short write transactions, it is important to keep them as short as possible. MonLogger highlights transactions with isolation modes and read-write settings which hold Oldest Active transaction and therefore provoke excessive record versions to be not cleared. If you see such transaction and it started a while ago, it means that it can be responsible for excessive records versions.

Sort on «started at» column and look for old transactions, marked in red: all writeable transactions and read only snapshots stuck Oldest Active Transaction and provoke excessive record versions to be hold. Identify where these transactions started (right-click and select «View parent attachment») and fix your code to commit this transaction earlier.

Statements


HQbird Firebird MON$ Logger: Snapshot for 13.04.2016 23:13:38 [connected to 172.16.225.2/3050:mssu]
Database
Options
Help
Get Snapshot
Available Snapsots
Reload Selected Snapshot
About
Exit
Aggregated performance statistics
Aggregated statements performance statistics
Attachments
Transactions
Statements
Response
Link to selected attachment:
Remove prepared

MON$USER=CHEA; MON$ROLE=NONE; MON$REMOTE_ADDRESS=172.16.225.8; MON$TIMESTAMP=13.04.2016 19:38:36; MON$ATTACHMENT_ID=1143

started at	statement_id	attachment_id	transaction_id	state	sql_text	record_seq_reads	record_idx_reads	record_inserts	record_updates
N/A	9447143	11430140	0	IDLE	select l.prtdetinfo, R.RName, L.LName, R.Comment, cas	14920	2	0	0
N/A	9378286	11437916	0	IDLE	select l.prtdetinfo, R.RName, L.LName, R.Comment, cas	14920	1	0	0
N/A	633382	11409832	0	IDLE	select t.pcode, payplan, t.doctype, t.naradclose, t.trea	8962	98203	0	0
N/A	9610440	11437959	0	IDLE	select distinct a.agrid, a.agnum, a.agname, l.lid, l.shortr	2414	17	0	0
N/A	9102254	11438363	0	IDLE	select jid, jname, cashid from jpersons	1692	0	0	0
N/A	9597079	11437959	0	IDLE	select jid, jname, cashid from jpersons	1692	0	0	0
N/A	9634244	11441112	0	IDLE	select jid, jname, cashid from jpersons	1692	0	0	0
N/A	9335793	11440038	0	IDLE	select jid, jname, cashid from jpersons	1692	0	0	0
N/A	9644275	11437916	0	IDLE	select case when tp.dcode is null then d.fullname else (	280	15002	0	0
N/A	9647284	11421275	0	IDLE	select case when tp.dcode is null then d.fullname else (	119	4113	0	0
N/A	9597080	11437959	0	IDLE	select cashid, cashname, jid	69	0	0	0
N/A	4796406	11424600	0	IDLE	SELECT DBPATH FROM FILIALS WHERE ISMAIN = ?	63	0	0	0
N/A	9634245	11441112	0	IDLE	select cashid, cashname, jid	46	0	0	0
N/A	9593799	11441112	0	IDLE	SELECT DBPATH FROM FILIALS WHERE ISMAIN = ?	42	0	0	0
N/A	9102255	11438363	0	IDLE	select cashid, cashname, jid	23	0	0	0
N/A	9335794	11440038	0	IDLE	select cashid, cashname, jid	23	0	0	0
N/A	9585627	11430140	0	IDLE	SELECT DBPATH FROM FILIALS WHERE ISMAIN = ?	21	0	0	0

Text

```
select cashid, cashname, jid
from cashref
where filial = ? and (cast(? as bigint) <= 0 or jid = ? or jid is null or cashid = ?)
order by cashname
```

13.04.2016 23:13:38: Attachments:120; Transactions:124; Statements:532 (prepared:529, active:3);

Tab «Statements» shows statements active at the moment of snapshot: if you need to catch all statements FBPerfMon or FBScanner should be used (all these tools are part of IBSurgeon Optimization Pack).

If «Link to selected attachment» is enabled, only statements for specific attachment will be shown, otherwise all active statements are in the list.

Some statements have no associated transaction id (=0): these queries are prepared, but not executed.

4.3. Monitoring with HQbird FBScanner

What is FBScanner?

FBScanner (Firebird Scanner) is a tool included in HQbird advanced distribution of Firebird, which 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, and 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/XE3). 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 disconnections 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

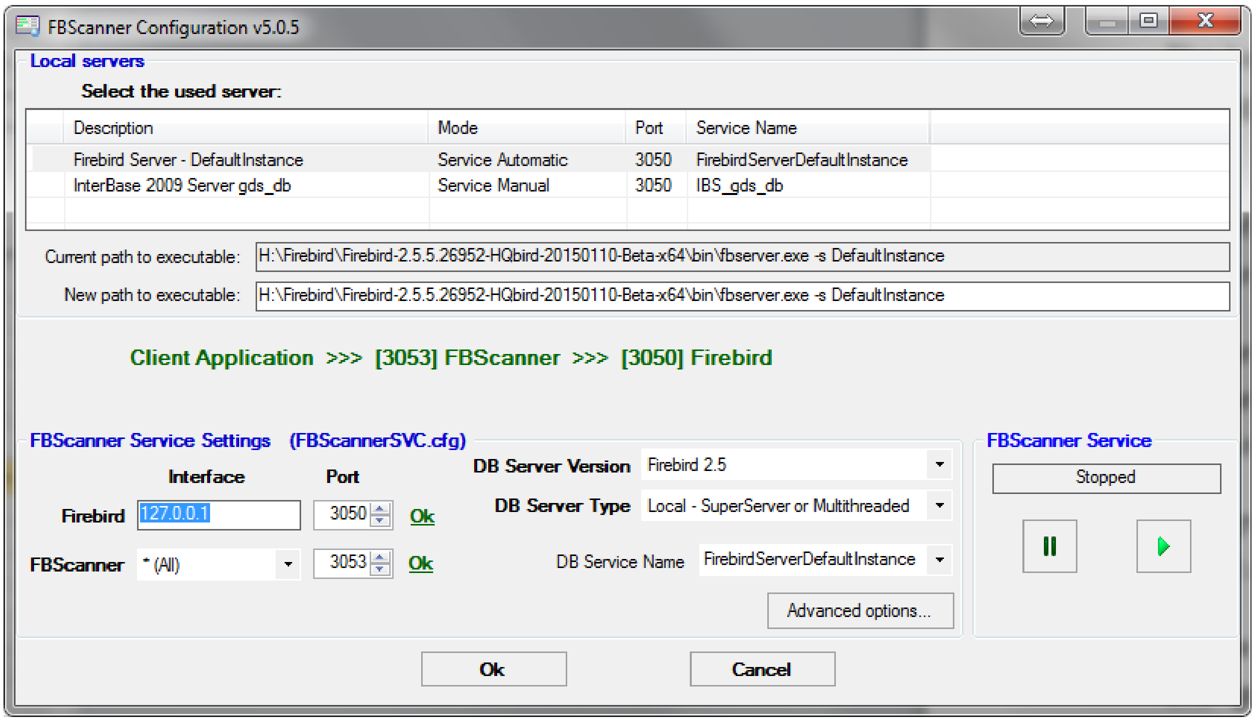
FBScanner does not change anything in transferred SQL traffic and works simply like a transparent proxy, so all applications will work normally.

FBScanner consume approximately 50-150Mb of memory (for 30-100 active clients), it is known that FBScanner adds approximately 150ms for every SQL statement.

How to configure FBScanner for local computer?

To configure FBScanner start «FBScanner Service Settings» from Start menu (IBSurgeon\HQbird Server Side\Firebird SQL Scanner\).This tool will help you to setup both basic and advanced configuration parameters for FBScanner.

The 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.



Description	Mode	Port	Service Name
Firebird Server - DefaultInstance	Service Automatic	3050	FirebirdServerDefaultInstance
InterBase 2009 Server gds_db	Service Manual	3050	IBS_gds_db

Current path to executable: H:\Firebird\Firebird-2.5.5.26952-HQbird-20150110-Beta-x64\bin\fbserver.exe -s DefaultInstance
New path to executable: H:\Firebird\Firebird-2.5.5.26952-HQbird-20150110-Beta-x64\bin\fbserver.exe -s DefaultInstance

Client Application >>> [3053] FBScanner >>> [3050] Firebird

FBScanner Service Settings (FBScannerSVC.cfg)

	Interface	Port	
Firebird	127.0.0.1	3050	Ok
FBScanner	* (All)	3053	Ok

DB Server Version: Firebird 2.5
DB Server Type: Local - SuperServer or Multithreaded
DB Service Name: FirebirdServerDefaultInstance

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).

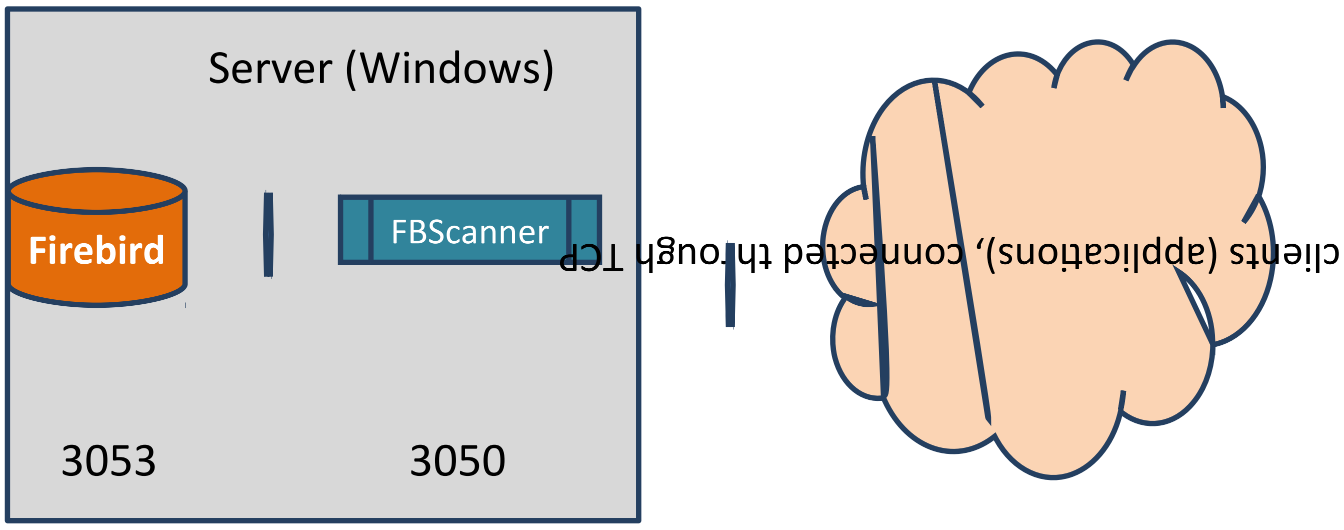
To approve the changes, click "Ok", otherwise "Cancel".

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 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 though 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:

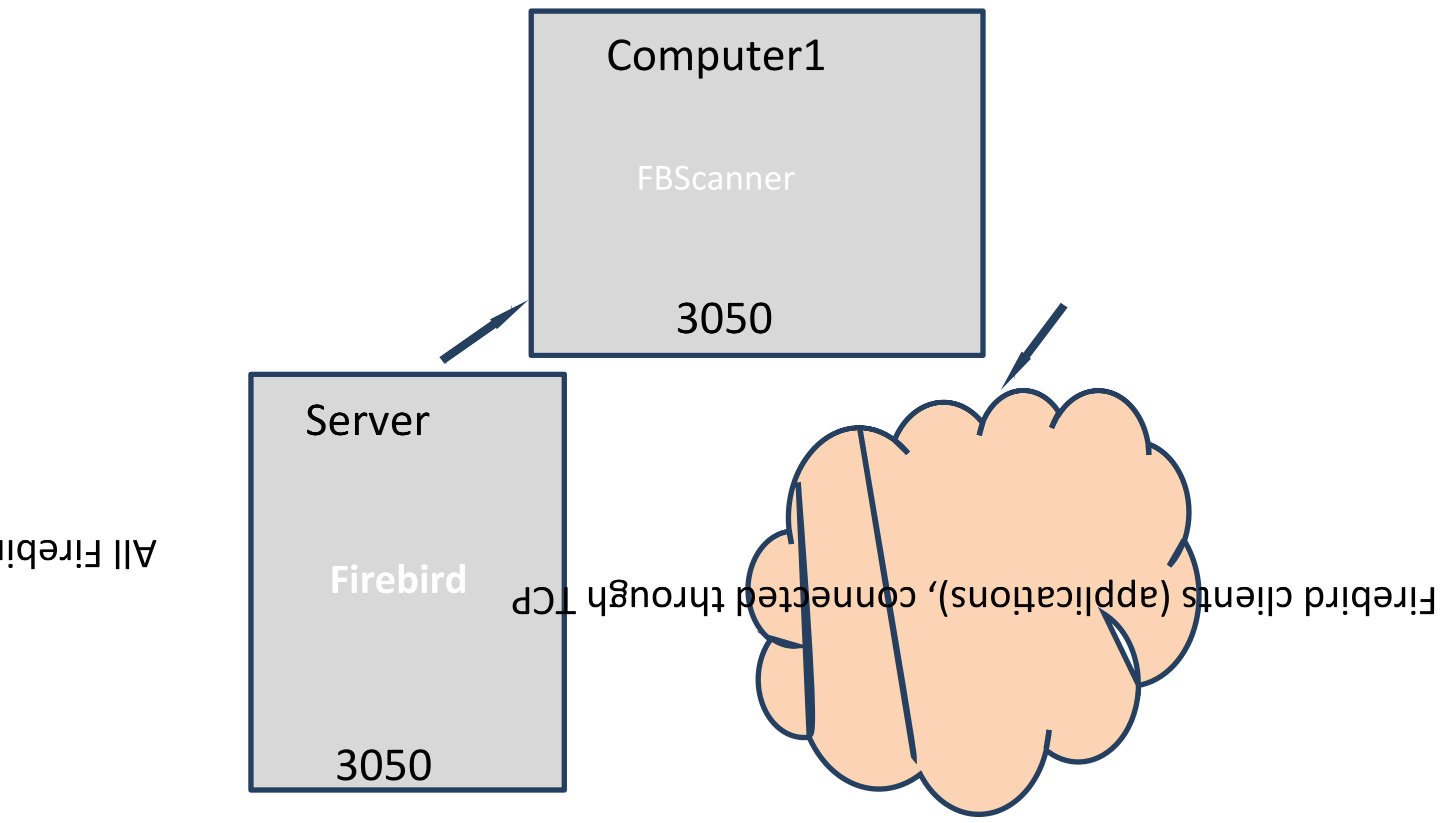
- Only several (may be, the single workstation) workstations need to be profiled/logged
- Only certain application or narrow functionality need to be profiled
- Developers need to check some SQL code on the live database – gather SQL log with execution statistics, plans, etc.
- Heavy load (too many workstations). 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

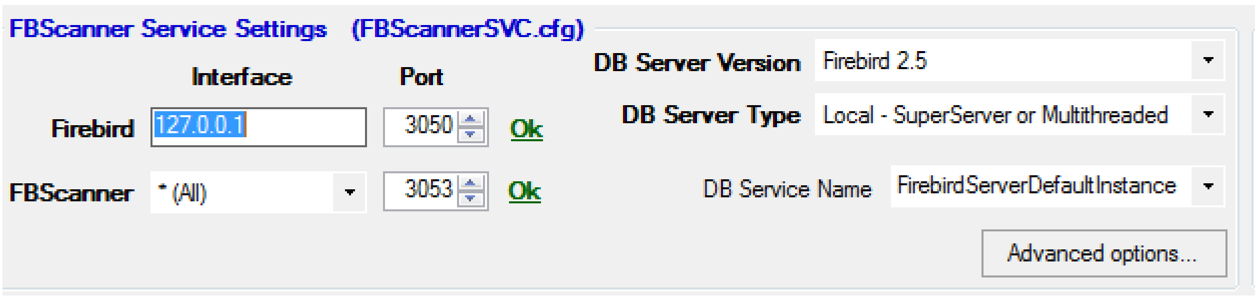
LogViewer) or afterwards (with FBScanner LogAnalyzer) all SQLs from own computer to the Firebird server..

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 2.5 example we considered above):


FBScanner Service Settings (FBScannerSVC.cfg)

	Interface	Port	
Firebird	127.0.0.1	3050	Ok
FBScanner	* (All)	3053	Ok

DB Server Version: Firebird 2.5
DB Server Type: Local - SuperServer or Multithreaded
DB Service Name: FirebirdServerDefaultInstance

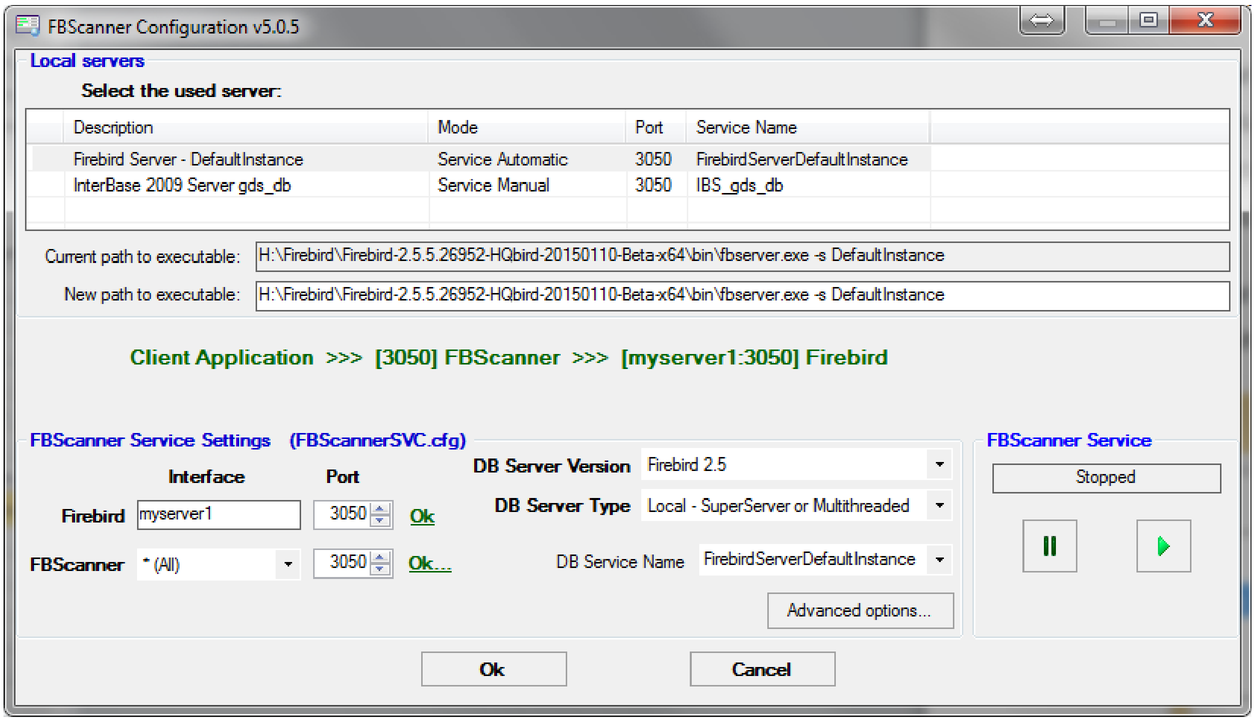
In order to setup FBScanner to route SQL traffic to the remote Firebird, we need to change "Server Type" from "Local…" to "Remote". It will change the main screen of the configuration tool.

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” text box.

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.


FBScanner Configuration v5.0.5

Local servers

Select the used server:

Description	Mode	Port	Service Name
Firebird Server - DefaultInstance	Service Automatic	3050	FirebirdServerDefaultInstance
InterBase 2009 Server gds_db	Service Manual	3050	IBS_gds_db

Current path to executable: H:\Firebird\Firebird-2.5.5.26952-HQbird-20150110-Beta-x64\bin\fbserver.exe -s DefaultInstance

New path to executable: H:\Firebird\Firebird-2.5.5.26952-HQbird-20150110-Beta-x64\bin\fbserver.exe -s DefaultInstance

Client Application >>> [3050] FBScanner >>> [myserver1:3050] Firebird

FBScanner Service Settings (FBScannerSVC.cfg)

	Interface	Port	
Firebird	myserver1	3050	Ok
FBScanner	* (All)	3050	Ok...

DB Server Version: Firebird 2.5

DB Server Type: Local - SuperServer or Multithreaded

DB Service Name: FirebirdServerDefaultInstance

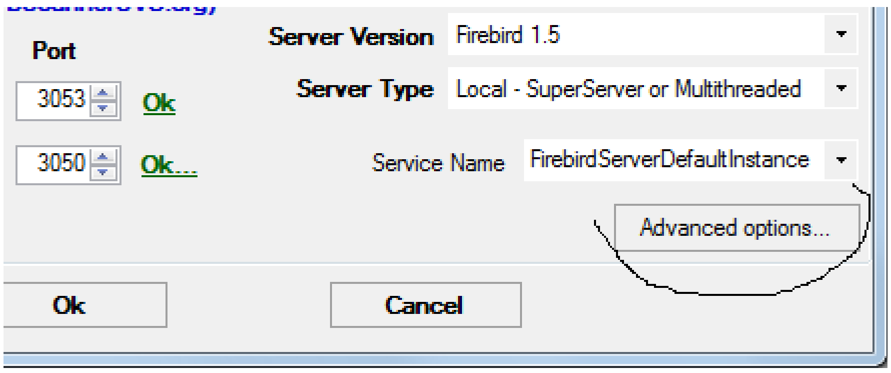
FBScanner Service: Stopped

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

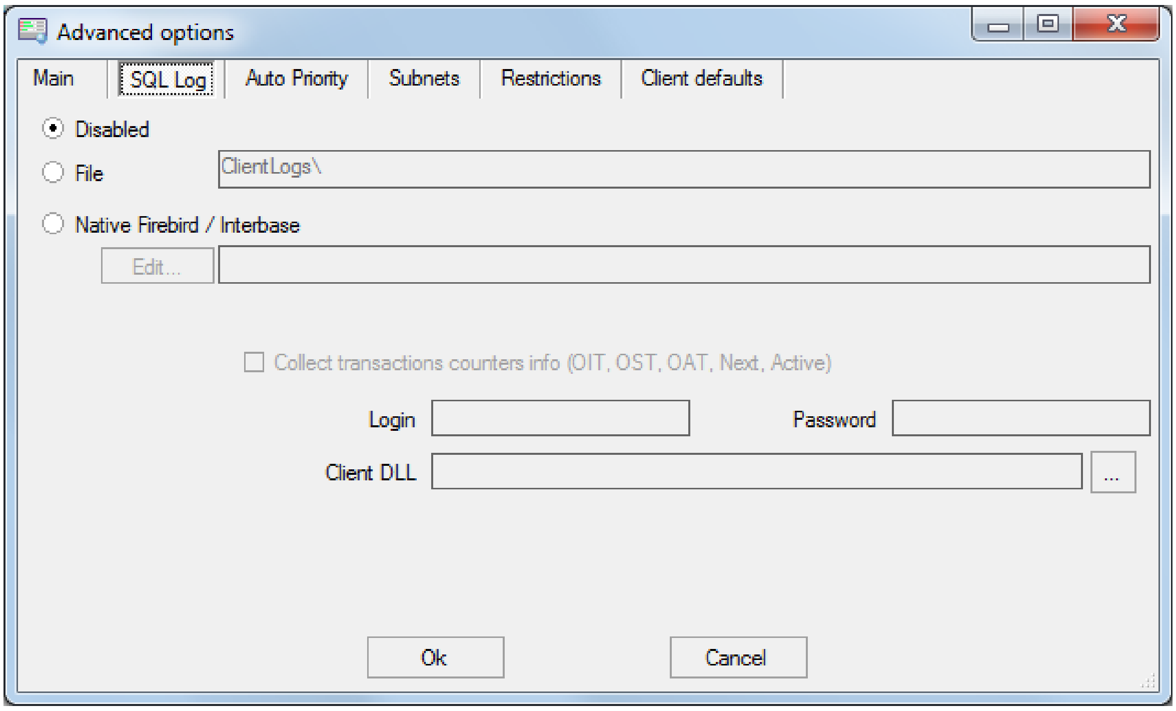
Important! If you need to pass SQL traffic from client applications through remote FBScanner, please change Firebird appropriate connection string. For example, if originally client applications have connected with “**myserver1:C:\Database\data.fdb**”, in order to pass SQL traffic through FBScanner in 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.

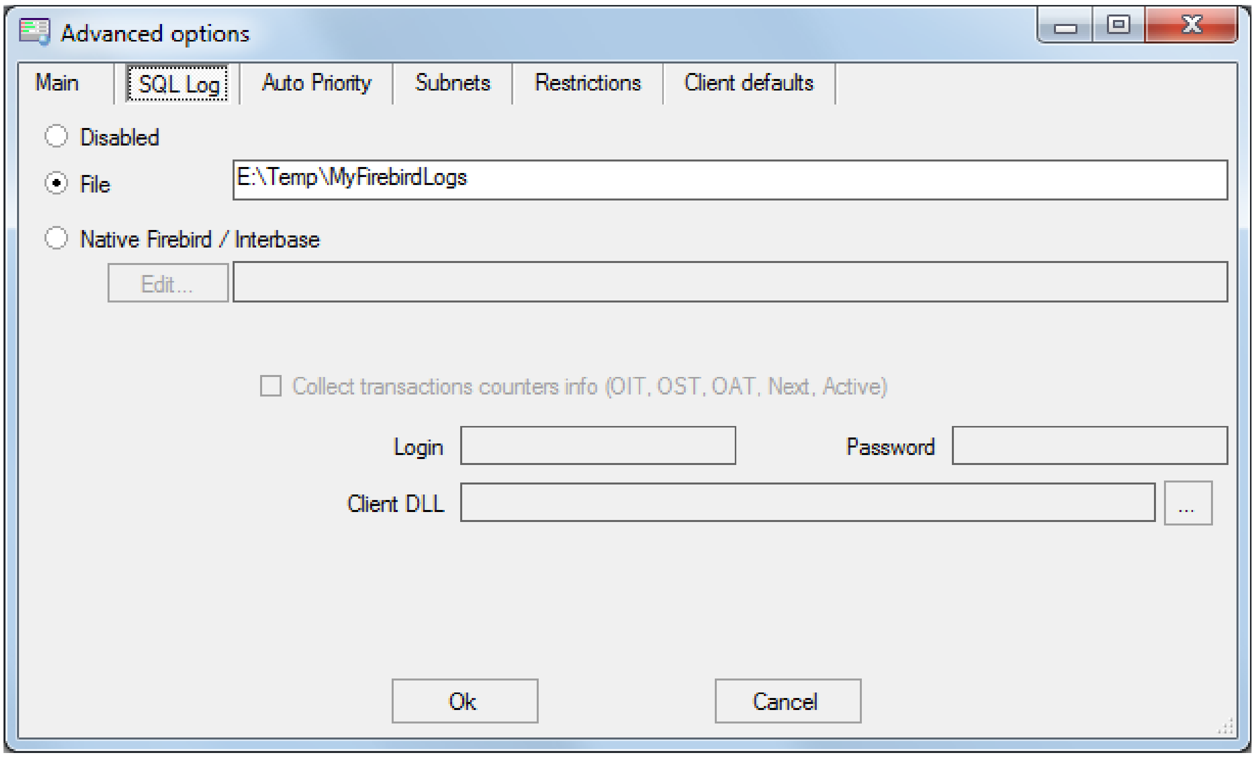
Important! It’s important to understand that logging to SQL database will write all SQL operations, including transactions, connects, etc. It means that SQL log database will consume the same amount of resources (CPU, HDD, etc) as the main database does. Due to this fact for heavy load environments we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



( ) Disabled
(●) File E:\Temp\MyFirebirdLogs
( ) Native Firebird / Interbase
☐ Collect transactions counters info (OIT, OST, OAT, Next, Active)
Login
Password
Client DLL

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.

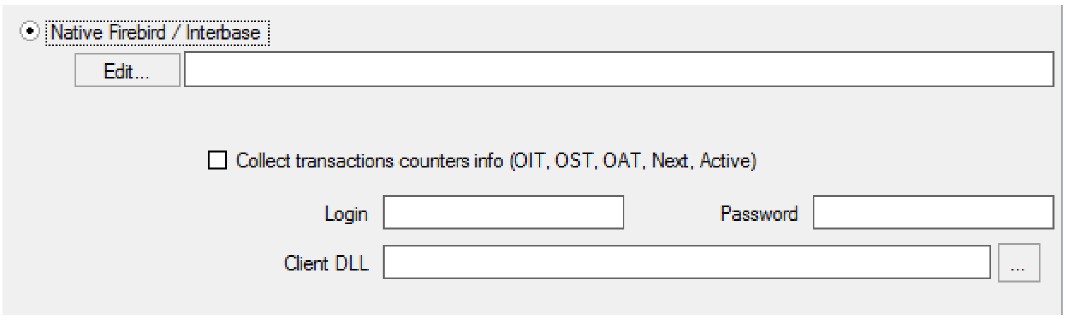
In general logging to Firebird database is implemented in the straightforward way: FBScanner service writes all traffic to the external Firebird database. Firebird database with log can be at the same computer where FBScanner resides, or at the remote computer.

Log database (Firebird 2.5)

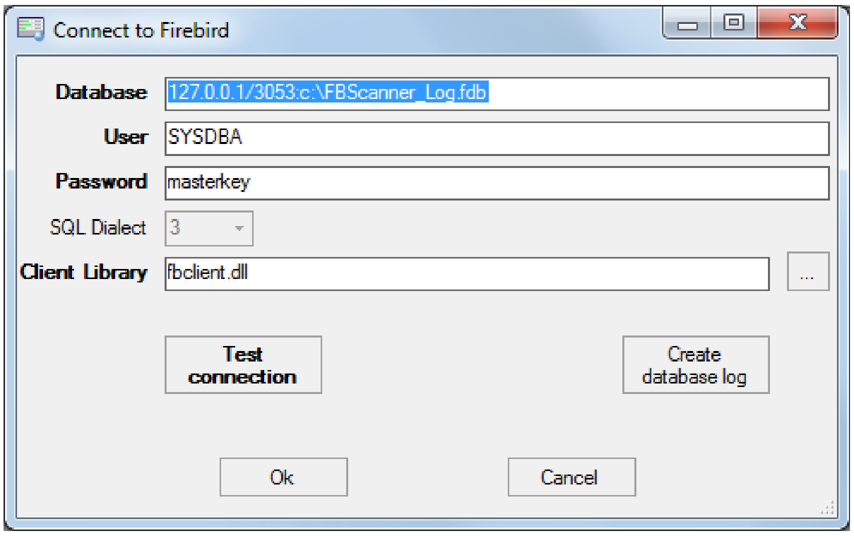
Please consider the following requirements for SQL log configuration:

- 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.


Native Firebird / Interbase
Edit...
Collect transactions counters info (OIT, OST, OAT, Next, Active)
Login
Password
Client DLL
...

First of all, click button "Edit".



Database: 127.0.0.1/3053:c:\FBScanner_Log.fdb
User: SYSDBA
Password: masterkey
SQL Dialect: 3
Client Library: fbclient.dll

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.

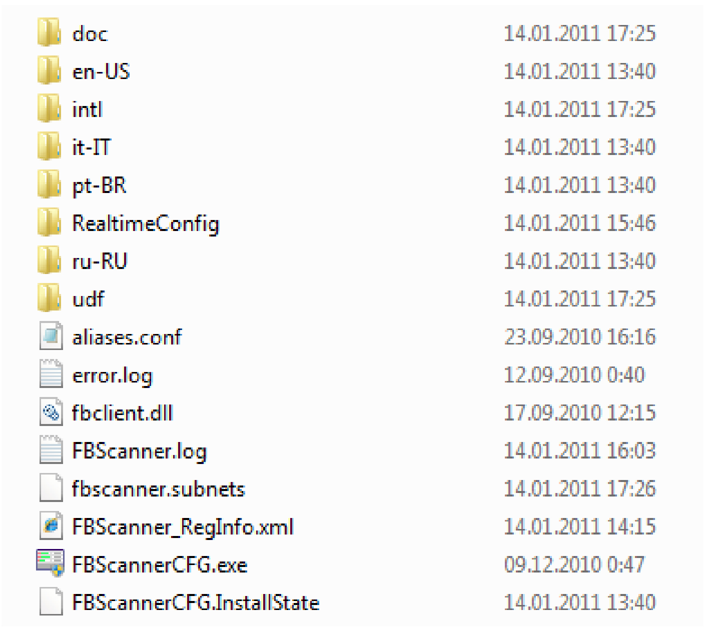
Using Embedded Firebird 2.5 for SQL log

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.

You can download Firebird 2.5 Embedded from www.firebirdsql.org.

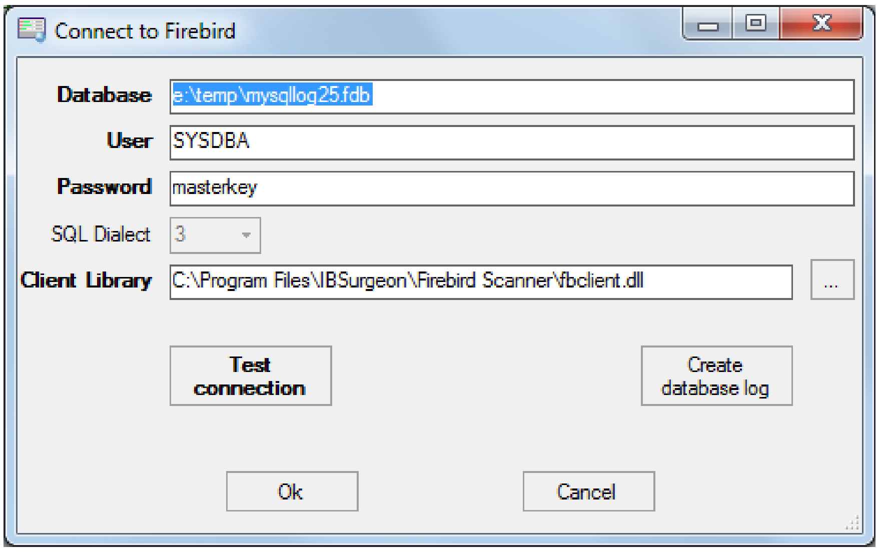
Unpack the archive right into the FBScanner folder (C:\Program Files\IBSurgeon\Firebird Scanner by default) and rename fbembed.dll into fbclient.dll.

Folder structure will look like this



doc	14.01.2011 17:25
en-US	14.01.2011 13:40
intl	14.01.2011 17:25
it-IT	14.01.2011 13:40
pt-BR	14.01.2011 13:40
RealtimeConfig	14.01.2011 15:46
ru-RU	14.01.2011 13:40
udf	14.01.2011 17:25
aliases.conf	23.09.2010 16:16
error.log	12.09.2010 0:40
fbclient.dll	17.09.2010 12:15
FBScanner.log	14.01.2011 16:03
fbscanner.subnets	14.01.2011 17:26
FBScanner_RegInfo.xml	14.01.2011 14:15
FBScannerCFG.exe	09.12.2010 0:47
FBScannerCFG.InstallState	14.01.2011 13:40

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.



Connect to Firebird

Database: e:\temp\mysqllog25.fdb
User: SYSDBA
Password: masterkey
SQL Dialect: 3
Client Library: C:\Program Files\IBSurgeon\Firebird Scanner\fbclient.dll

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?

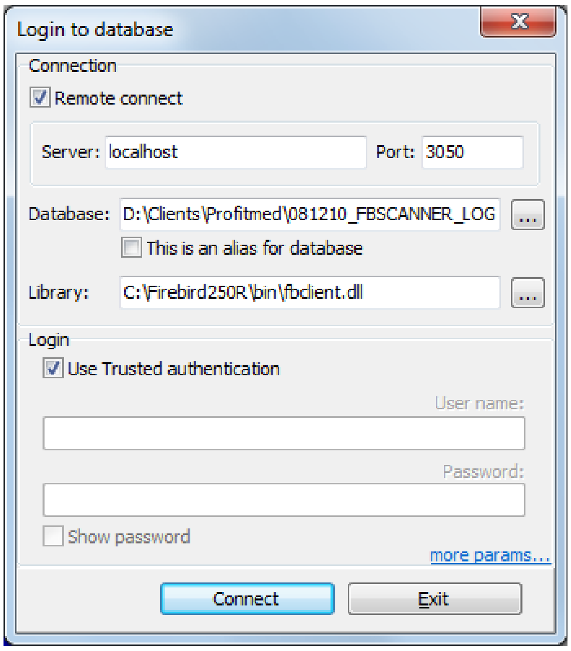
Many users told us that they did not realize how many queries, transactions and other operations are performed by their software. As you remember, FBScanner stores all information into the single table. It uses self-links to reduce the amount of stored information and it makes raw log hard to read and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

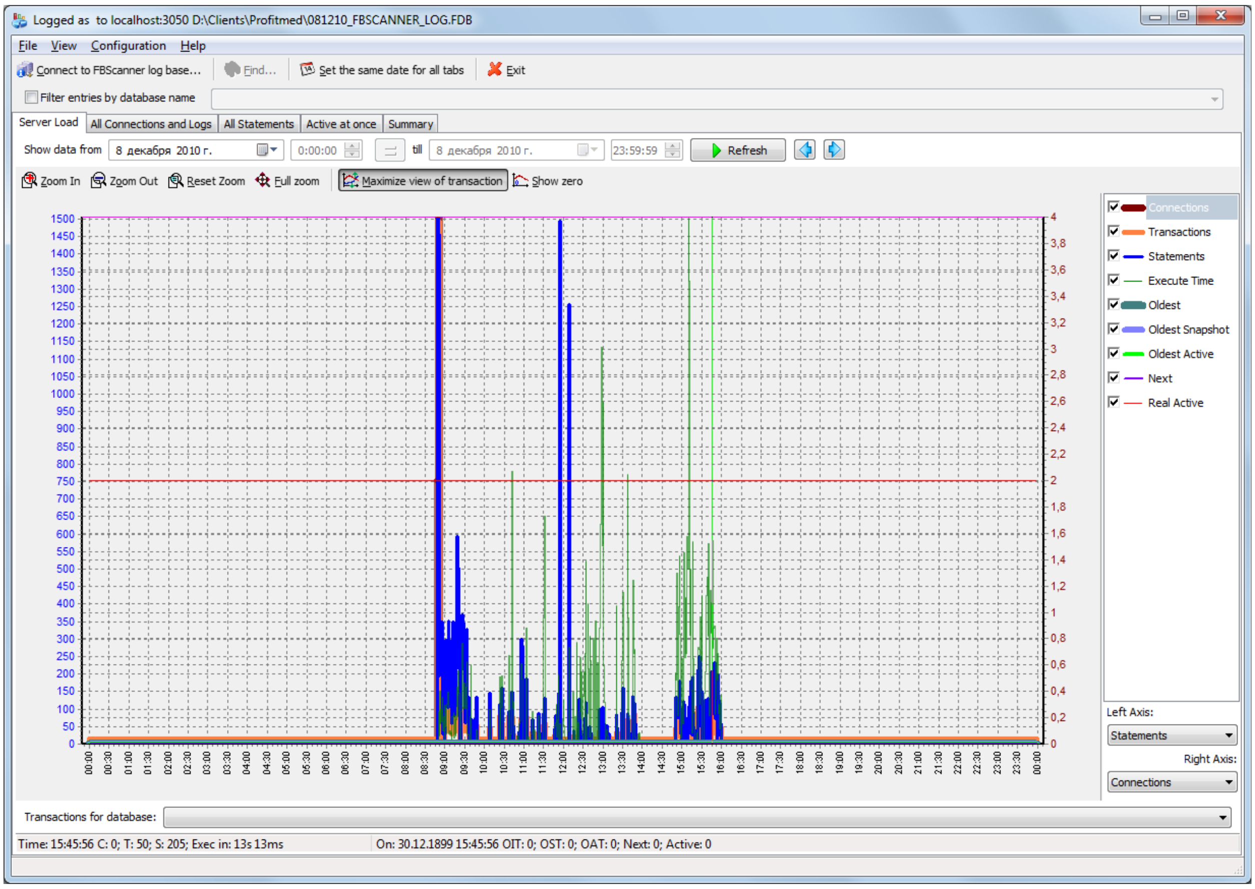
1) Setup logging and gather statistics for at least 1 day
2) Copy log database to another computer with Firebird 2.5
3) Connect to the copy of log database and perform analysis at the developer's computers
4) Copy updated versions of log databases as necessary

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


Login to database
Connection
[✓] Remote connect
Server: localhost Port: 3050
Database: D:\Clients\Profitmed\081210_FBSCANNER_LOG
[ ] This is an alias for database
Library: C:\Firebird250R\bin\fbclient.dll
Login
[✓] Use Trusted authentication
User name:
Password:
[ ] Show password

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:



Logged as to localhost:3050 D:\Clients\Profitmed\081210_FBSCANNER_LOG.FDB

Show data from 8 декабря 2010 г. 0:00:00 till 8 декабря 2010 г. 23:59:59

Left Axis: Statements

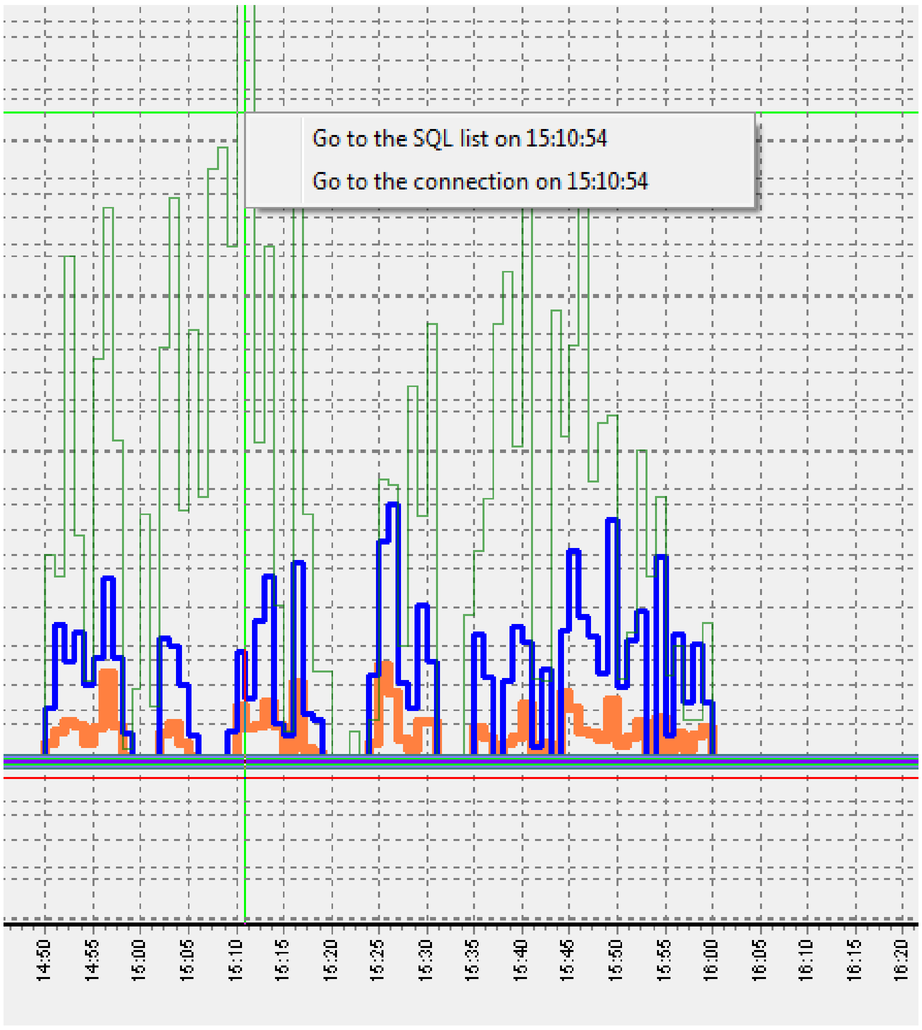
Right Axis: Connections

Time: 15:45:56 C: 0; T: 50; S: 205; Exec in: 13s 13ms

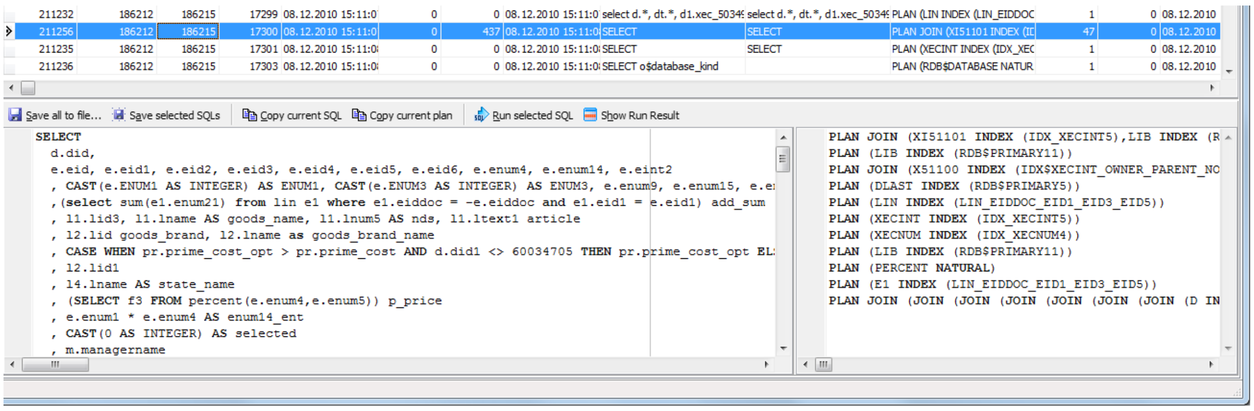
On: 30.12.1899 15:45:56 OIT: 0; OST: 0; OAT: 0; Next: 0; Active: 0

“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



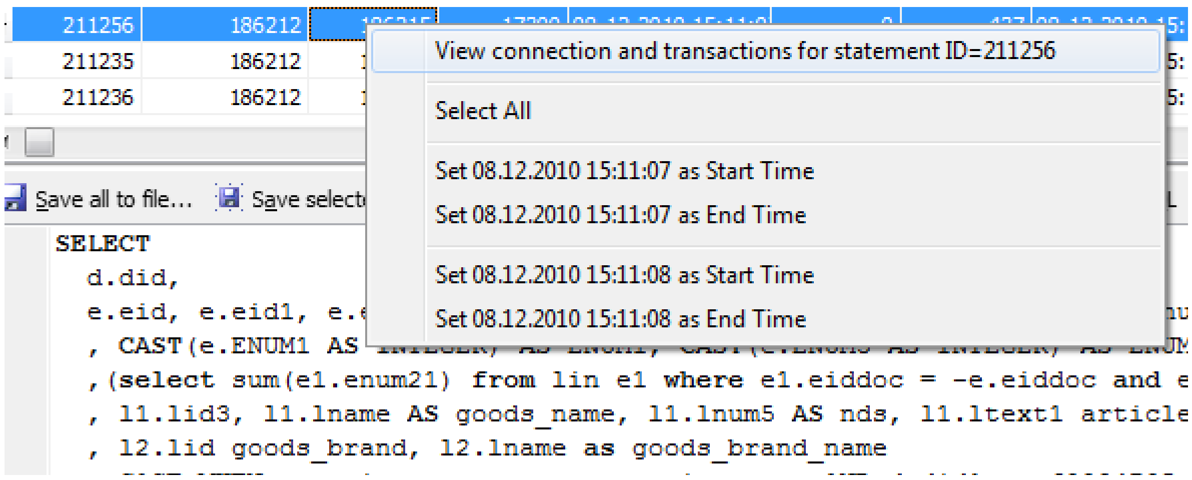
	211232	186212	186215	17299	08.12.2010 15:11:0	0	0	08.12.2010 15:11:0	select d.*, dt.*, d1.xec_5034	select d.*, dt.*, d1.xec_5034	PLAN (LIN INDEX (LIN_EIDDOC	1	0	08.12.2010
›	211256	186212	186215	17300	08.12.2010 15:11:0	0	437	08.12.2010 15:11:0	SELECT	SELECT	PLAN JOIN (XI51101 INDEX (ID	47	0	08.12.2010
	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 o$database_kind		PLAN (RDB$DATABASE 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.ltext1 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 (XI51101 INDEX (IDX_XECINT5),LIB INDEX (R
PLAN (LIB INDEX (RDB$PRIMARY11))
PLAN JOIN (X51100 INDEX (IDX$XECINT_OWNER_PARENT_NO
PLAN (DLAST 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 (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

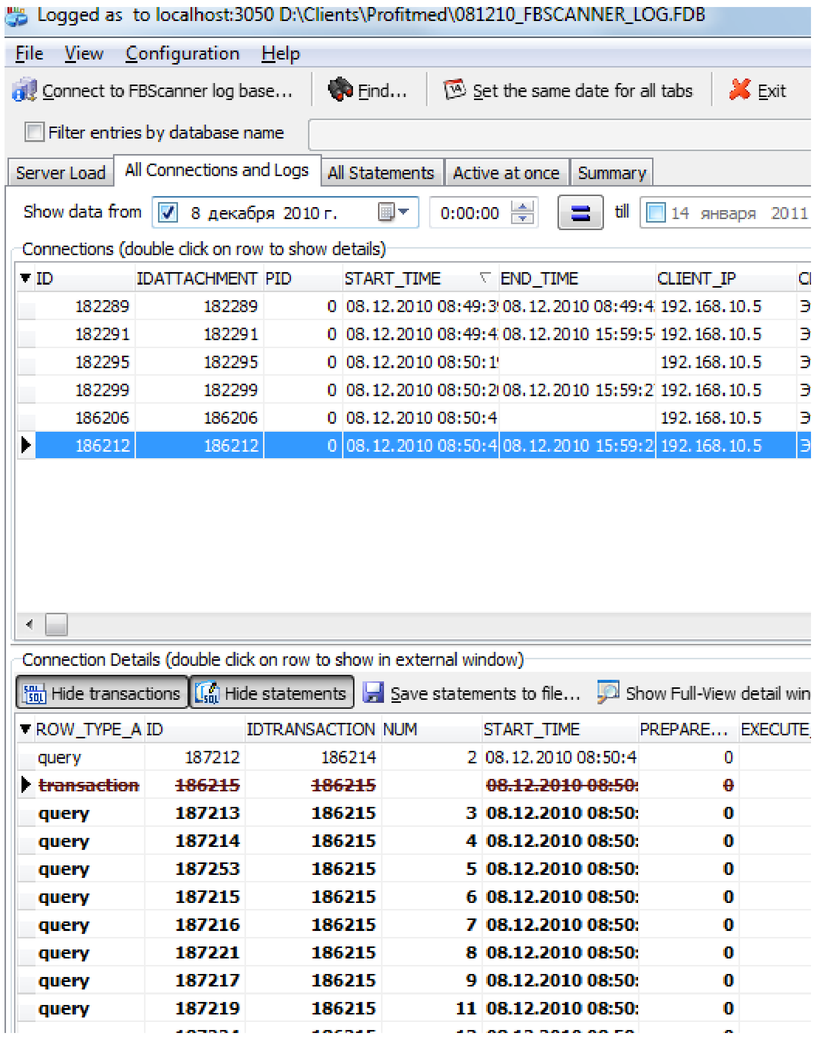


211256	186212	
211235	186212	
211236	186212	

View connection and transactions for statement ID=211256
Select All
Set 08.12.2010 15:11:07 as Start Time
Set 08.12.2010 15:11:07 as End Time
Set 08.12.2010 15:11:08 as Start Time
Set 08.12.2010 15:11:08 as End Time
Save all to file...
Save select

```
SELECT
  d.did,
  e.eid, e.eid1, e.
  , CAST(e.ENUM1 AS
  ,(select sum(e1.enum21) from lin e1 where e1.eiddoc = -e.eiddoc and e
  , l1.lid3, l1.lname AS goods_name, l1.lnum5 AS nds, l1.ltext1 article
  , l2.lid goods_brand, l2.lname as goods_brand_name
```

LogAnalyzer marks bold queries in the same transaction:

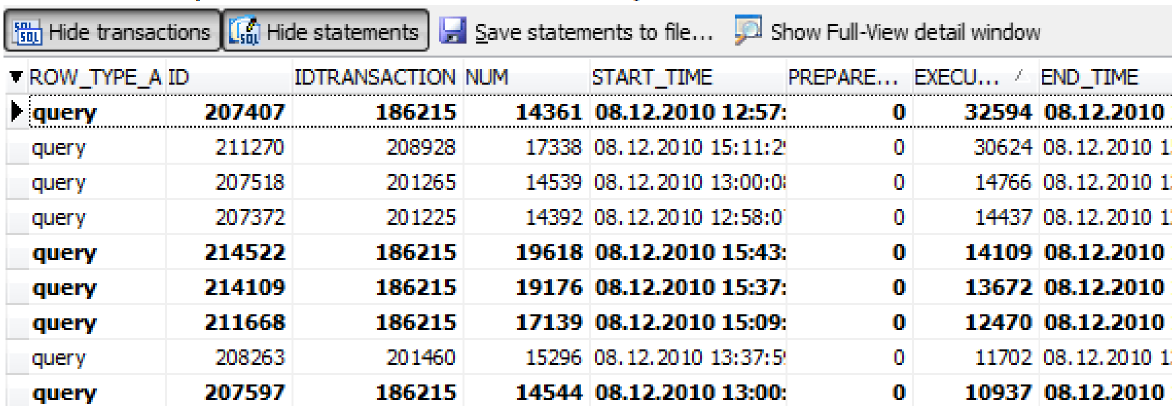


Logged as to localhost:3050 D:\Clients\Profitmed\081210_FBSCANNER_LOG.FDB
File View Configuration Help
Connect to FBScanner log base... Find... Set the same date for all tabs Exit
Filter entries by database name
Server Load All Connections and Logs All Statements Active at once Summary
Show data from 8 декабря 2010 г. 0:00:00 till 14 января 2011
Connections (double click on row to show details)
Connection Details (double click on row to show in external window)
Hide transactions Hide statements Save statements to file... Show Full-View detail win

ID	IDATTACHMENT	PID	START_TIME	END_TIME	CLIENT_IP
182289	182289	0	08.12.2010 08:49:3	08.12.2010 08:49:4	192.168.10.5
182291	182291	0	08.12.2010 08:49:4	08.12.2010 15:59:5	192.168.10.5
182295	182295	0	08.12.2010 08:50:1		192.168.10.5
182299	182299	0	08.12.2010 08:50:2	08.12.2010 15:59:2	192.168.10.5
186206	186206	0	08.12.2010 08:50:4		192.168.10.5
186212	186212	0	08.12.2010 08:50:4	08.12.2010 15:59:2	192.168.10.5

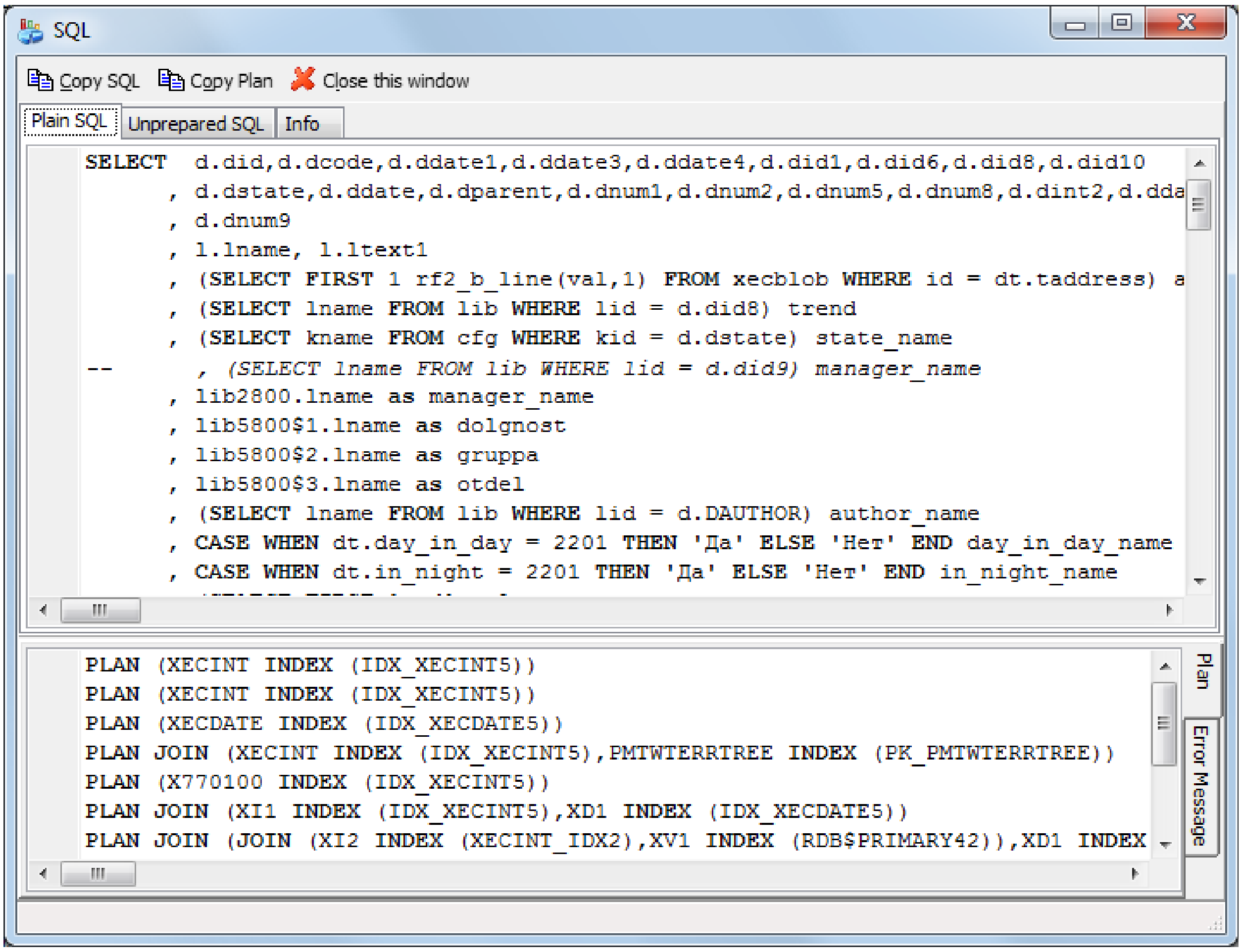
ROW_TYPE_A	ID	IDTRANSACTION	NUM	START_TIME	PREPARE...	EXECUTE
query	187212	186214	2	08.12.2010 08:50:4	0	
~~transaction~~	~~186215~~	~~186215~~		~~08.12.2010 08:50:~~	~~0~~	
query	**187213**	**186215**	**3**	**08.12.2010 08:50:**	**0**	
query	**187214**	**186215**	**4**	**08.12.2010 08:50:**	**0**	
query	**187253**	**186215**	**5**	**08.12.2010 08:50:**	**0**	
query	**187215**	**186215**	**6**	**08.12.2010 08:50:**	**0**	
query	**187216**	**186215**	**7**	**08.12.2010 08:50:**	**0**	
query	**187221**	**186215**	**8**	**08.12.2010 08:50:**	**0**	
query	**187217**	**186215**	**9**	**08.12.2010 08:50:**	**0**	
query	**187219**	**186215**	**11**	**08.12.2010 08:50:**	**0**	

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



```
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.DAUTHOR) 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
```

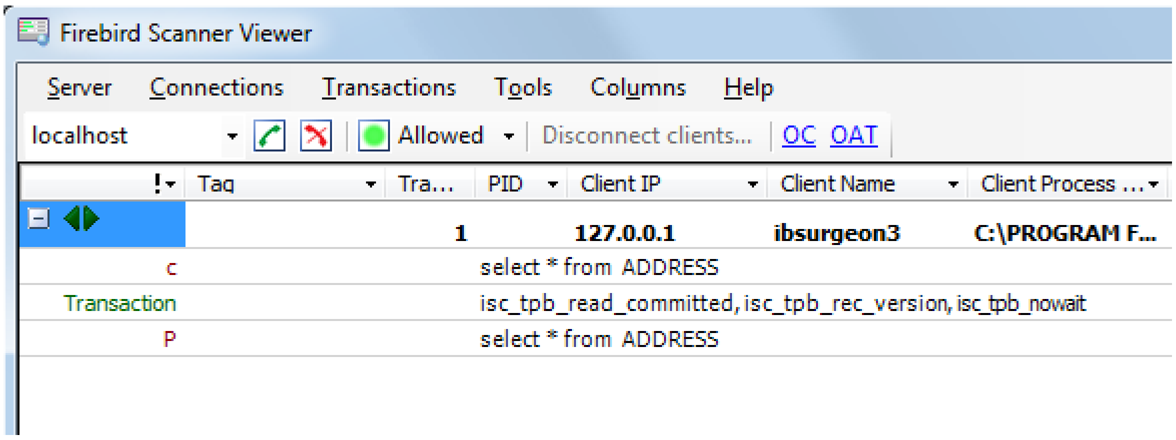
and, of course, using local connection string will always bypass FBScanner:

```
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.



!	Tag	Tra...	PID	Client IP	Client Name	Client Process ...
			1	127.0.0.1	ibsurgeon3	C:\PROGRAM F...
c			select * from ADDRESS			
Transaction			isc_tpb_read_committed, isc_tpb_rec_version, isc_tpb_nowait			
P			select * from ADDRESS			

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 RDB$DATABASE
	/*FBSCANNER$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 RETAINING" in the current transaction
Retaining	'NOW' – Latest Retaining
Received	Bytes, received by client
Sent	Bytes, sent by client
CPU Time Prepare Time Execute 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.
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: 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

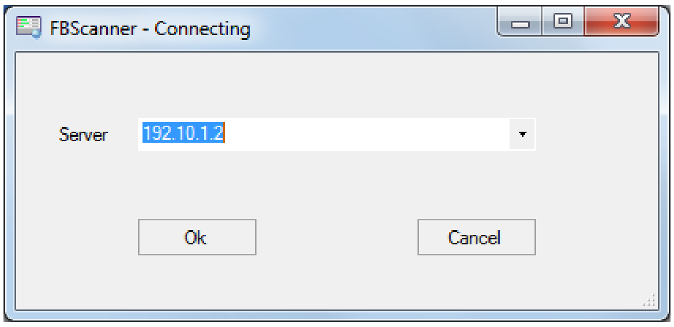
FBScanner Viewer offers wide range of options to make debugging and optimization easier, which are accessible through its menu:

- **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**

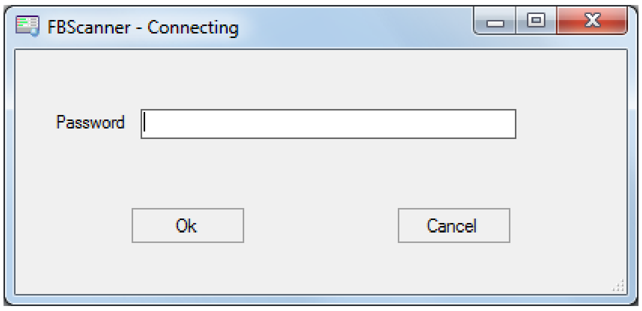
  - OAT
- **Tools**
  - **View Style**
    - Database Administrator (connections only)
    - Database Developer (without transactions)
    - Database Developer (with transactions)
  - Language – English, Italian, Russian, Portuguese
  - Plugins
  - Options
- **Columns** – list of columns
- **Help**

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.

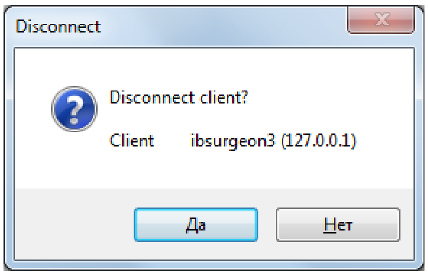
Server\Recent Servers shows list of most recent FBScanner Services where FBScanner Viewer connected to.

Exit closes FBScanner Viewer.

Connections

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

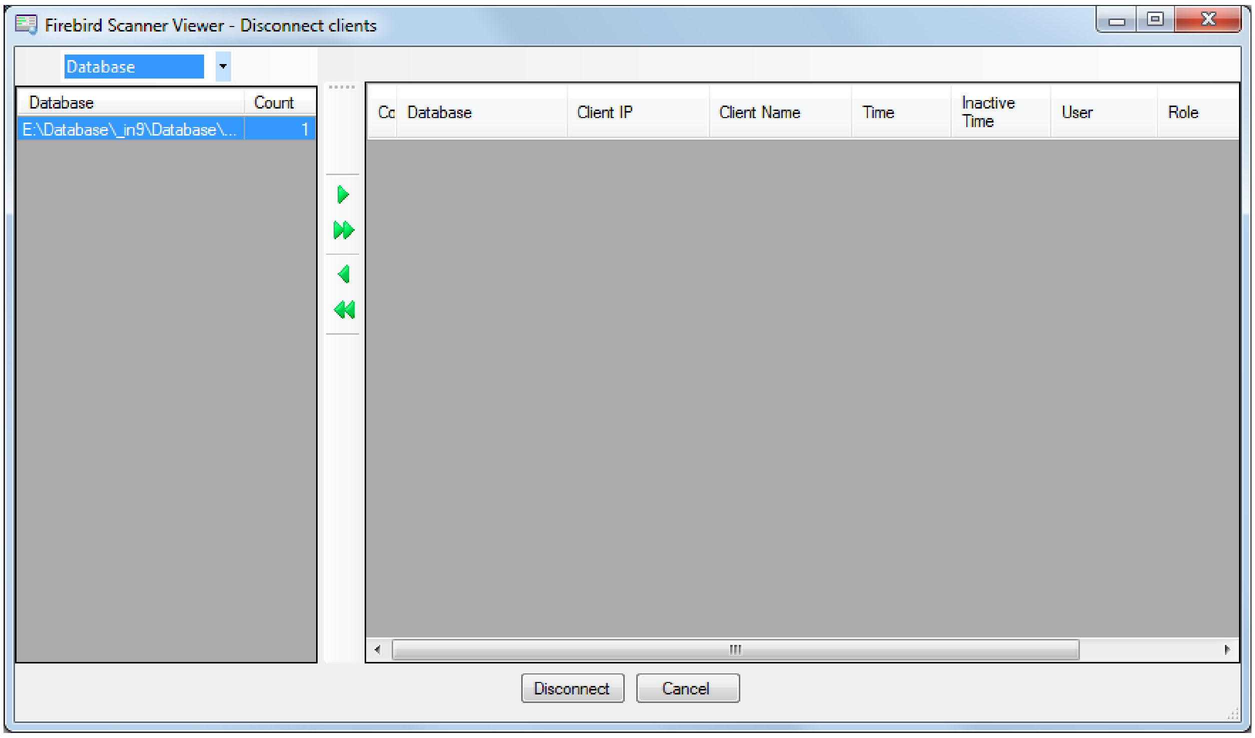
Disconnect will ask to close the current connection (highlighted in the main FBScanner Viewer grid):


Disconnect

Disconnect client?

Client ibsurgeon3 (127.0.0.1)

"**Disconnect clients**" runs the following dialog:


Firebird Scanner Viewer - Disconnect clients

Database

Database	Count
E:\Database_in9\Database\...	1

Cc	Database	Client IP	Client Name	Time	Inactive Time	User	Role

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

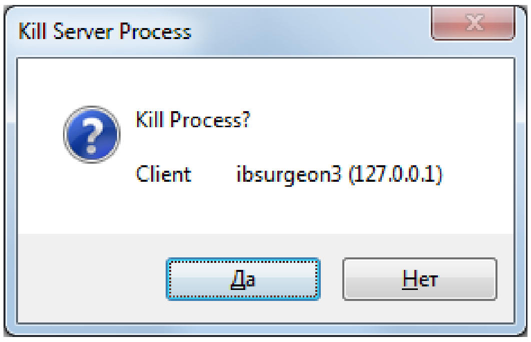
Using > and < buttons, administrator can select connections to be disconnected and then click "Disconnect" button.

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

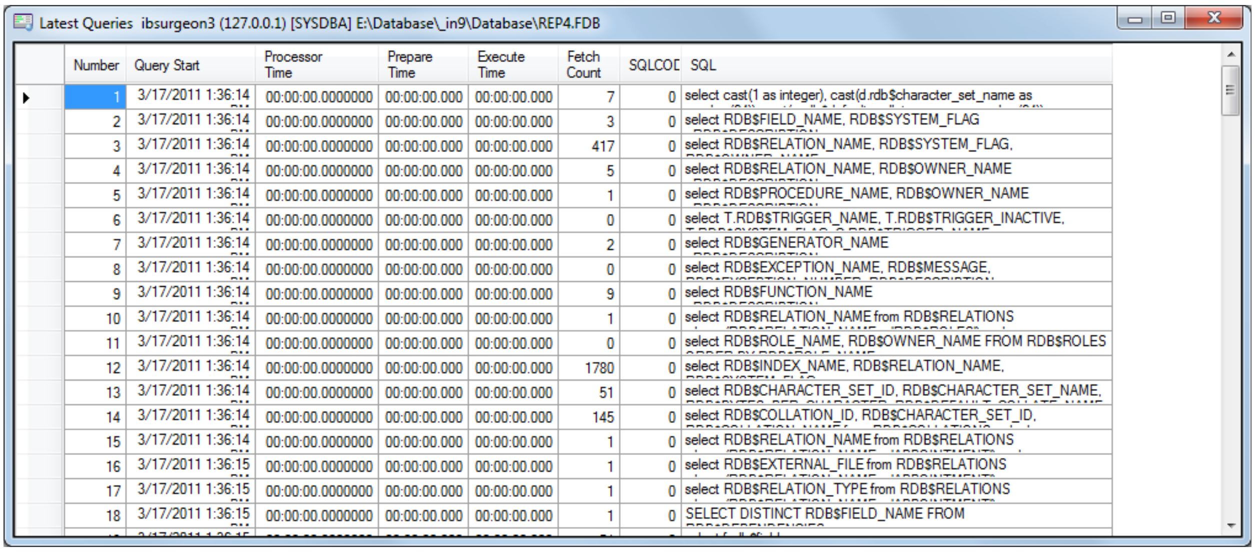
There are few cases when you need to kill Firebird process, and we do not recommend it.

"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:


Latest Queries ibsurgeon3 (127.0.0.1) [SYSDBA] E:\Database_in9\Database\REP4.FDB

Number	Query Start	Processor Time	Prepare Time	Execute Time	Fetch Count	SQLCOD	SQL
1	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	7	0	select cast(1 as integer), cast(d.rdb$character_set_name as
2	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	3	0	select RDB$FIELD_NAME, RDB$SYSTEM_FLAG
3	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	417	0	select RDB$RELATION_NAME, RDB$SYSTEM_FLAG,
4	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	5	0	select RDB$RELATION_NAME, RDB$OWNER_NAME
5	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	1	0	select RDB$PROCEDURE_NAME, RDB$OWNER_NAME
6	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	0	0	select T.RDB$TRIGGER_NAME, T.RDB$TRIGGER_INACTIVE,
7	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	2	0	select RDB$GENERATOR_NAME
8	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	0	0	select RDB$EXCEPTION_NAME, RDB$MESSAGE,
9	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	9	0	select RDB$FUNCTION_NAME
10	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	1	0	select RDB$RELATION_NAME from RDB$RELATIONS
11	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	0	0	select RDB$ROLE_NAME, RDB$OWNER_NAME FROM RDB$ROLES
12	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	1780	0	select RDB$INDEX_NAME, RDB$RELATION_NAME,
13	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	51	0	select RDB$CHARACTER_SET_ID, RDB$CHARACTER_SET_NAME,
14	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	145	0	select RDB$COLLATION_ID, RDB$CHARACTER_SET_ID,
15	3/17/2011 1:36:14	00:00:00.0000000	00:00:00.000	00:00:00.000	1	0	select RDB$RELATION_NAME from RDB$RELATIONS
16	3/17/2011 1:36:15	00:00:00.0000000	00:00:00.000	00:00:00.000	1	0	select RDB$EXTERNAL_FILE from RDB$RELATIONS
17	3/17/2011 1:36:15	00:00:00.0000000	00:00:00.000	00:00:00.000	1	0	select RDB$RELATION_TYPE from RDB$RELATIONS
18	3/17/2011 1:36:15	00:00:00.0000000	00:00:00.000	00:00:00.000	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

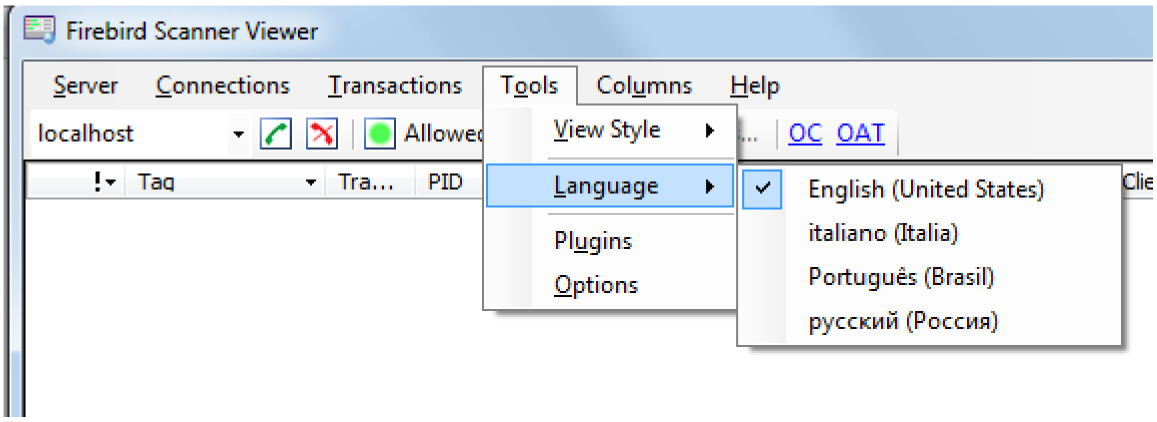
The single option **Transactions\OAT** will put selection in the grid to the oldest active transaction.

Tools

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

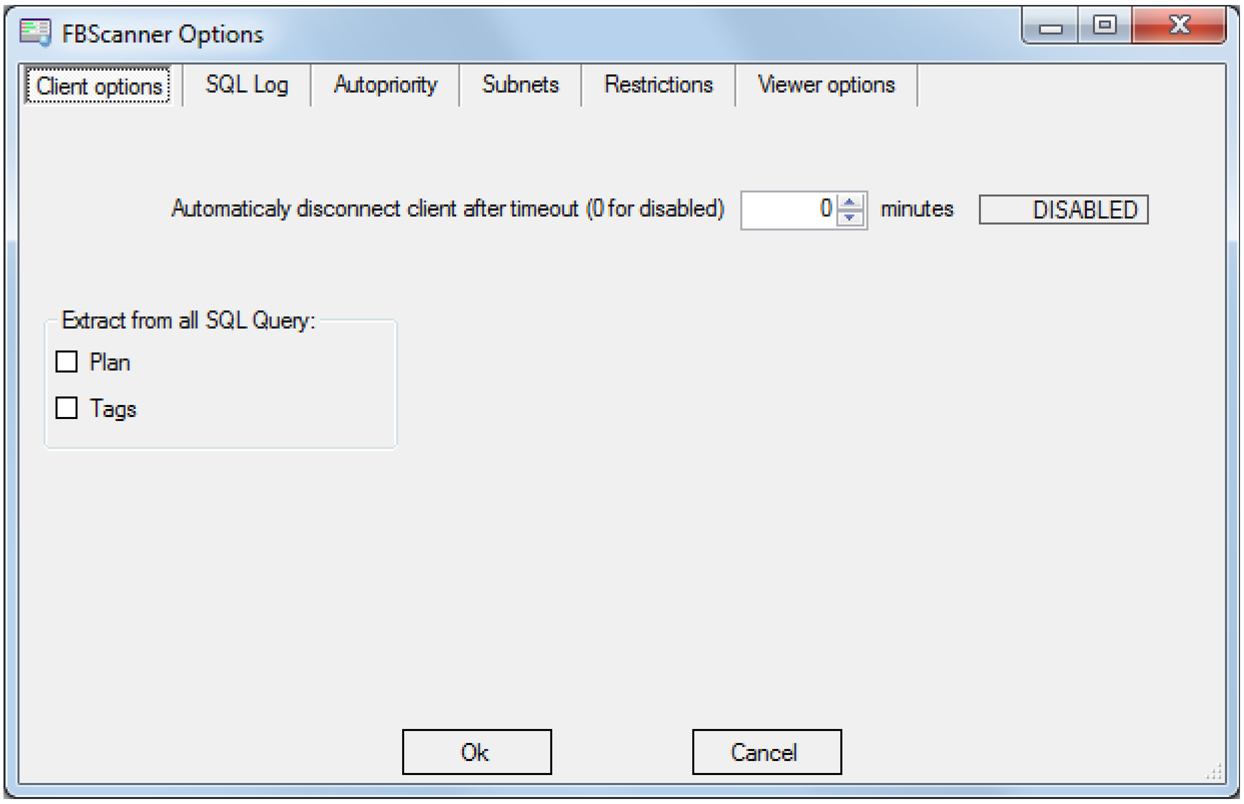
- Database Administrator (connections only)
- Database Developer (without transactions)
- Database Developer (with transactions)

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 another way to change some of FBScanner Service parameters.



Automaticaly disconnect client after timeout (0 for disabled) 0 minutes DISABLED

Extract from all SQL Query:

- [ ] Plan
- [ ] Tags

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

Level3. 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	***Information about established connections in the FBScanner Viewer:***		
	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	***Connection management (requires logging to FBScanner Viewer with Admin rights)***		
	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: 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	**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	***Logging events related with connections***	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
	If connection was broken (10054 error), FBScanner determines and logs one of the 5 type of disconnects: 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	X	X
2.	**Transactions**		
2.1.	***Transactions are shown inside appropriate connections***		
	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	X	X

	hidden)		
	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***		
	Ad-hoc plan extraction for queries 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 In both cases plans will be logged to the overall log if logging is ON.	X	X
4.	**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; */
		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:	X	X

	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")		
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

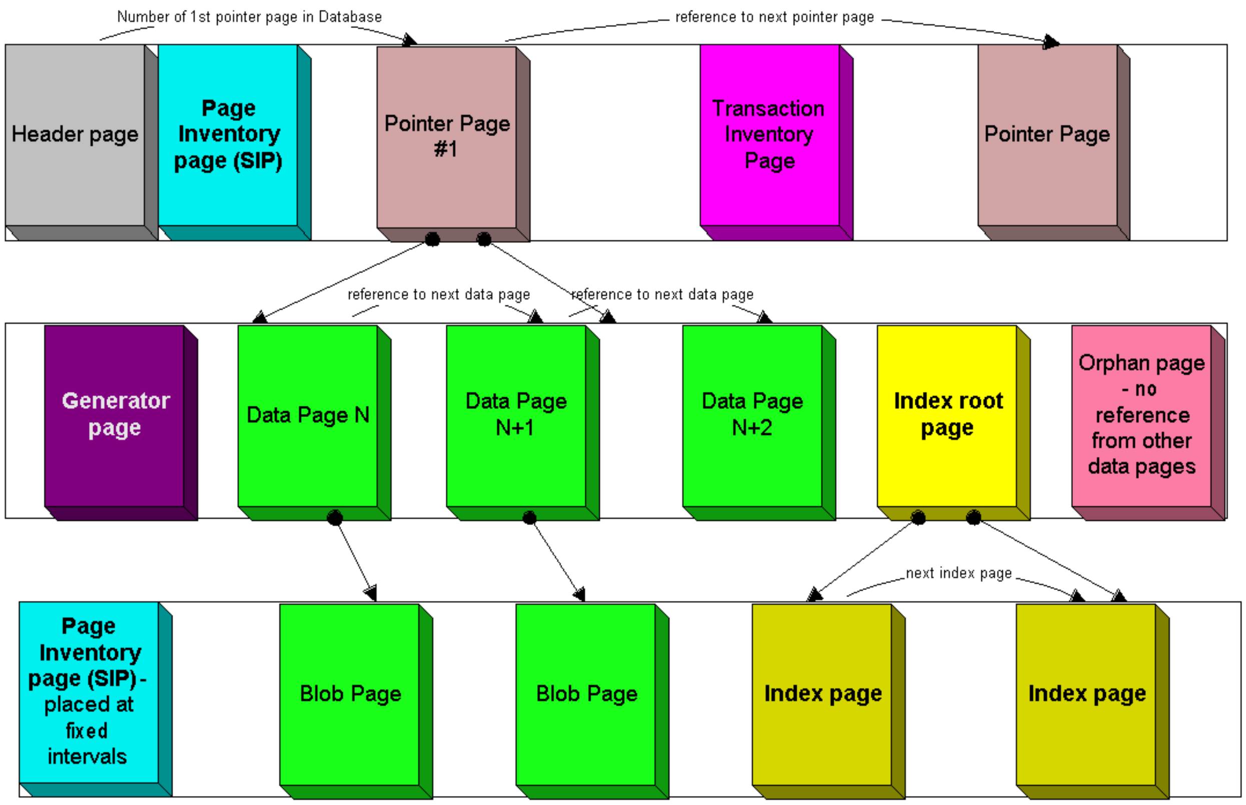
5. Database structure analysis

5.1. Overview of Firebird database structure

The first thing we have to say about the structure of Firebird database is that it represents a set of pages of strictly defined size: 4096, 8192, 16384 (previous versions of Firebird supported page sizes 1024 and 2048).

Pages can be of different types, each of which serves its certain purpose.

Pages of the same type don't go strictly one by one – they can be easily mixed, allocated in file in the order they were created by server when extending or creating databases.


Number of 1st pointer page in Database
reference to next pointer page
Header page
Page Inventory page (SIP)
Pointer Page #1
Transaction Inventory Page
Pointer Page
reference to next data page
reference to next data page
Generator page
Data Page N
Data Page N+1
Data Page N+2
Index root page
Orphan page - no reference from other data pages
next index page
Page Inventory page (SIP) - placed at fixed intervals
Blob Page
Blob Page
Index page
Index page

Page types

Page Type	ID	Description
pag_undefined	0	Undefined – If a page has this page type, it is most likely empty
pag_header	1	Database header page
pag_pages	2	Page inventory page (or Space inventory page – SIP)
pag_transactions	3	Transaction inventory page (TIP)
pag_pointer	4	Pointer page
pag_data	5	Data page
pag_root	6	Index root page
pag_index	7	Index (B-tree) page
pag_blob	8	Blob data page
pag_ids	9	Gen-ids
pag_log	10	Write ahead log information

5.2. How to analyze database structure with HQbird Database Analyst (IBAnalyst)

IBAnalyst is a tool that assists a user to analyze in detail Firebird database statistics and identify possible problems with database performance, maintenance and how an application interacts with the database. IBAnalyst graphically displays Firebird database statistics in a user-friendly way and highlights the following problems:

- tables and BLOBs fragmentation,
- record versioning,

- garbage collection,
- indices effectiveness, etc

Moreover, IBAnalyst can automatically make intelligent suggestions about improving database performance and database maintenance.

IBAnalyst can get statistics from the live production databases through Services API (recommended), or analyze text output of **gstat -a -r ...** commands. Statistics from the peak load periods can give a lot of information about actual performance problems in production databases.

Main features of IBAnalyst are listed below:

- Retrieving database statistics via Service API and from gstat output.
- Summary of all actual and possible problems in database
- Colored grid representation of tables, indices and table->indices, which highlights fragmented tables, poor indices and so on.
- Automatic expertise of database statistics provides recommendations and "how-to" for the following things:
  - Optimal database page size
  - Transactions state and gap between critical transactions
  - Different database flags
  - Index Depth
  - Index Key Duplicates
  - Fragmented Tables
  - Record Versions
  - Very Big Tables
- and more…

How to get statistics from Firebird database in right way

Right time, right place

It sounds strange, but just taking statistics via gstat or Services API is not enough. Statistics must be taken at the right moment to show how applications affect data and transactions in database. Worst time to take statistics is

Right after restore

After backup (gbak –b db.gdb) without –g switch is made

After manual sweep (gfix –sweep)

It is also correct that during work there can be moments where database is in correct state, for example, when applications make less database load than usual (users at launch, dinner or its by specific business process times).

How to catch when there is something wrong in database?

Yes, your applications can be designed so perfect that they will always work with transactions and data correctly, not making sweep gaps, lot of active transactions, long running snapshots and so on. Usually it does not happen. At least because some developers test their applications running 2-3 simultaneous users at the same time, not more. Thus, when they set up written applications for 15 and more simultaneous users, database can behave unpredictably. Of course, multi-user mode can work Ok, because most of multi-user conflicts can be tested with 2-3 concurrently running applications. But, next, when more concurrent applications will run, garbage collection problems can come (at least). And this can be caught if you take statistics at the correct moments.

If you does not experience periodical performance problems

This can happen when your applications are designed correctly, there is low database load, or your hardware is modern and very powerful (enough to handle well current user count and data).

The most valuable information is transactions load and version accumulation. This can be seen only if you setup regular statistics saving.

The best setup is to get hourly transaction statistics. This can be done by running

```
gstat –h db.gdb>db_stat_<time>.txt
```

where

- db.gdb is your database name,
- db_stat_<time>.txt is text file where statistics will be saved,
- <time> - current date and time when statistics was taken.

Also you can schedule to gather database statistics with HQbird FBDataGuard, job Database: Statistics.

If you experience periodical performance problems

These problems usually caused by automatic sweep run. First you need to determine time period between such a performance hits. Next, divide this interval minimally to 4 (8, 16 and so on). Now information systems have lot of concurrent users, and most of performance problems with not configured server and database happens 2 or 3 timers per day. For example, if performance hits happens each 3 hours, you need to take

```
gstat –h db.gdb
```

statistics each 30-45 minutes, and

```
gstat –a –r db.gdb –user SYSDBA –pass masterkey
```

each 1-1.5 hour. The best is when you take gstat –a –r statistics right before forthcoming performance hit. It will show where real garbage is and how many obsolete record versions accumulated.

What to do with this statistics

If your application explicitly uses transactions and uses them well, i.e. you know what is read read_committed and when to use it, your snapshot transactions lasts no longer than needed, and transactions are being active minimal duration of time, you can tune sweep interval or set it off, and then only care about how many updates application(s) makes and what tables need to be less updated or cared about updates.

What does this mean, you can ask? We'll give example of some system, where performance problems happened each morning for 20-30 minutes. That was very sufficient for "morning" applications, and could not last longer.

Database admin was asked correct questions, and here is the picture:

Daily work was divided by sections – analytic works in the morning, than data is inserted and edited by usual operators, and at the end of the day special procedures started gathering data, that would be used for analytic next day (at least).

The last work on database at the end of day was lot of updates, and updates of those tables which analytic used in the morning. So, there were a lot of garbage versions, which started to be collected by application, running in the morning.

And, the answer to that problem was found simple – to run gfix –sweep at the end of the day.

Sweep reads all tables in database and tries to collect all garbage versions for committed and rolled back transactions. After sweeping database became clear nearly it comes after restore.

And, "morning problem" has gone.

So, you need to understand statistics with lot of other factors:

how many concurrent users (average) work during the day

how long is the working day (8, 12, 16, 24 hours)

what kind of applications running at different day times, and how they affect data being used by other applications, running at the same time or next. I.e. you must understand business processes happening during the whole day and whole week.

When DBA can't do nothing

Sadly to say, these situations happen. And again, example:

Some system installed for ~15 users. Periodically performance is so bad, that DBA needs to restart server. After server restart everything works fine for some time, then performance gets bad again. Statistics showed that average daily transactions is about 75,000, and there are active transactions running from the start of day to the moment when performance getting down.

Unfortunately, applications were written with BDE and with no transactions using at all; i.e. all transaction handling was automatic and used by BDE itself. This caused some transactions to stay active for a long time, and garbage (record versions) accumulated until DBA restarted server. After restart the automatic sweep will start, and the garbage will be collected (eliminated).

All these was caused by applications, because they were tested only with 2-3 concurrent users, and when they became ~15, applications started to make very high load.

Need to say that in that configuration 70% of users were only reading data, and other 30% were inserting and updating some (!) data.

In this situation the only thing that can make performance better is to redesign transaction management in this application.

How IBAnalyst can help find problems in your Firebird database

Let's walk through the key features of IBAnalyst. When you look at your database statistics in IBAnalyst first time, things can be not clear, especially if IBAnalyst shows lot of warnings by colored red and yellow cells at Summary, Tables and Index views. Let's consider several real statistics examples.

Summary View

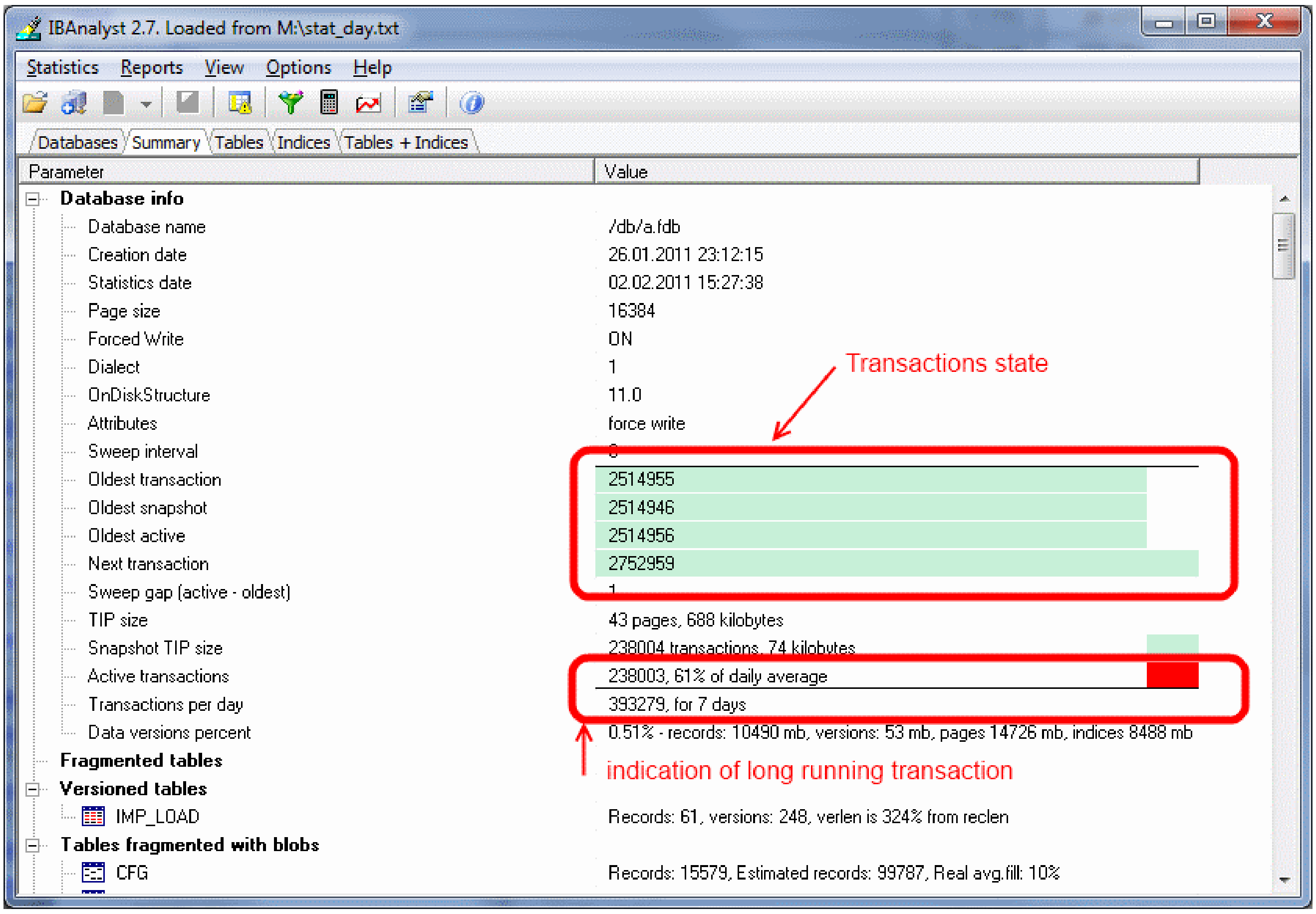
Summary contains the most important information extracted from database statistics. Usually full statistics of database contains hundreds of Kbytes and it is not easy to recognize the important information.

Below is the description of database objects and parameters that you may see in Summary. For description of visible problems (marked red or yellow) see column hints or Recommendations output.

Object or parameter	Description
Database name	Name of analyzed database.
Creation date	Database creation date. When it was created by CREATE DATABASE statement or restore (gbak -c/-r).
Statistics date	When statistics was taken – statistics file date or Services API call date (now).
Page size	Page size is the physical parameter of database. The best page size is 4096 or 8192 bytes. Other page sizes (less than 4096) marked as red. For better performance restore database from backup using 4K or 8K page size. (Note: Firebird *2.0+ can use 16K page size*).
Forced Write	It shows the mode of changed pages writing: synchronized or asynchronized – appropriate setting is ON or OFF. OFF is not recommended, because server crush, power failure or other problems can cause database corruption.
Dialect	Current database dialect.
Sweep interval	Current sweep interval value. Marked yellow if it is not 0, and marked red if

	Sweep Gap greater than Sweep interval.
On Disk Structure	ODS. It is a database physical format. See hint to know ODS number for particular IB/FB versions
Transaction block	
Oldest transaction	Oldest interesting transaction. The oldest transaction id that was rolled back, or in limbo.
Oldest snapshot	Oldest snapshot transaction Id of transaction that was oldest active when currently oldest snapshot started.
Oldest active	Oldest active transaction Id of oldest still active transaction.
Next transaction	Newest available transaction id
Sweep gap (snapshot – oldest)	For ODS 10.x databases. Difference between Oldest Snapshot and Oldest Interesting transaction. If it is greater than sweep interval, and sweep interval is > 0, Firebird tries to run sweep, and it can slowdown performance.
Snapshot gap (active – oldest)	Difference between Oldest Active and Oldest transaction. Same as previous sweep gap.
TIP size	Transaction Inventory Page size, in pages and kilobytes. TIP holds transaction state for every transaction was started from database creation (or restore). It is computed as Next transaction div 4 (bytes).
Snapshot TIP Size	Size of Transaction Inventory Pages that needed for snapshot transactions. Indicates how much memory will take each snapshot transaction to check concurrent transactions state.
Active transactions	currently active (on the moment when statistics was taken from database) transaction count (Next – Oldest Active). Maybe incorrect, because it can be one active transaction and lot of ahead transactions committed. Anyway, active transactions prevent garbage collection.
Transactions per day	Simply divides Next transaction by days count between database creation date and date statistics taken. Shows average transaction per day, and useless if it is not production database. Transaction warnings mostly based on average transactions per day count.
Data versions percent	Percent of record versions in database. Also total records size and versions size for all tables is shown, and total index size. Row is not shown when statistics does not contain record count information (gstat -a without -r option). Note that there can be lot of other data (transaction inventory pages, empty pages and so on) in your database.
Table/Index lists (also reported in recommendations)	
Fragmented Tables	Here you can view tables (with data > 200 kilobytes) that have average fill less than 60% (File/Options/Table average fill).
Versioned Tables	List of tables that have Versions greater than Records, set in Options/Tables.
Tables fragmented with blobs	List of tables that have blob fields with data size less than database page size.
Massive deletes/updates	List of tables that had lot of data deleted/updated by one delete/update statement.

Very big tables	Tables that are close to technical InterBase limit (36 gigabytes per table). You will see warning beforehand problem can occur.
Deep Indices	Indices with depth more than 3 (Options/Index)
Bad Indices	Indices with big MaxDup and TotalDup values
Broken or incomplete indices	Indices with key count less than record count. This can happen when index is broken or when statistics is taken during index creation or re-activation.
Useless Indices	Indices with Unique column = 1. May be deleted or deactivated, because they are useless for index search or sort operations.
Tables with no records	List of tables with Records = 0. This can be by design (temporary tables), or they can be just forgotten by database developer.


IBAnalyst 2.7. Loaded from M:\stat_day.txt
Statistics Reports View Options Help
Databases Summary Tables Indices Tables + Indices
Transactions state
indication of long running transaction

Parameter	Value
Database info	
Database name	/db/a.fdb
Creation date	26.01.2011 23:12:15
Statistics date	02.02.2011 15:27:38
Page size	16384
Forced Write	ON
Dialect	1
OnDiskStructure	11.0
Attributes	force write
Sweep interval	0
Oldest transaction	2514955
Oldest snapshot	2514946
Oldest active	2514956
Next transaction	2752959
Sweep gap (active - oldest)	1
TIP size	43 pages, 688 kilobytes
Snapshot TIP size	238004 transactions, 74 kilobytes
Active transactions	238003, 61% of daily average
Transactions per day	393279, for 7 days
Data versions percent	0.51% - records: 10490 mb, versions: 53 mb, pages 14726 mb, indices 8488 mb
Fragmented tables	
Versioned tables	
IMP_LOAD	Records: 61, versions: 248, verlen is 324% from reclen
Tables fragmented with blobs	
CFG	Records: 15579, Estimated records: 99787, Real avg.fill: 10%

Summary page shows a lot of information, but the most valuable is transactions state (*please read description of possible transactions states in IBAnalyst help, it is available by clicking F1 or in menu Help*).

At this screenshot you can see that some transaction is active for a long time, "60% of daily average". IBAnalyst marks such transaction's state by red, because this transaction may prevent accumulated versions to be considered as garbage by server, and so, to be garbage collected. This is a possible reason of slowness: the more versions exist for some record, the more time it will takes to read it.

In order to find this long-running transaction you can use MON$Logger module of FBScanner, or perform direct query of MON$ tables. Then, to find out which tables were affected by long running transactions (tables with a lot of record versions) you need to go to "Tables" view of IBAnalyst.

Tables view

View **Tables** contains the information about all database tables. It represents important statistical information about each table. All table warnings are marked (see details below).

You can see the following columns (Columns **Records, RecLength, VerLen, Versions, Max Vers** are visible only if statistics was generated with **gstat -r** or with "Include record/rec versions" checkbox enabled):

Column	Description
Records	Record count. Marked pink if table fragmented by blob fields which data is less than database page size. Hint shows real table fragmentation and average records if there were no blob fields. Such fragmentation can cause bad performance for big table joins or natural scans.
RecLength	Average record length. Depends on record data, especially on char/varchar columns data. Min physical record length is 17 bytes (record header + all fields are null), max – as declared in table. Statistics show this data without record header count, in this case RecLength can be 0 (if nearly all records are deleted)
VerLen	Average record version length. If it is close to RecLength, almost all record is being updated. If VerLen is 40-80% and not greater of RecLength, then Versions are mostly updates. If VerLen greater than 80-90% of RecLength, than maybe Versions are mostly deletes, or update is made by char/varchar columns with new, greater data.. Marked **yellow** if it's size is greater than specified % (Options/Record/Version size) of average record size.
Versions	Current record version count. More versions slowdown table reads. Also lot of versions means that there is no garbage collection performed or records are not read by anyone. Marked **red** if version count is greater than Records. (Options/Record Versions).
Max Vers	Max record versions for one particular record. Marked blue when it is equal to 1 and Versions is non-zero. It means that there were massive update/delete operation. See Options, Table, Massive deletes updates option.
Data Pages	Allocated data pages
Size, Mb	DataPages * Page Size, in megabytes. I.e. this is total table size, records + versions. Graph shows percentage of that table from the whole data size.
Idx Size, Mb	Sum of all indices size for that table. Graph shows percentage of that value to total size of all indices.
Slots	Count of links to data pages. Empty links are Slots-Data Pages.

	Doesn't affect disk space or performance.
Average Fill	Average data page fill %. Can be computed as **(DataPages * Page_Size)/ Records * RecLength**. Low page fill means that table is "fragmented". Frequent updates/deletes can fragment data pages. Marked red if average fill rate is less than 60% (go to Options/Average Fill to adjust it). Marked yellow if it is an artifact of high table fragmentation when it's record is too small (11-13 bytes).
Real Fill	Because we found that Average Fill, calculated by gstat, sometimes gives wrong results (at least for tables with small blobs), we placed here calculated column, that counts average fill not by data pages, but by records+versions, including record header.
20%, 40%, 60% and 100% fill	Shows page count having corresponding fill rate. Can be turned on/off in Options dialog
Total %	How big is that table plus it's indices in %, related to other tables.

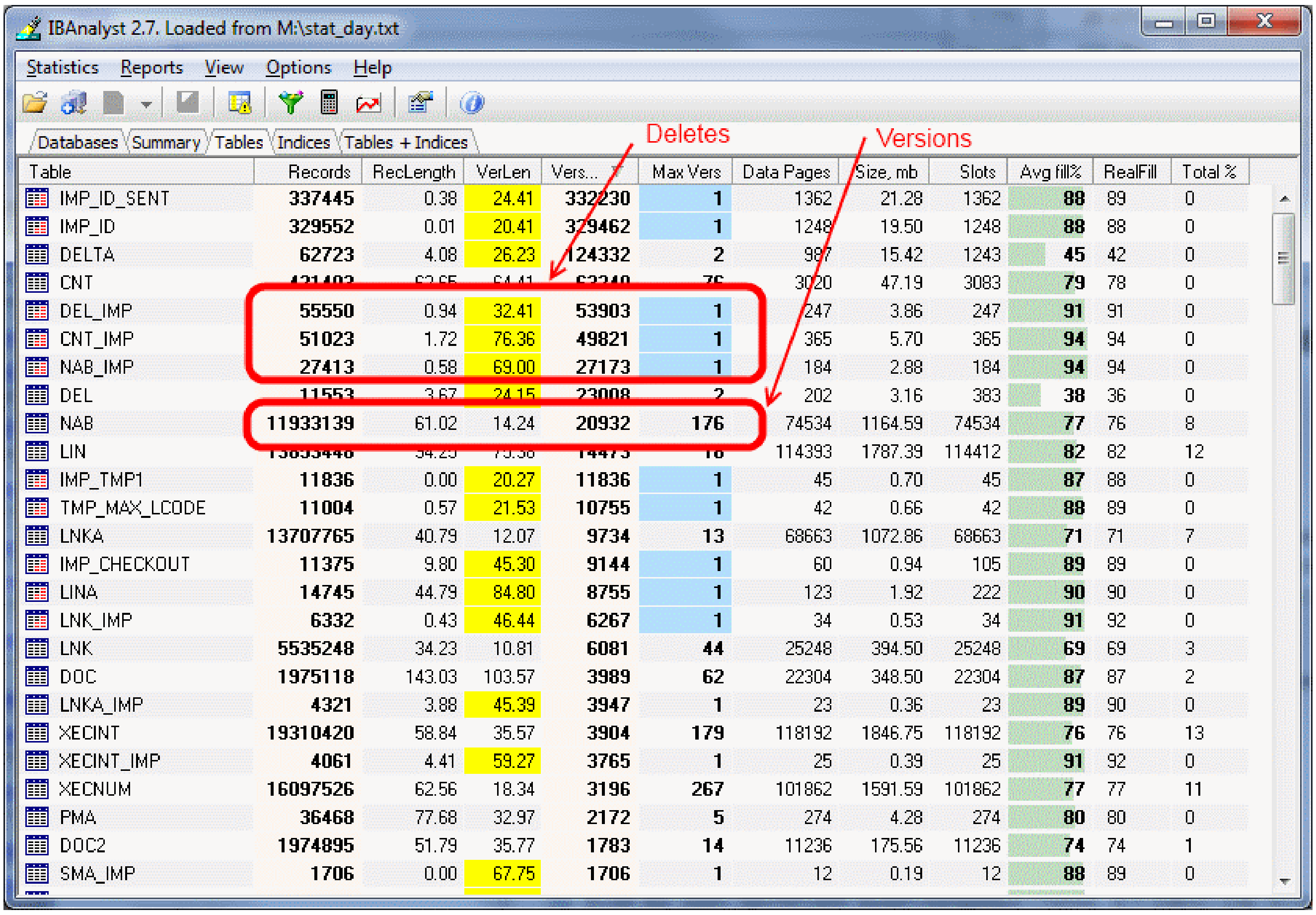

IBAnalyst 2.7. Loaded from M:\stat_day.txt
Statistics Reports View Options Help
Databases Summary Tables Indices Tables + Indices
Deletes
Versions

Table	Records	RecLength	VerLen	Vers...	Max Vers	Data Pages	Size, mb	Slots	Avg fill%	RealFill	Total %
IMP_ID_SENT	337445	0.38	24.41	332230	1	1362	21.28	1362	88	89	0
IMP_ID	329552	0.01	20.41	329462	1	1248	19.50	1248	88	88	0
DELTA	62723	4.08	26.23	124332	2	987	15.42	1243	45	42	0
CNT	421403	63.65	64.41	63240	76	3020	47.19	3083	79	78	0
DEL_IMP	55550	0.94	32.41	53903	1	247	3.86	247	91	91	0
CNT_IMP	51023	1.72	76.36	49821	1	365	5.70	365	94	94	0
NAB_IMP	27413	0.58	69.00	27173	1	184	2.88	184	94	94	0
DEL	11553	3.67	24.15	23008	2	202	3.16	383	38	36	0
NAB	11933139	61.02	14.24	20932	176	74534	1164.59	74534	77	76	8
LIN	13653448	34.23	75.38	14473	16	114393	1787.39	114412	82	82	12
IMP_TMP1	11836	0.00	20.27	11836	1	45	0.70	45	87	88	0
TMP_MAX_LCODE	11004	0.57	21.53	10755	1	42	0.66	42	88	89	0
LNKA	13707765	40.79	12.07	9734	13	68663	1072.86	68663	71	71	7
IMP_CHECKOUT	11375	9.80	45.30	9144	1	60	0.94	105	89	89	0
LINA	14745	44.79	84.80	8755	1	123	1.92	222	90	90	0
LNK_IMP	6332	0.43	46.44	6267	1	34	0.53	34	91	92	0
LNK	5535248	34.23	10.81	6081	44	25248	394.50	25248	69	69	3
DOC	1975118	143.03	103.57	3989	62	22304	348.50	22304	87	87	2
LNKA_IMP	4321	3.88	45.39	3947	1	23	0.36	23	89	90	0
XECINT	19310420	58.84	35.57	3904	179	118192	1846.75	118192	76	76	13
XECINT_IMP	4061	4.41	59.27	3765	1	25	0.39	25	91	92	0
XECNUM	16097526	62.56	18.34	3196	267	101862	1591.59	101862	77	77	11
PMA	36468	77.68	32.97	2172	5	274	4.28	274	80	80	0
DOC2	1974895	51.79	35.77	1783	14	11236	175.56	11236	74	74	1
SMA_IMP	1706	0.00	67.75	1706	1	12	0.19	12	88	89	0

At "Tables" view you can see tables and their important parameters: number of records, number of record versions, record length, maximum number of versions, etc.

You can sort this view to find the largest tables. Especially we are interested tables with many record versions – many record versions will make garbage collection for affected tables longer.

Usually it is necessary to change update and delete algorithms to get rid of many record versions.

Row Versions show total versions count for particular table, and row Max Vers shows maximum versions reached by some record. For example, if you look at table NAB, there are 11.9 million records, total versions are 20932, but one record has 176 versions. Reading and parsing such packet from disk takes more time, so, reading this record is slower than reading others.

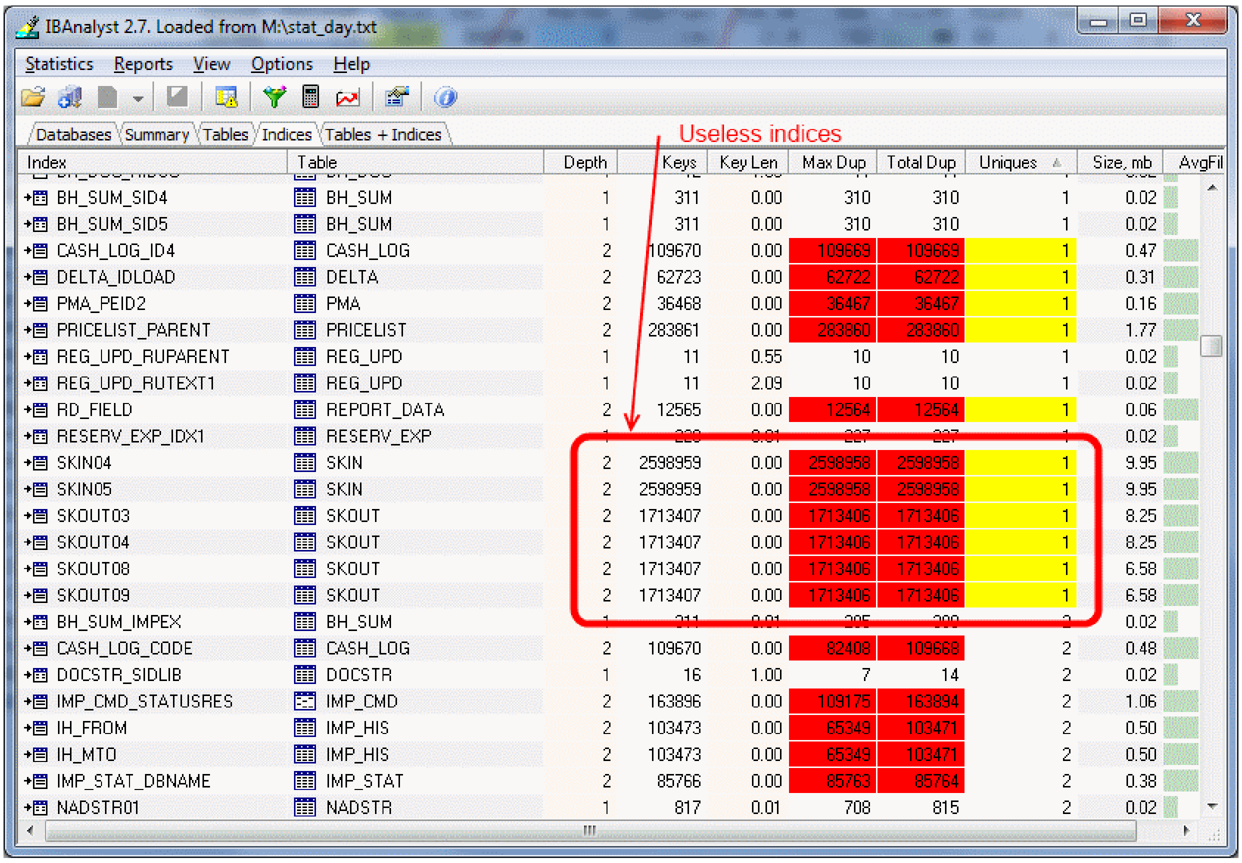
This picture also shows a lot of tables where data was deleted. But, because of long running transaction, server can't delete these versions, and they still on disk, still indexed, and still being read by server when reading data.

Index view

View **Indices** represents all indices in your database. You can estimate the effectiveness of indices with the following parameters (problem indices are marked red – see smart hints for details)

Column	Description
Depth	Index depth is the page count that engine reads from disk to walk from index root to record pointer. Optimal index depth is 3 or less. When Depth is 4 and higher, it is recommended to increase database page size (backup, then restore with -page_size option). This column will be marked red if index depth is greater than 3 (Options/Index/Index Depth). More chances to exceed optimal depth have indices built on long char/varchar columns.
Keys	Index key count. Usually equals to Records. If Keys is bigger than Records and Versions count is greater than 0 it means that concrete field value was changed in those record versions. If Table.RecVersions is bigger than Keys, than this index field(s) are not changed during updates.
KeyLen	Average index key length. The less KeyLen, the more equal or similar (postfix) values (keys) stored in index.
Max Dup	Maximum duplicates count for particular key value. Some old gstat versions show no more than 32767 or 65535 – this bug is fixed in latest Firebird versions. Marked red if duplicates count is 30% of all keys. (Options/Index/Lot of key duplicates).
Total Dup	The overall count of keys with the same values. Some old gstat versions show no more than 32767 or 65535 – this bug is fixed in latest Firebird versions. The closer this value to Keys count, the less effective will be searching using this index, especially when search is made using more than one index. Total Dup value can be counted as Keys minus unique keys count (index statistics is nonlinear). Marked yellow if 1/(Keys – TotalDup) greater than 0.01, and red if in addition MaxDup is marked red too. This constant (0.01) is used by optimizer (see sources in opt.c/opt.cpp) as usable index selectivity border. Optimizer will still use that index if none other index with better selectivity exists for some condition.

Uniques	Count of different key values. Primary and unique key indices will show same value as in Keys column. Useful to understand how many different values stored in index – is it useful or not. Index is useless if Unique column shows 1 (marked yellow).
Selectivity	Information from rdb$indises.rdb$statistics, only visible if "load table/index metadata" was On. If selectivity stored in database differs from computed selectivity, yellow warning shown (less than 20% difference) or red (higher than 20% difference). Blue warning is shown when index is empty but it's selectivity is not 0. Selectivity of inactive indices are ignored.
Size, Mb	Index size in megabytes. Gap show percentage of that index size related to total size of all indices.
Average Fill	Average index pages fill rate, in %. Marked red if average fill rate is less than 50% (go to Options/Average Index Fill to adjust it). Fragmented index results more page reads as usual, and it's Depth can be higher. Can be fixed by alter index inactive/active, if it is not index created by primary, unique or foreign key constraints.
Leafs	Leaf page count (pages with keys and record pointers).
20%, 40%, 60% and 100% fill	Shows page count having corresponding fill rate. Can be turned on/off in Options dialog


IBAnalyst 2.7. Loaded from M:\stat_day.txt
Statistics Reports View Options Help
Databases Summary Tables Indices Tables + Indices
Useless indices

Index	Table	Depth	Keys	Key Len	Max Dup	Total Dup	Uniques	Size, mb	AvgFil
BH_SUM_SID4	BH_SUM	1	311	0.00	310	310	1	0.02	🟩
BH_SUM_SID5	BH_SUM	1	311	0.00	310	310	1	0.02	🟩
CASH_LOG_ID4	CASH_LOG	2	109670	0.00	109669	109669	1	0.47	🟩
DELTA_IDLOAD	DELTA	2	62723	0.00	62722	62722	1	0.31	🟩
PMA_PEID2	PMA	2	36468	0.00	36467	36467	1	0.16	🟩
PRICELIST_PARENT	PRICELIST	2	283861	0.00	283860	283860	1	1.77	🟩
REG_UPD_RUPARENT	REG_UPD	1	11	0.55	10	10	1	0.02	🟩
REG_UPD_RUTEXT1	REG_UPD	1	11	2.09	10	10	1	0.02	🟩
RD_FIELD	REPORT_DATA	2	12565	0.00	12564	12564	1	0.06	🟩
RESERV_EXP_IDX1	RESERV_EXP	1	228	0.01	227	227	1	0.02	🟩
SKIN04	SKIN	2	2598959	0.00	2598958	2598958	1	9.95	🟩
SKIN05	SKIN	2	2598959	0.00	2598958	2598958	1	9.95	🟩
SKOUT03	SKOUT	2	1713407	0.00	1713406	1713406	1	8.25	🟩
SKOUT04	SKOUT	2	1713407	0.00	1713406	1713406	1	8.25	🟩
SKOUT08	SKOUT	2	1713407	0.00	1713406	1713406	1	6.58	🟩
SKOUT09	SKOUT	2	1713407	0.00	1713406	1713406	1	6.58	🟩
BH_SUM_IMPEX	BH_SUM	1	311	0.01	205	309	2	0.02	🟩
CASH_LOG_CODE	CASH_LOG	2	109670	0.00	82408	109668	2	0.48	🟩
DOCSTR_SIDLIB	DOCSTR	1	16	1.00	7	14	2	0.02	🟩
IMP_CMD_STATUSRES	IMP_CMD	2	163896	0.00	109175	163894	2	1.06	🟩
IH_FROM	IMP_HIS	2	103473	0.00	65349	103471	2	0.50	🟩
IH_MTO	IMP_HIS	2	103473	0.00	65349	103471	2	0.50	🟩
IMP_STAT_DBNAME	IMP_STAT	2	85766	0.00	85763	85764	2	0.38	🟩
NADSTR01	NADSTR	1	817	0.01	708	815	2	0.02	🟩

Some production databases can have indices with the only key value indexed. This can happen because database was developed "to be extended in the future", or, someone just experimented with the indices during development or tests. You can see these indices as "Useless" in IBAnalyst:

SKIN04, SKIN05, SKOUT03, etc, built on the column that has only one value for all rows (million rows). These indices are really useless, because

- Optimizer may use this index if you specify "where field =...". Since field contains only one value, using index will cause useless reading of index pages from disk to memory, and consume memory (and time) when server will prepare which rows to show for that query.
- Creating indices is the part of restore process. Extra indices adds extra time.

Of course, that is not all that you can find about your database in IBAnalyst. You can also find

- average number of transactions per day
- was there rollbacks or lost connections, and when
- how big (in megabytes) each table and index
- tables that have records interchanged by blobs, and thus reading only records is slower
- empty tables – just forgotten, or empty at the time when statistics was taken
- indices with lot of duplicate keys (you can consider about column value distribution)
- indices with depth 4 and greater – maybe you need to increase page size to speed up

6. HQbird Enterprise: Native Firebird replication

What is HQbird Enterprise?

HQbird Enterprise is the advanced distribution of Firebird for big databases with monitoring, optimization and administration tools, it also includes the plugin for native master-slave replication.

Compatibility

HQbird Enterprise is 100% compatible with Firebird 2.5.5+ and Firebird 3.0 – no changes in ODS are needed. To switch to HQbird and back no backup/restore is required, just stop/start Firebird and replace binaries. The replication is possible between nodes with the same version, i.e., not possible between3.0 and 2.5.

How the replication works

HQbird replication works on the logical level: it replicates DML statements (INSERT/UPDATE/DELETE, stored procedures, etc) and DDL (CREATE/ALTER TABLE, etc) changes; no additional triggers needed. The only requirement for the current version of replication is to have unique or primary keys for all tables that need to be replicated.

In order to use the replication, you need to install HQbird instead of Firebird, register it (with trial or with the full license) and configure replication. HQbird should be running on the master server and on all replica servers.

Below we will consider how to setup Firebird replication with HQbird Enterprise.

You can use HQbird for Windows and Linux, Firebird 2.5 and 3.0, 32bit and 64bit.

Installation

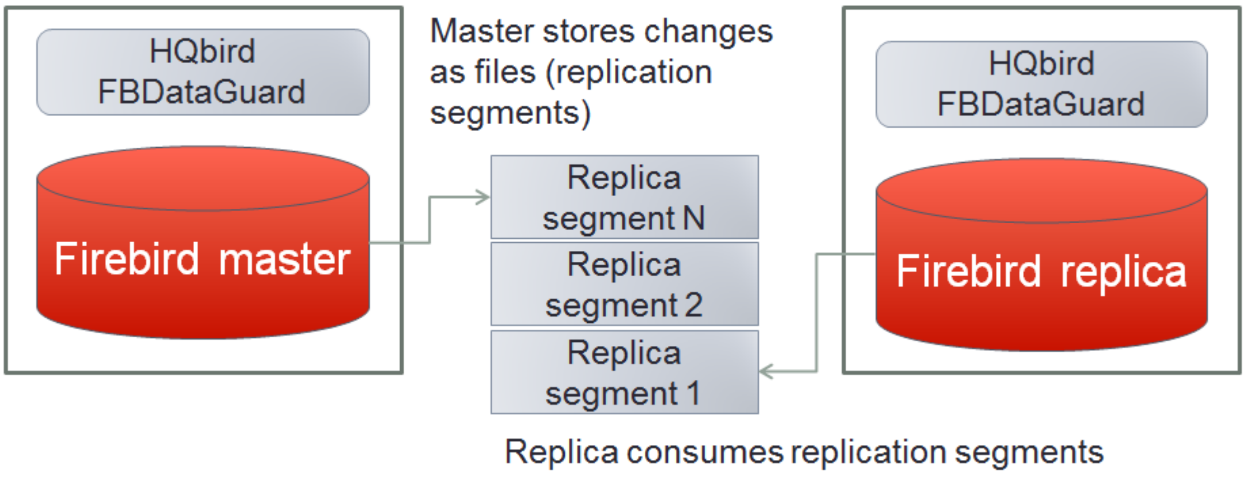
Please install HQbird Enterprise from the supplied distributive. If you have other version of Firebird installed, uninstall it first.

To enable replication you need to have working and registered (trial or full) copy of HQbird 2018 or higher on your master and replica servers.

Please refer to the section 2 (from page 8) of HQbird User Guide for details of HQbird ServerSide installation.

Asynchronous replication for Firebird

HQbird supports 2 types of replication: asynchronous and synchronous. In the case of an asynchronous replication, the master server stores committed changes from the master database to the files (replication segments), which can be consumed asynchronously by one or more replica servers.



How the asynchronous replication works:

- Changes on the master side are journaled into the replication log files
- Journal consists of multiple segments (files)
- Replication (archived) segments are transferred to the slave and applied to the replica in the background
- Replica can be created and recreated online (without master's stop)

Important things to consider:

- Practical delay between master and replica is configurable, can be set to 15-30 seconds (default is 90 seconds)
- Delay between master and replica can grow in case of heavy load (due to the delayed processing of replication segments)
- Replica can be switched to the master (i.e., normal) mode with 1 command

Asynchronous replication is the recommended choice for HQbird Enterprise:

- it provides stability and anti-corruption protection of Firebird database,
- it can be configured quickly and easily,
- it does not require downtime to setup,
- it has online re-initialization (in HQbird 2018 and higher),
- it is suitable for distributed environments (when the replica is located in the cloud or at the remote location).

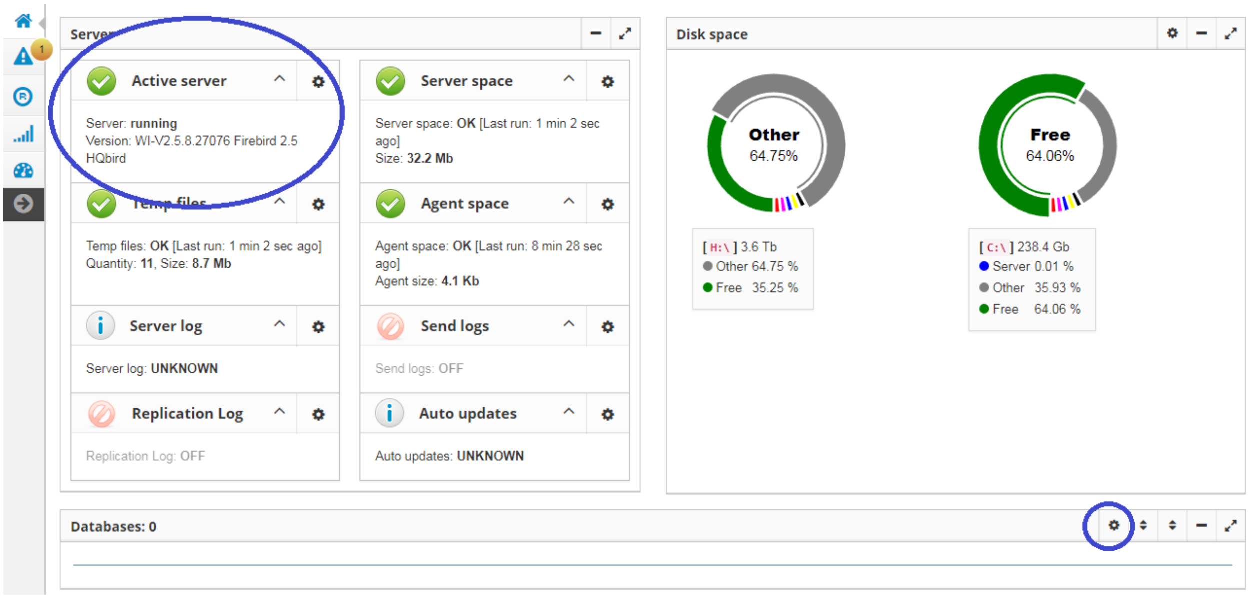
The following steps will be required to setup the asynchronous replication:

1. Configure HQbird for replication at the master

2. Create a copy of master database file
3. Setup database for replication at the replica(slave) server

Step 1: Configure HQbird for replication at the master

To setup replication, open HQbird FBDataGuard: run modern browser (Chrome, Firefox, etc) and open this local URL: http://127.0.0.1:8082(port if configurable in HQBird ini files)
Enter default name and password: **admin/strong password**. Register Firebird server, and the following picture will appear:



Server: running
Version: WI-V2.5.8.27076 Firebird 2.5 HQbird

Temp files: OK [Last run: 1 min 2 sec ago]
Quantity: 11, Size: 8.7 Mb

Server log: UNKNOWN

Replication Log: OFF

Server space: OK [Last run: 1 min 2 sec ago]
Size: 32.2 Mb

Agent space: OK [Last run: 8 min 28 sec ago]
Agent size: 4.1 Kb

Send logs: OFF

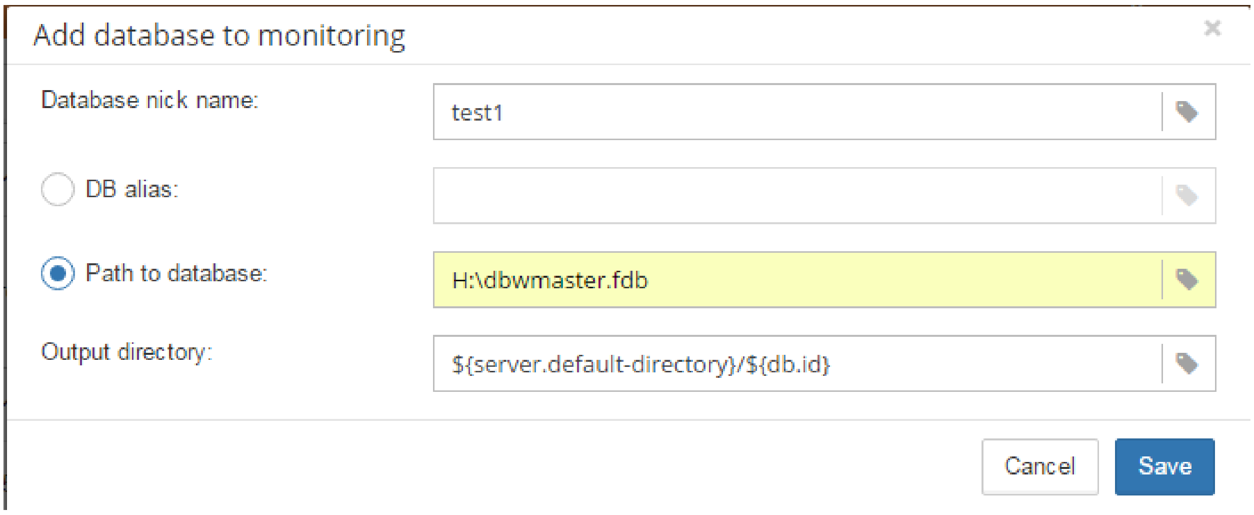
Auto updates: UNKNOWN

[H:\] 3.6 Tb
● Other 64.75 %
● Free 35.25 %

[C:\] 238.4 Gb
● Server 0.01 %
● Other 35.93 %
● Free 64.06 %

Check that you are actually connected to the correct Firebird version – in the upper left corner in «Active server» widget should be version «... Firebird 2.5 HQbird» or «... Firebird 3.0 HQbird».

After that click «Add database» in the right bottom corner and configure nick name and path to the database which will be master:

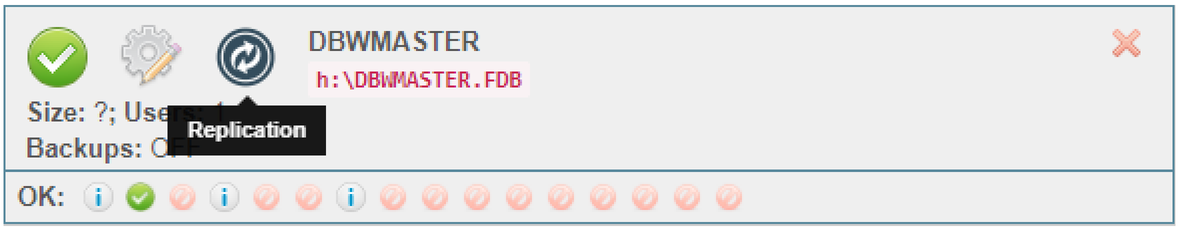


Add database to monitoring

Database nick name:	test1
DB alias:	
[✓ Path to database:]	H:\dbwmaster.fdb
Output directory:	${server.default-directory}/${db.id}

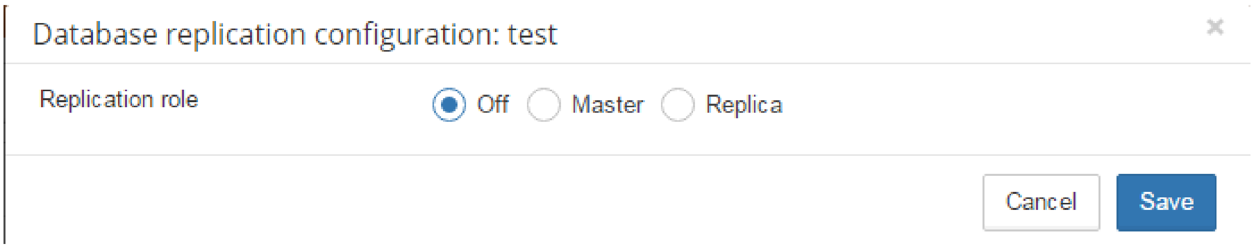
Please note that database should be registered with its explicit path, not with the alias – the replication will not work with the alias.

After the successful registration of the database click on the icon in the header of database to setup replication:



DBWMASTER
h:\DBWMASTER.FDB
Size: ?; Users: 1
Backups: Off
OK:

After that the main configuration dialog for master and replica databases will appear. When replication is not configured, this dialog is almost empty:



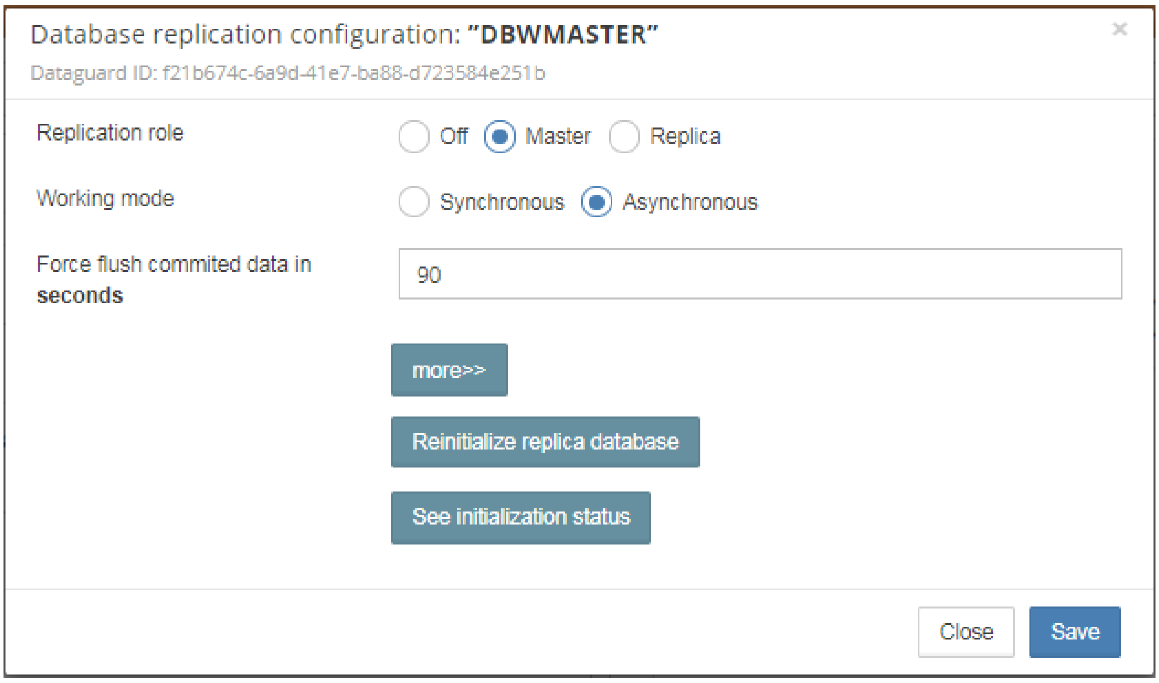
Database replication configuration: test

Replication role: [✓ Off] Master Replica

Asynchronous replication at master

Asynchronous replication writes all changes in the master database to the replication log: the set of files called «replication segments». Replica server pulls these segments and inserts into the replica database.

Previously we have registered H:\dbwmaster.fdb, it is the asynchronous master database in this example. To configure the asynchronous replication on the master side: select replication role: «Master», then «Asynchronous», and click “Save”.



Database replication configuration: "DBWMASTER"
Dataguard ID: f21b674c-6a9d-41e7-ba88-d723584e251b

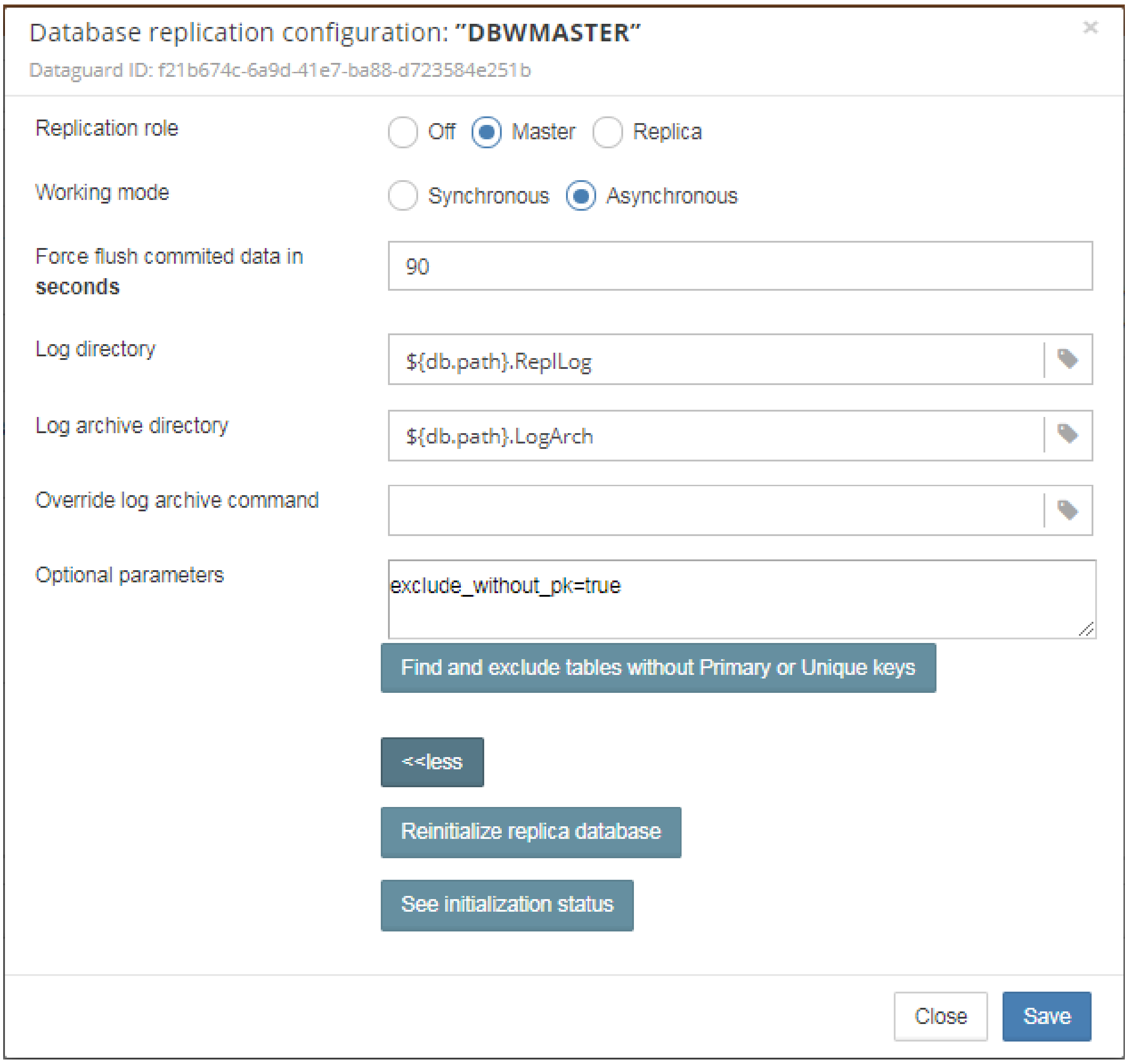
Replication role: Off [✓ Master] Replica

Working mode: Synchronous [✓ Asynchronous]

Force flush commited data in seconds: 90

Replication setup dialog for asynchronous replication

There are several optional parameters which you can change if you open detailed dialog with button “more>>”.



Database replication configuration: **"DBWMASTER"**

Dataguard ID: f21b674c-6a9d-41e7-ba88-d723584e251b

Replication role	Off [✓ Master] Replica
Working mode	Synchronous [✓ Asynchronous]
Force flush commited data in **seconds**	90
Log directory	${db.path}.ReplLog
Log archive directory	${db.path}.LogArch
Override log archive command	
Optional parameters	exclude_without_pk=true

Find and exclude tables without Primary or Unique keys
<<less
Reinitialize replica database
See initialization status
Close
Save

Let's consider all parameters in this dialog – just to give you idea what they do, no need to change them:

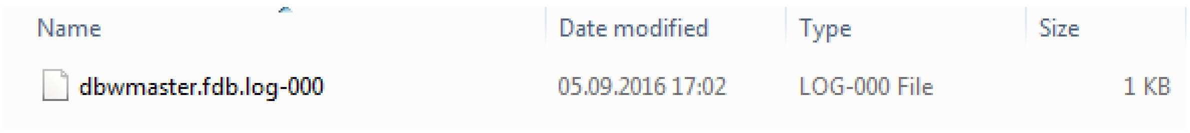
- «Log directory» – folder where operational logs will be stored. It is a system folder, completely operated by Firebird. By default, **no need to change its default value** $ {db.path}.ReplLog" (db.path is where the database is located).
- «Log archive directory» – folder, where archived logs will be stored. According the default value "${db.path}.LogArch", HQbird will create folder "DatabaseName.LogArch" in the folder with the database, so there is **no need to change this parameter**.
- The third parameter ("Override log archive command") is optional, **leave it empty**.

- The fourth parameter «Force flush committed data in, seconds» is also optional, it indicates how often we should move committed data to the archived segments. By default, it is set to 90 seconds.

Please note that replication parameters are initialized at the first connection to the database. That's why you need restart Firebird service (or all connections in case of Classic) after the replication configuration – such restart ensures that replication will start properly.

In this case, the replication log segments will be written first to ${db.path}.ReplLog (db.path is where the database is located – in our example it will be H:\DBWMaster.fdb.ReplLog), and after reaching the maximum segment size, or commit, or another trigger, the default archive command will be started – it will copy archived replication segments to ${db.path}.LogArch (in our example it will be H:\DBWMaster.fdb.LogArch).

After replication's start, you should be able to see replication segment files in the folder specified in «Log directory» immediately after any operation at master database:



Name	Date modified	Type	Size
dbwmaster.fdb.log-000	05.09.2016 17:02	LOG-000 File	1 KB

The operational segments are rotated by the engine, and once each segment is completed, it is copied to archive log. Default segment size is 16Mb. Please note – you don't need to do anything with operational segments!

After the commit and specified timeout of committed data, you will see archived segments in the folder, specified by «Log archive directory».

Archive replication log is essentially the chronologically ordered list of completed operational segments. These files should be imported by replica server into the replica database.

Important! For Linux users – make sure that folder with the database is owned by firebird user. HQbird runs under «firebird» user in Linux, and the folder with the database must have permissions for «firebird» to create logs folder (chown firebird -R /your/database/folder).

How to copy replication segments from master server to the replica server?

There 2 popular ways to copy archived segments from the master server to the replica server(s): through network share and using Cloud Backup on master and Cloud Backup Receiver on replica.

Network share

You can share the folder with archived segments as a network share. In this case, Firebird service should have enough rights to read, write and delete files on that network share. Normally Firebird and HQbird services are started under LocalSystem account, which do not have access to the network shares. Change it to some powerful account (like Domain Admin).

Cloud Backup/Cloud Backup Receiver

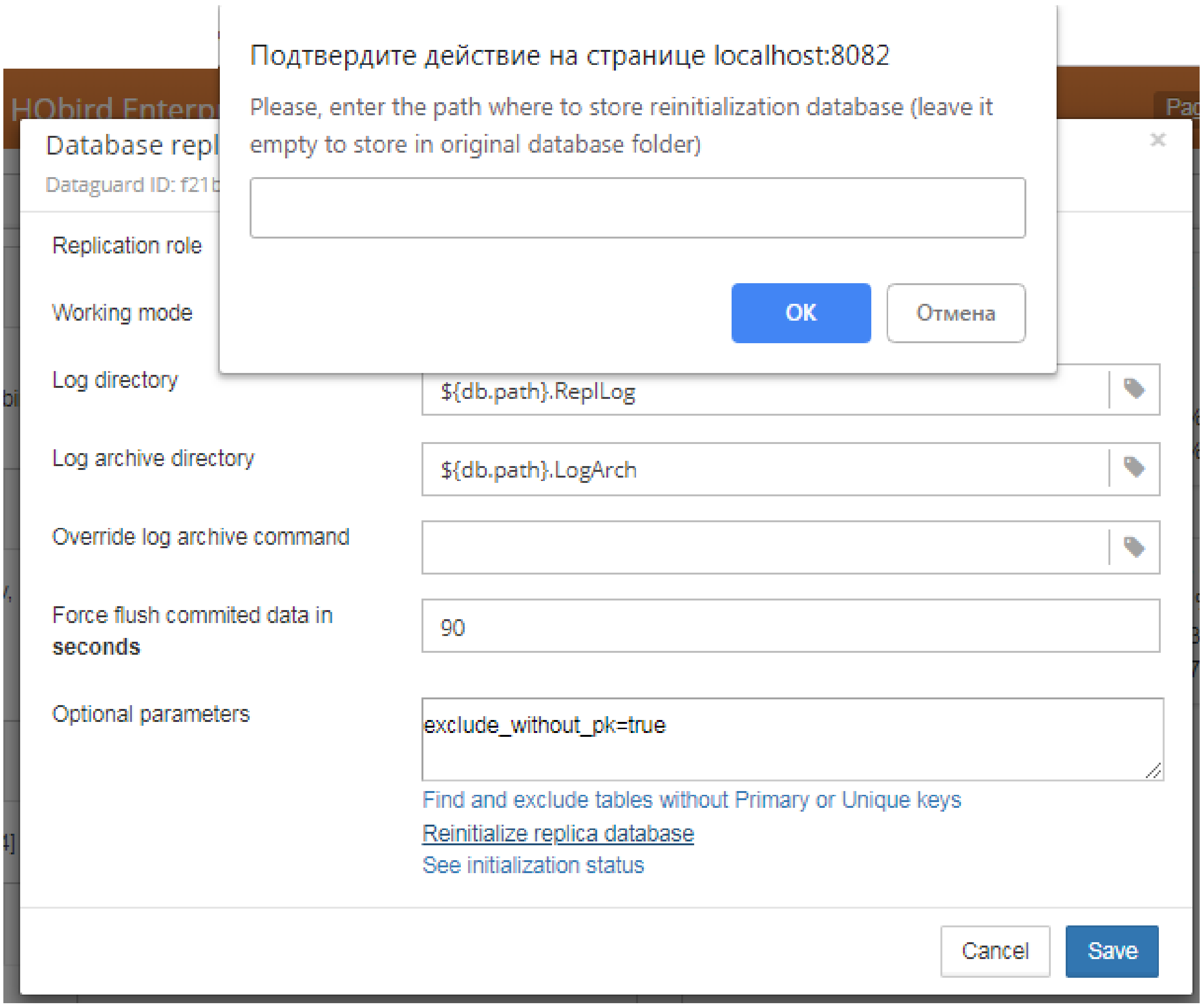
We recommend using HQbird FBDataGuard to send replication segments from the master server to the replica through FTP: it compresses, encrypts and uploads segments to the specified FTP server. On that server, another HQbird FBDataGuard unpacks segments and copies to the necessary folder for further consumption by the replica.

Please read Database: Cloud Backup section for more details how to setup transfer of archived segments between master and replica(s).

Step 2: Create a copy of master database

To start replication we need to create an initial copy of the database file, which will be used as a target for the replication process. Let's refer to such database file as «replica».

Starting with HQbird 2018 R2, the replica will be created automatically in the folder which will you specify in the dialog after clicking on "Reinitialize replica database".


Подтвердите действие на странице localhost:8082

Please, enter the path where to store reinitialization database (leave it empty to store in original database folder)

Log directory	${db.path}.ReplLog
Log archive directory	${db.path}.LogArch
Override log archive command	
Force flush commited data in **seconds**	90
Optional parameters	exclude_without_pk=true

HQbird Enterp
Database repl
Dataguard ID: f21b
Replication role
Working mode
OK
Отмена
Find and exclude tables without Primary or Unique keys
Reinitialize replica database
See initialization status
Cancel
Save

If you have enough space in the folder with the database, just leave the path empty, and click Ok, and replica will be created near the database. Or, you can specify other destination on the local drives with enough free space.

Important! *If there will be not enough free space (less than 105% of the database size), HQbird will not create replica copy – there will appropriate error message.*

In case of default action, the resulted database will be in the same folder with the database. The name of the replica will be DATABASE_NAME.EXT.DD-MMM-YYYY_NNNN.4replica – for example, **employee30.fdb.17-Apr-2018_142507.4replica**

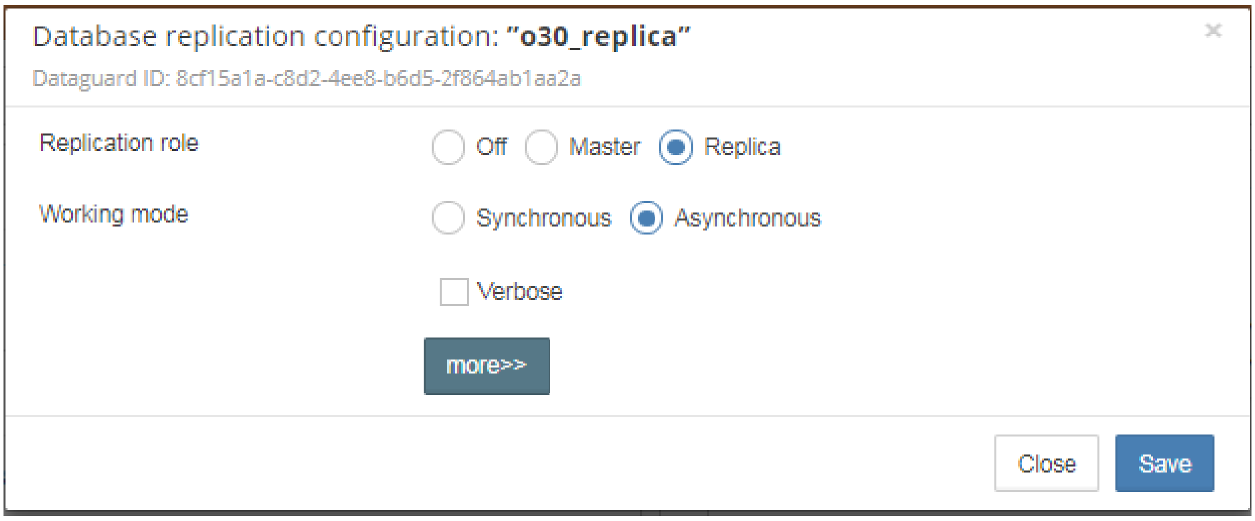
Step 3: Setup database for async replication at the replica(slave) server

After completing the configuration of asynchronous replication on the master server we need to configure it for the replica database at the replica server instance.

First of all, the replica database should be registered in HQbird FBDataGuard. Also, we assume that you have managed to setup transfer of logs with Cloud Backup/Cloud Backup Receiver or with network share.

Please note: the database should have replica database GUID before the registration! This GUID is created automatically if you have used link «Reinitialize replica database», but if you are performing manual registration, don't forget to set it.

Then complete the replication setup – the only required parameter is a path to the folder with archived replication segments, and by default it is already set – HQbird will create folder with logs near the database:


Database replication configuration: "o30_replica"

Dataguard ID: 8cf15a1a-c8d2-4ee8-b6d5-2f864ab1aa2a

Replication role: Off Master [✓ Replica]

Working mode: Synchronous [✓ Asynchronous]

☐ Verbose

So, no need to change anything here, just click Save.

Assuming the replica database is configured in D:\DATABASE\DBWREPLICA.FDB, the HQBird will create folder D:\DATABASE\DBWREPLICA.FDB.LogArch, and replica will import replication segment files from it.

Click «Save» and restart Firebird service (to ensure that replication parameters were applied).

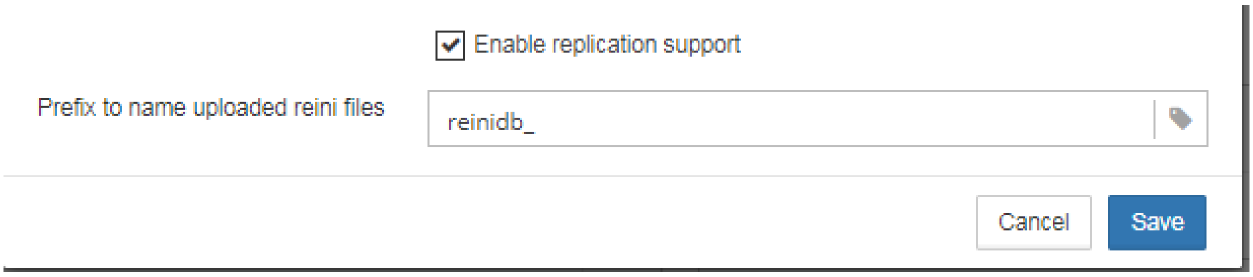
After restart, the replica server will start to consume the replication segments from the folder – please note, after the import all processed segments will be deleted. Also, it will create file with the name {DATABASE-GUIDE} – Firebird stores there some internal information about replication progress.

Note: It is not recommended to store archived replication segments from the different databases into the same folder! Always allocate the separate folder for each pair of master-replica databases!

Automatic initialization and re-initialization of replica

We recommend using Cloud Backup on the master and Cloud Backup Receiver on the replica to implement the transfer and check integrity of the replication segments through FTP. In this case, it is also possible to implement 1-click re-initialization for the replica database.

If Cloud Backup and Cloud Backup Receiver have the following options enabled (by default), HQbird perform the re-initialization automatically, including restart of replica database:



☑ Enable replication support

Prefix to name uploaded reini files: reinidb_

Parameter «Prefix to name uploaded reini files» should be changed if you intend to initialize several copies of the master database through the single folder – in this case set it should be unique for each database.

In case of the single database, no changes are required.

How re-initialization works

If Cloud Backup/Cloud Backup Receiver are configured, it is possible to perform the complete re-initialization with 1 click to «Reinitialize replica database».

Once clicked, the master HQbird will do the following:

1) Ask you where to store copy of the database (by default it is near the master database, click Ok to store database there).
2) Master database will be copied (with nbackup)
3) The created copy of the database will be set to the replica mode
4) md5 hash-sum will be calculated for the copy
5) According the settings in Cloud Backup (Enable replication should be Enabled), master HQbird will upload database to the specified FTP

Next steps will be done by replica HQbird instance:

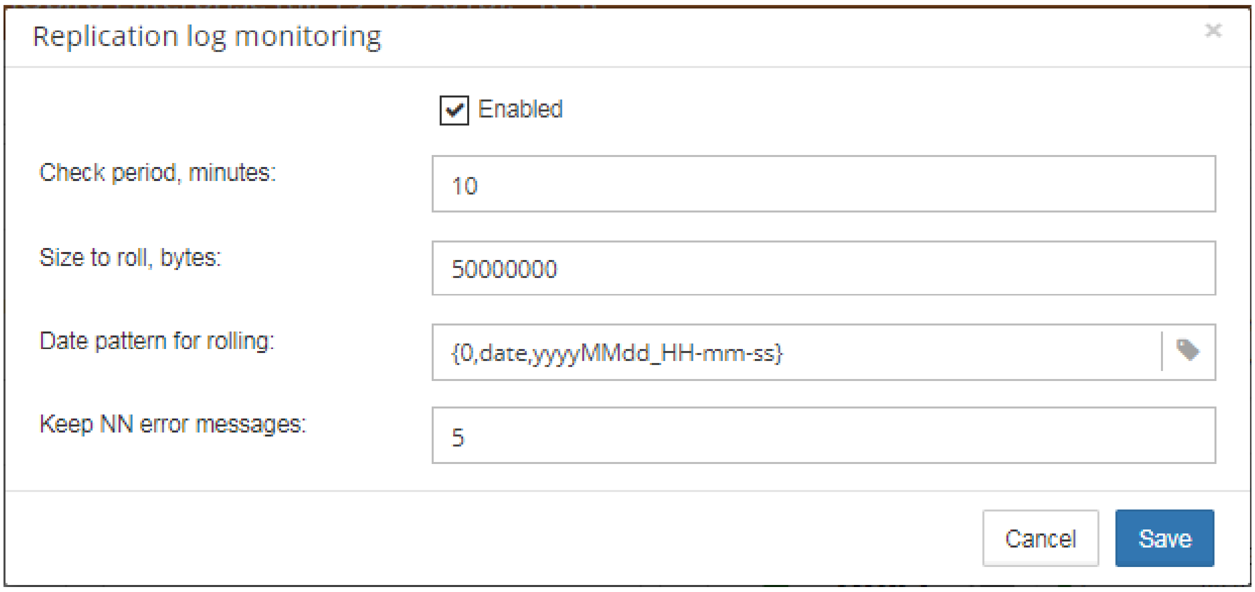
1) Once replica HQbird will notice the reini* files in the incoming FTP folder, Cloud Backup Receiver will start the procedure of re-initialization.
2) Processing if usual arch-segments will be stopped
3) The arrived database will be checked – md5 hash-sum will be calculated and compared with the value in the accompanied report file.
4) The existing replica database will be shutdown to disconnect all users

5) New replica database will be copied over the existing database

Replica is back to the normal mode.

Troubleshooting asynchronous replication

If you have setup asynchronous replication, but it does not work, the first thing is to enable job «Replication Log» on the master and on the replica. This job parses replication.log files, and if there are errors, creates the appropriate alert.

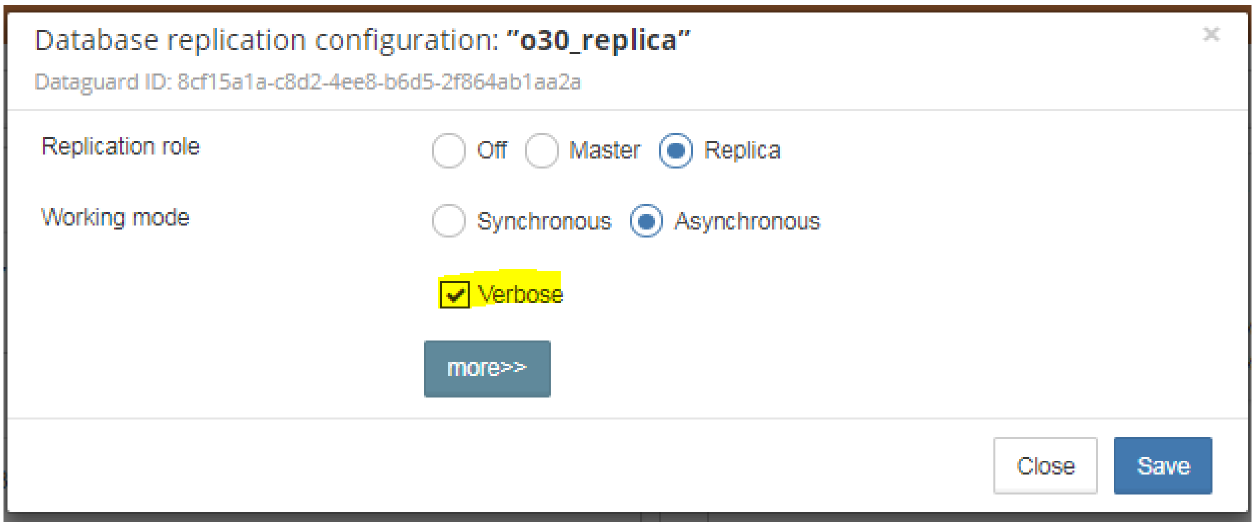


Replication log monitoring

[✓] Enabled

Check period, minutes:	10
Size to roll, bytes:	50000000
Date pattern for rolling:	{0,date,yyyyMMdd_HH-mm-ss}
Keep NN error messages:	5

Also, the good thing is to enable «Verbose» option on the replica, and restart Firebird. Verbose will make Firebird to write a lot of details about replication into the replication.log file (near firebird.log).



Database replication configuration: "o30_replica"

Dataguard ID: 8cf15a1a-c8d2-4ee8-b6d5-2f864ab1aa2a

Replication role: Off Master [✓ Replica]

Working mode: Synchronous [✓ Asynchronous]

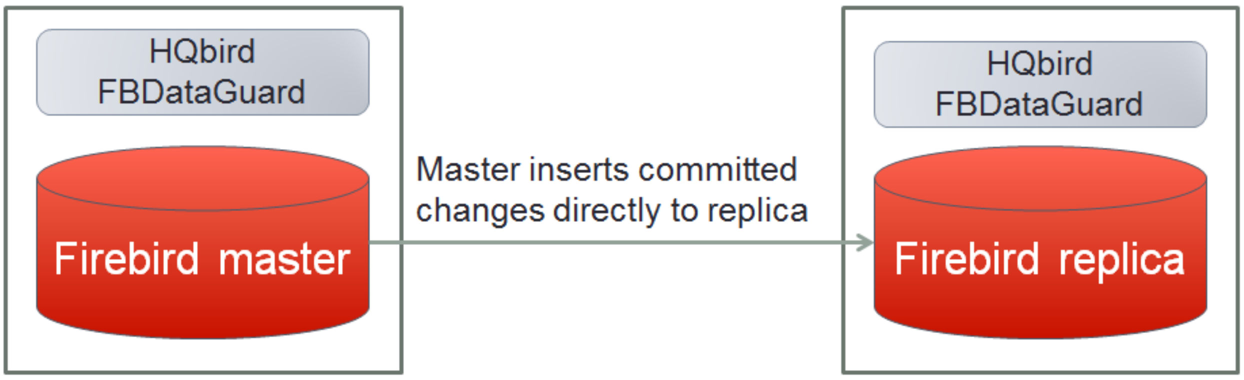
[✓] Verbose

Usually the text of the error is self-explanatory, but since there are some popular questions which occur regularly, we decide to create the table with the list of main problems with asynchronous replication and ways to resolve it.

Problem	Possible reasons and how to resolve
Master part of replication was configured, but folders for operational or archived segments (${dbpath}.LogRepl or ${dbpath}.LogArch) are not created	HQbird creates these folders automatically, but it requires permissions. On Windows: these folders should be on local drives, or HQbird and Firebird services must run with «Run As» with the powerful account (Domain Admin?). On Linux: folders must have permissions for «firebird» user.
Master part of replication was configured; folders for ReplLog and LogArch were created, but nothing in them. Replication.log is empty.	It means that Firebird did not see the replication configuration. Restart Firebird service (all connections in case of Classic) to make read the new configuration.
Master part of replication was configured; there are files **databasename.log-000** in ReplLog folder, but no files in LogArch. Also, could be errors about insufficient space or out of space in replication.log	It means that there is no permission for Firebird to access the LogArch folder and create replication segment files (databasename-logarch.000XXX) there. If LogArch folder on the network share or mounted drive, make sure that Firebird has rights (full access)to access it.
«Verbose» option on replica is enabled, but replication.log is empty or nor created.	Sometimes Firebird cannot create replication.log or even write to already created file. Try to create it manually and apply necessary permissions to it (especially on Linux). Verbose output should be written to the replication.log every 60 seconds even if there is no segments to import.
Master part of replication is Ok, but replica does not consume replication segments. Replication.log file is empty.	Replica did not read the new replication configuration. Restart Firebird.
Master part of replication is Ok, but replica does not consume replication segments. Replication.log contains errors about permissions.	Replica does not have enough permissions to read from the LogArch folder. Set necessary permissions or run replica under powerful account.
Replica has errors in replication.log «Segment NNN is missing»	Check is there such segment on the replica side, and if it is on the master size. If segment has size = 0 on replica, copy it manually or use «Perform fresh backup» checkmark in Cloud Backup.
Replica has errors in replication.log about wrong foreign keys and stopped consume segments	It means that replica copy is desynchronized, so some records do not have the appropriate values in referenced tables for the specified Foreign Key. Replica should be reinitialized. If you see this errors often, please contact IBSurgeon support.

Synchronous replication for Firebird

In case of synchronous replication, master server directly inserts committed changes of the master database to one or more replicas databases:



The main features of the synchronous replication are the following:

- Changes are buffered per transaction, transferred in batches, synchronized at commit
- Practical delay is below1 second
- Follows the master priority of locking
- Replication errors can either interrupt operations or just detach replica
- Replica is available for read-only queries (with caveats)
- Automatic fail-over can be implemented (with HQbird Cluster Manager)

Issues to be considered

- Additional CPU and I/O load on the replica side
- Requires direct and permanent network connection from master to replica(s), 1+Gbps recommended
- Replica can be recreated online, re-initialization of synchronous replication requires stop of master

When to use synchronous replication:

- Custom fail-over cluster solutions with 3+ nodes (especially for web applications)
- Scale performance by moving reads to the separate replica server (report servers, data marts or read-only web representation)
- In combination with asynchronous replication for performance scaling

Steps to setup synchronous replication

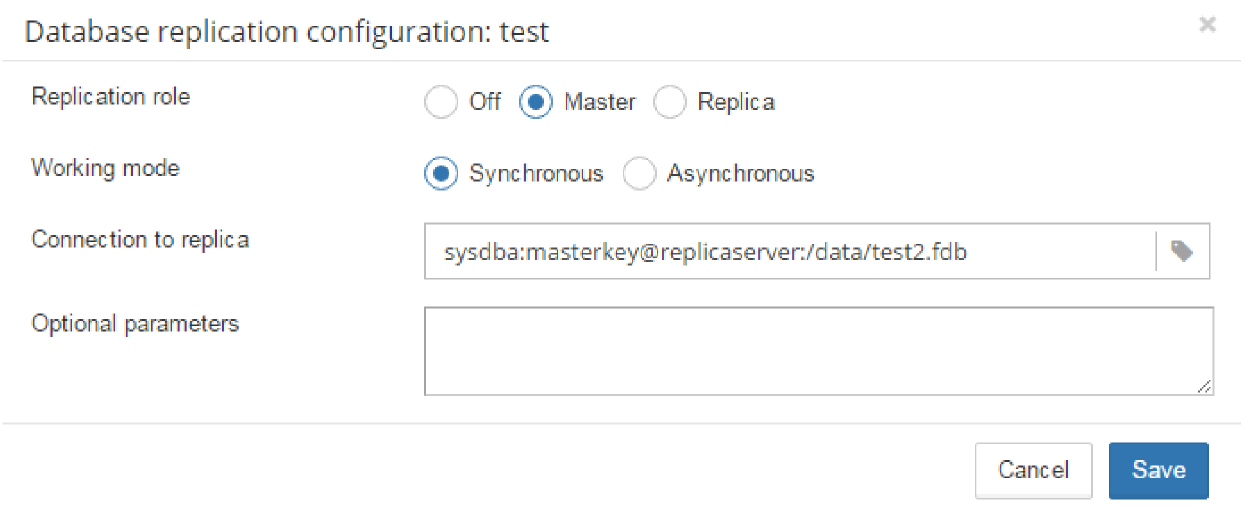
1. Stop Firebird
2. Create a copy of master database file, switch it to replica mode and copy it to the replica server(s)
3. Setup replica server(s) and database(s) for replication with HQbird FBDataGuard

4. Start replica server(s) - before master server!
5. Setup master server and master database for replication with HQBird FBDataGuard
6. Start master server

As you can see, the downtime required for initialization the synchronous replication is bigger than downtime to configure asynchronous replication, because replica database must be online before master's start.

Synchronous replication at master and replica

Synchronous replication is designed to write changes from the master database directly to the replica database. The big advantage of synchronous replication that replication delay can be very small, but the disadvantage is that in the case of the lost connection between master and replica servers there will be gaps in transmitted data.


Database replication configuration: test

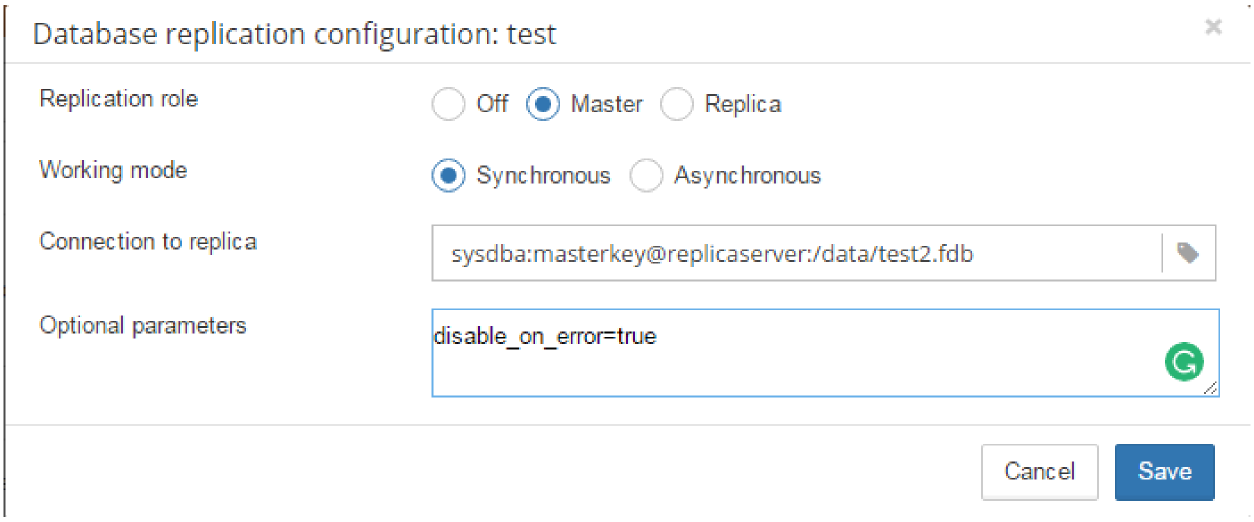
Replication role	Off [✓ Master] Replica
Working mode	[✓ Synchronous] Asynchronous
Connection to replica	sysdba:masterkey@replicaserver:/data/test2.fdb
Optional parameters	

In this example, the synchronous replica database is on the remote server with IP address **replica server** and path **/data/test2.fdb**.

No setup is necessary for synchronous replication on the replica server, except **gfix –replica {master-guid}** for the replica database to switch it to the replica mode.

Replication parameters for testing synchronous replication

In the case of testing synchronous replication of HQbird Enterprise on the production system, we recommend setting parameter disable_on_error to true.



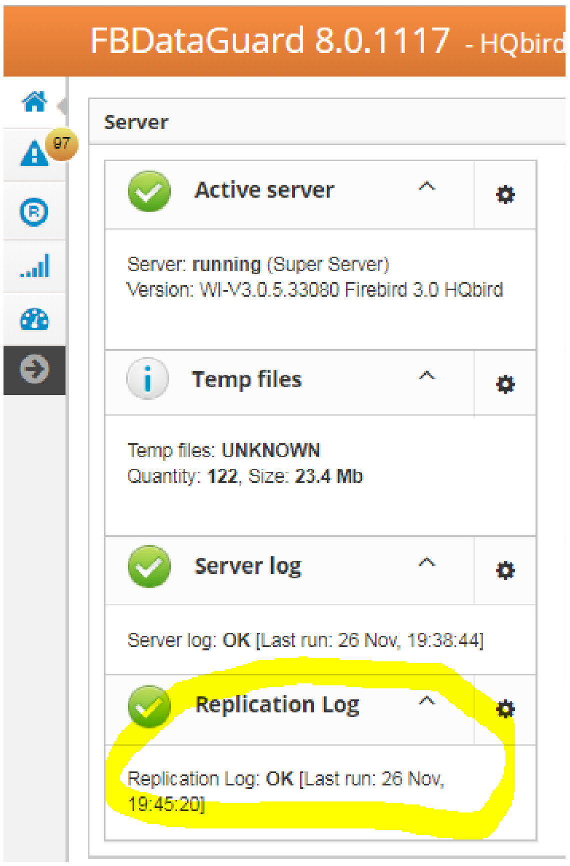
Database replication configuration: test

Replication role	Off [✓ Master] Replica
Working mode	[✓ Synchronous] Asynchronous
Connection to replica	sysdba:masterkey@replicaserver:/data/test2.fdb
Optional parameters	disable_on_error=true

It will switch off replication in case of replication error, and the master server will continue to work without replication.

To reinitialize replication the replication log should be analyzed and all initialization steps should be done again.

Also, please enable job «Replication log» in HQbird FBDataGuard to monitor replication log for errors and warnings:



Active server

Server: **running** (Super Server)
Version: WI-V3.0.5.33080 Firebird 3.0 HQbird

Temp files

Temp files: **UNKNOWN**
Quantity: **122**, Size: **23.4 Mb**

Server log

Server log: **OK** [Last run: 26 Nov, 19:38:44]

Replication Log

Replication Log: **OK** [Last run: 26 Nov, 19:45:20]

How to manually create replica of the database?

If for some reason you cannot use the automatic replica creation (which is available since v. 2018 R2), you can create replica copy of the master database manually.

To start replication we need to create an initial copy of the database file, which will be used as a target for the replication process. Let's refer to such database file as «replica».

Starting with HQbird 2018, it is possible to create replica file without stopping the master server, with nbackup. It is very for asynchronous replication, it also makes possible to create additional replicas online – i.e., without stopping a master.

Of course, it is still possible to create replica with the simple copy process: stop Firebird on master, copy database file, complete setup of replication on the replica, then start Firebird.

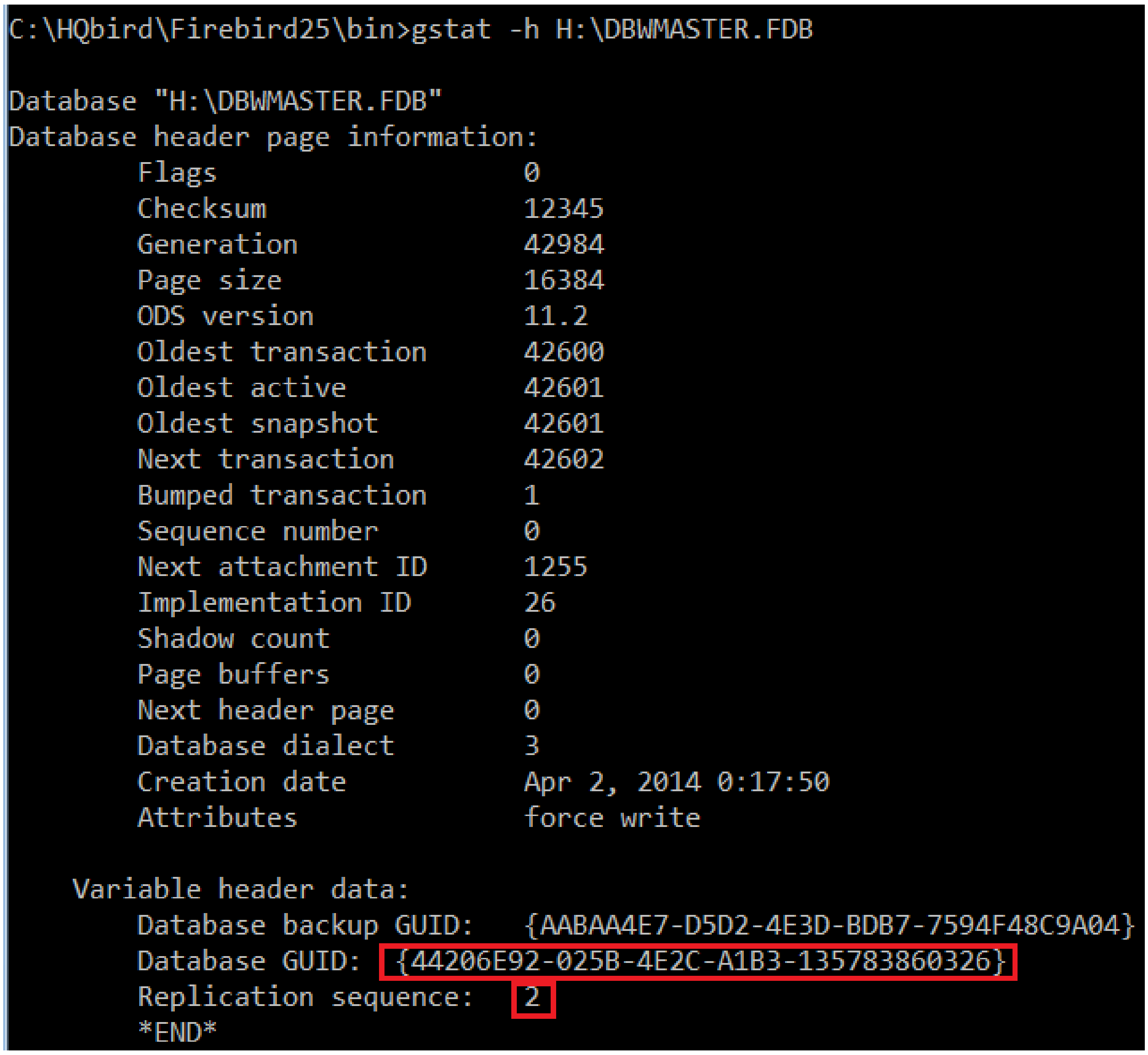
Creating copy online (with nbackup)

Let's consider how to create replica for asynchronous replication using nbackup:

1. apply nbackup lock
   ```
   nbackup -l database_path_name  -user SYSDBA -pass masterkey
   ```
2. copy locked database file to create a replica
   ```
   copy database_path_name replica_path_name
   ```
3. unlock master database
   ```
   nbackup -n database_path_name  -user SYSDBA -pass masterkey
   ```
4. Fixup replica database
   ```
   nbackup -f replica_path_name_name
   ```
5. Switch database to replica mode
   ```
   gfix replica_path_name -replica {DATABASEGUID}  -user
   SYSDBA -pass masterkey
   ```

What is {DATABASEGUID}?

Database GUID is the unique identifier of a master database. To find out {DATABASEGUIDE}, run command gstat–h:



```
C:\HQbird\Firebird25\bin>gstat -h H:\DBWMASTER.FDB

Database "H:\DBWMASTER.FDB"
Database header page information:
        Flags                   0
        Checksum                12345
        Generation              42984
        Page size               16384
        ODS version             11.2
        Oldest transaction      42600
        Oldest active           42601
        Oldest snapshot         42601
        Next transaction        42602
        Bumped transaction      1
        Sequence number         0
        Next attachment ID      1255
        Implementation ID       26
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        3
        Creation date           Apr 2, 2014 0:17:50
        Attributes              force write

    Variable header data:
        Database backup GUID:   {AABAA4E7-D5D2-4E3D-BDB7-7594F48C9A04}
        Database GUID:  {44206E92-025B-4E2C-A1B3-135783860326}
        Replication sequence:   2
        *END*
```

To switch database to the replica mode run the following command:

```
gfix disk:\path\mydatabase.fdb  -replica {guid} -user SYSDBA
-pass masterkey
```

How to set replica database to the master mode

To switch database to the normal (master) mode run the same command with the empty {} instead of database GUID:

```
gfix disk:\path\mydatabase.fdb  -replica {} -user SYSDBA -pass
masterkey
```

Note: If you don't see Database GUID in gstat –h output, connect to the master database using Firebird binaries from HQbird distribution (with isql or any other application), and run gstat –h again.

How to distinguish master database from replica

Using gstat –h

If you run gstat –h database_name, the output will contain the keyword «replica» in Attributes section for database configured as replica:

```
Database "D:\O30.FDB"
Gstat execution time Mon Nov 26 17:47:07 2018

Database header page information:
        Flags                   0
        Generation              187842
        System Change Number    15
        Page size               8192
        ODS version             12.0
        Oldest transaction      173630
        Oldest active           185440
        Oldest snapshot         185440
        Next transaction        185441
        Sequence number         0
        Next attachment ID      24975
        Implementation          HW=AMD/Intel/x64 little-endian OS=Windows
CC=MSVC
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        3
        Creation date           Jan 11, 2017 15:12:20
        Attributes              replica

    Variable header data:
        Database backup GUID:   {37E7918F-5478-43CF-E3B2-D80B0E7D3F63}
        Sweep interval:         0
        Database GUID:  {BBBD2881-ACDE-4636-CEB2-7EE31AF66CC3}
        Replication master GUID: {BBBD2881-ACDE-4636-CEB2-7EE31AF66CC3}
        *END*
Gstat completion time Mon Nov 26 17:47:07 2018
```

For master database there is no special marks in Attributes.

With SQL query to the context variable SYSTEM.REPLICA

There is a new context variable in the SYSTEM area, which contains the information about database status:

```
SQL> select RDB$GET_CONTEXT('SYSTEM', 'REPLICA') from rdb$database;

RDB$GET_CONTEXT

===============================================================

FALSE
```

Optional parameters for replication

It is possible to specify several additional parameters for fine tuning of the replication process. These parameters can be specified in the «Optional parameters» of replication setup dialog.

1) Size of the local buffer used to accumulate replication events that can be deferred until the transaction commit/rollback. The bigger this value the less network round-trips between master and slave hosts are performed. However, it costs longer replication "checkpoints" (time to synchronize the original database with its replica).

buffer_size = 1048576

2) If enabled, any error during replication causes the master to stop replicating changes and continue working normally. Otherwise (the default behavior), the master reports an error.

disable_on_error = false

3) If enabled, replicated records are RLE-compressed before transmission and decompressed on the slave side. It reduces the traffic and (indirectly) a number of round-trips at the cost of extra CPU cycles on both sides.

compress_records = false

4) If enabled, conflicting records in the target database are modified to match records in the master database. In particular:

- if there's an insert and the target record exists, it gets updated;
- if there's an update and the target record does not exist, it gets inserted;
- if there's a delete and the target record does not exist, it gets ignored.

master_priority = false

5) Pattern (regular expression) that defines what tables must be included into replication. By default, all tables are replicated.

include_filter

6) Pattern (regular expression) that defines what tables must be excluded from replication. By default, all tables are replicated.

exclude_filter

7) If enabled, tables without primary key (or unique index) excluded from replication. By default, all tables are replicated.

exclude_without_pk = false

8) Program (complete command line with arguments) that is executed when the current replication session notices a critical error. This command is executed once per every failed replication session. Please note that the program is executed synchronously and the server is waiting for its completion before continuing its operations.

alert_command

9) Prefix for replication log file names. It will be automatically suffixed with an ordinal sequential number. If not specified, database filename (without path) is used as a prefix.

log_file_prefix

10) Maximum allowed size for a single replication segment. It must at least double the specified buffer_size.

log_segment_size = 16777216

11) Maximum allowed number of full replication segments. Once this limit is reached, the replication process is delayed for log_archive_timeout seconds (see below) to allow the archiving to catch up. If any of the full segments is not archived and marked for reuse during the timeout, the replication fails with an error.

Zero means an unlimited number of segments pending archiving.

log_segment_count = 8

7. Performance enhancements

7.1 Pool of external connections

HQbird 2018 supports a pool of external connections for Firebird 2.5 and (from 2018R3) in Firebird 3. This pool allows running parallel EXECUTE ON EXTERNAL statements to external Firebird databases.

Please note – this pool is allocated per Firebird instance.

The feature is managed in the firebird.conf:

```
# ============================
# Settings of External Connections Pool
# ============================

# Set the maximum number of inactive (idle) external connections to retain at
# the pool. Valid values are between 0 and 1000.
# If set to zero, pool is disabled,
# i.e. external connection is destroyed immediately after the use.
#
# Type: integer
#
#ExtConnPoolSize = 0

# Set the time before destroying inactive external connection, seconds.
# Valid values are between 1 and 86400.
#
# Type: integer
#
#ExtConnPoolLifeTime = 7200
```

From the application point of view, no additional steps are required to use or do not use – it is enabled or disabled in the server configuration, and absolutely seamless for the applications.

The following commands exist to manage pool:

//changes the pool size
```
  ALTER EXTERNAL CONNECTIONS POOL SET SIZE N;
```
//example – this command sets the size of a pool to 190 connections.
```
 ALTER EXTERNAL CONNECTIONS POOL SET SIZE 190
```
//changes the lifetime of the pooled connection
```
ALTER EXTERNAL CONNECTIONS POOL SET LIFETIME N SECOND | MINUTE |
HOUR
```
//example - this command limits the lifetime of a connection
//in the pool to 1 hour.
```
ALTER EXTERNAL CONNECTIONS POOL SET LIFETIME 1 HOUR;
```
//clear all pooled connections.
```
ALTER EXTERNAL CONNECTIONS POOL CLEAR ALL
```
//clear the oldest connection in the pool
```
ALTER EXTERNAL CONNECTIONS POOL CLEAR OLDEST
```

To get information about pool status, new context variables were introduced. The following example demonstrates their usage

```
SELECT CAST(RDB$GET_CONTEXT('SYSTEM', 'EXT_CONN_POOL_SIZE') AS
INT) AS POOL_SIZE,        CAST(RDB$GET_CONTEXT('SYSTEM',
'EXT_CONN_POOL_IDLE_COUNT') AS INT) AS POOL_IDLE,
CAST(RDB$GET_CONTEXT('SYSTEM', 'EXT_CONN_POOL_ACTIVE_COUNT') AS
INT) AS POOL_ACTIVE,        CAST(RDB$GET_CONTEXT('SYSTEM',
'EXT_CONN_POOL_LIFETIME') AS INT) AS POOL_LIFETIME   FROM
RDB$DATABASE;
```

7.2 Cached prepared statements

HQbird 2018 has the feature to improve the performance of Firebird (version 3+ only!) engine in case of the many frequent and fast SQL queries: server-side cache of prepared SQL statements.

This feature can be enabled in firebird.conf with the parameter DSQLCacheSize:

```
# Size of DSQL statements cache.
# Maximum number of statements to cache.
# Use with care as it is per-attachment and could lead to big
memory usage.
# Value of zero disables caching.
# Per-database configurable.
# Type: integer
#DSQLCacheSize = 0
```

The number specifies how many recent queries for each database connection to cache.

To apply the new value, Firebird restart is required.

By default, cache of prepared statements is 0, it means OFF. We recommend to carefully enable it: start with values like 4,8,16, to find the best performance effect.

Please note: enabling cache of prepared statements increases the memory usage.

Support contacts

We will answer all your questions regarding HQbird FBDataGuard. Please send inquiries to support@ib-aid.com.

Please note that customers with active Firebird Support have the priority in the technical support: https://ib-aid.com/en/firebird-support-service/